

Übungen zur Vorlesung Computermathematik

Serie 2

Aufgabe 2.1. Schreiben Sie eine Funktion `tensor`, die für gegebenes $n \in \mathbb{N}$ den Schachbrett-Tensor $B \in \mathbb{N}^{n \times n \times n}$ mit

$$B_{jkl} = \begin{cases} 0 & \text{falls } j + k + \ell \text{ gerade} \\ 1 & \text{falls } j + k + \ell \text{ ungerade} \end{cases}$$

Die Funktion soll auf zwei unterschiedliche Arten programmiert werden: erstens unter Vermeidung von Schleifen und Verwendung geeigneter MATLAB-Arithmetik an deren Stelle; zweitens mithilfe von Schleifen und skalarer Arithmetik. **Hinweis:** Der Befehl `A = zeros(n,n,n)` erstellt ein drei dimensionales Array mit 0 Einträgen. Der Koeffizientenzugriff funktioniert analog zu Matrizen mittels `A(i,j,k)`.

Aufgabe 2.2. Schreiben Sie eine MATLAB-Funktion `ishermitian`, die überprüft ob eine gegebene Matrix $A \in \mathbb{C}^{n \times n}$ hermitesch ist, d.h., es gilt $A = A^H := \overline{A}^T$ bzw. $a_{ij} = \overline{a_{ji}}$ für alle $0 \leq i, j \leq n$. Vermeiden Sie Schleifen, und verwenden Sie geeignete Matrix/Vektor-Funktionen an deren Stelle.

Aufgabe 2.3. Das Polynom $p(x) = \sum_{j=0}^n a_j x^j$ sei gegeben in Form seines Koeffizientenvektors $a \in \mathbb{C}^{n+1}$. Schreiben Sie eine MATLAB-Funktion, die a übernimmt und den Koeffizientenvektor der Ableitung p' zurückgibt.

Die Funktion soll auf zwei unterschiedliche Arten programmiert werden: erstens unter Vermeidung von Schleifen und Verwendung geeigneter Vektor-Funktionen und Arithmetik an deren Stelle; zweitens mithilfe von Schleifen und skalarer Arithmetik. Ihre Funktion soll für Spalten- und Zeilenvektoren a funktionieren und stets einen Spaltenvektor zurückliefern; siehe z.B. `help reshape` Überlegen Sie sich, wie Sie ihren Code auf Korrektheit testen können! Was sind geeignete Test-Beispiele?

Aufgabe 2.4. Schreiben Sie eine MATLAB-Funktion, die für zwei gegebene Polynome $p(x)$ und $q(x)$ das Ergebnis $r(x) = p(x) + q(x)$ berechnet und den Koeffizientenvektor $r \in \mathbb{C}^{n+1}$ zurückgibt. $r(x)$ soll ein Polynom von kleinstmöglichem Grad sein, d.h. für den Leitkoeffizienten muss $r_{n+1} \neq 0$ gelten. Die Funktion soll auf zwei unterschiedliche Arten programmiert werden: erstens unter Vermeidung von Schleifen und Verwendung geeigneter MATLAB-Arithmetik an deren Stelle; zweitens mithilfe von Schleifen und skalarer Arithmetik. Überlegen Sie sich, wie Sie ihren Code auf Korrektheit testen können! Was sind geeignete Test-Beispiele?

Aufgabe 2.5. Das Polynom $p(x) = \sum_{j=0}^n a_j x^j$ sei gegeben in Form seines Koeffizientenvektors $a \in \mathbb{C}^{n+1}$. Es sei $x = (x_{jk}) \in \mathbb{C}^{M \times N}$ eine Matrix von Auswertungsstellen. Schreiben Sie eine MATLAB-Funktion, die die Matrix $(p(x_{jk})) \in \mathbb{C}^{M \times N}$ der Auswertungen berechnet und ausgibt. Ihre Funktion soll für Spalten- und Zeilenvektoren a funktionieren. Die Funktion soll auf zwei unterschiedliche Arten programmiert werden: erstens unter Vermeidung von Schleifen und Verwendung geeigneter Vektor-Funktionen und Arithmetik an deren Stelle; zweitens mithilfe von Schleifen und skalarer Arithmetik. Überlegen Sie sich, wie Sie ihren Code auf Korrektheit testen können! Was sind geeignete Test-Beispiele?

Hinweis: Sie können `reshape` verwenden, um den Fall einer Matrix x auf den eines Vektors zurückzuführen. Beachten Sie bei Ihrer Lösung, dass die Auswertungsstellen komplexwertig sein können.

Aufgabe 2.6. MATLAB bietet viele Möglichkeiten um die Laufzeit von Funktionen oder Operationen zu messen. Eine einfache Variante ist `tic-toc`. Die Zeitmessung startet dabei mit dem Befehl `tic` und mittels `t=toc` wird vergangene Zeit in der Variable `t` gespeichert; siehe `help tic` bzw. `help toc`. Schreiben Sie eine MATLAB-Funktion, welche die Laufzeiten von mindestens 2 der vorhergehenden Aufgaben bestimmt. Vergleichen Sie dabei jeweils die Laufzeiten Ihrer Implementierung mittels Schleifen bzw. Vektor-Arithmetik für verschiedene Problemgrößen. Die Ergebnisse sollen dabei mittels `fprintf` ausgegeben werden.

Aufgabe 2.7. Das Integral $\int_a^b f dx$ einer stetigen Funktion $f : [a, b] \rightarrow \mathbb{R}$ kann man durch eine sogenannte Quadraturformel

$$\int_a^b f dx \approx \sum_{j=1}^n \omega_j f(x_j)$$

approximieren, wobei man sich einen Vektor $x \in [a, b]^n$ mit $x_1 < \dots < x_n$ vorgibt und die Funktion f (formal = theoretisch) durch ein Polynom $p(x) = \sum_{j=1}^n a_j x^{j-1}$ vom Grad $\leq n-1$ mit $p(x_j) = f(x_j)$ für alle $j = 1, \dots, n$ approximiert. Die Gewichte ω_j lassen sich aus der Forderung berechnen, dass

$$\int_a^b q dx = \sum_{j=1}^n \omega_j q(x_j) \quad \text{für alle Polynome } q \text{ vom Grad } \leq n-1$$

gilt. Dies ist nämlich äquivalent zur Lösung des linearen Gleichungssystems

$$\frac{b^{k+1}}{k+1} - \frac{a^{k+1}}{k+1} = \int_a^b x^k dx = \sum_{j=1}^n \omega_j x_j^k \quad \text{für alle } k = 0, \dots, n-1.$$

Warum ist das so? Schreiben Sie eine Funktion `integrate`, die den (Zeilen- oder Spalten-) Vektor $x \in [a, b]^n$ und den Funktionswert-Vektor $f(x)$ übernimmt und den approximativen Wert des Integrals zurückgibt. Dazu bauen Sie das lineare Gleichungssystem möglichst effizient auf und lösen dieses mittels Backslash-Operator. Mit Hilfe des Ergebnisvektors $\omega \in \mathbb{R}^n$ ergibt sich das approximative Integral als Skalarprodukt mit dem $f(x)$ -Vektor. Überlegen Sie sich, wie Sie ihren Code auf Korrektheit testen können! Was sind geeignete Test-Beispiele? Vermeiden Sie Schleifen, und verwenden Sie geeignete Vektor-Funktionen und Arithmetik an deren Stelle.

Aufgabe 2.8. Es sei $L \in \mathbb{R}^{n \times n}$ eine untere Dreiecksmatrix mit Einträgen $\ell_{jj} \neq 0$ für alle $j = 1, \dots, n$, d.h. L hat die Form

$$L = \begin{pmatrix} \ell_{11} & 0 & \cdots & \cdots & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \ell_{n-1,1} & \ell_{n-1,2} & \cdots & \ell_{n-1,n-1} & 0 \\ \ell_{n1} & \ell_{n2} & \cdots & \ell_{n,n-1} & \ell_{nn} \end{pmatrix}$$

Wegen $\det(L) = \prod_{j=1}^n \ell_{jj} \neq 0$ ist L dann invertierbar, und die Inverse lässt sich rekursiv wie folgt berechnen: Wir schreiben L in Block-Form

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}$$

mit $L_{11} \in \mathbb{R}^{p \times p}$, $L_{21} \in \mathbb{R}^{q \times p}$ und $L_{22} \in \mathbb{R}^{q \times q}$, wobei $p + q = n$ gilt. Üblicherweise wählt man $p = n/2$ für gerades n und $p = (n-1)/2$ für ungerades n . Man beachte, dass L_{11} und L_{22} wieder reguläre untere Dreiecksmatrizen sind. Elementares Nachrechnen zeigt, dass die Inverse die Block-Form

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1} L_{21} L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}$$

besitzt. Schreiben Sie eine Funktion `invertL`, die die Inverse L^{-1} auf die beschriebene Weise rekursiv berechnet. Sie können die Korrektheit Ihrer Funktion mithilfe von `inv` überprüfen. Vermeiden Sie Schleifen, und verwenden Sie geeignete Vektor-Funktionen und Arithmetik an deren Stelle.