Dirk Praetorius
Winfried Auzinger

Sommersemester 2017
15.03.2017

**Übungen zur Vorlesung
Computermathematik**

**Serie 2**

**Aufgabe 2.1.** Schreiben Sie eine Funktion, die für einen Vektor $x \in \mathbb{C}^n$ und $1 \le p < \infty$ die $\ell_p$-Norm

$$\|x\|_p := \Big(\sum_{j=1}^n |x_j|^p\Big)^{1/p}$$

berechnet und 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.

**Aufgabe 2.2.** 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

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

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.

**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 kleinstmöglichen 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 $\big(p(x_{jk})\big) \in \mathbb{C}^{M\times N}$ 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 min. 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 `fprinft` ausgegeben werden.

**Aufgabe 2.7.** Das Integral $\int_a^b f\,dx$ einer stetigen Funktion $f : [a,b] \to \mathbb{R}$ 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 $x_1 < \cdots < x_n$ vorgibt und die Funktion $f$ (formal = theoretisch) durch ein Polynom $p(x) = \sum_{j=1}^n a_j x^{j-1}$ vom Grad $\le n-1$ mit $p(x_j) = f(x_j)$ für alle $j = 1, \ldots, n$ approximiert. Die Gewichte $\omega_j$ lassen sich aus der Forderung berechnen, dass

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

gilt. Dies ist nämlich äquivalent zur Lösung des linearen Gleichungssystems

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

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.** Schreiben Sie eine MATLAB Funktion `saveMatrix`, welche die Matrix $A \in \mathbb{R}^{M \times N}$ nimmt und via `fprintf` (siehe auch `help fopen`) in eine ASCII Datei `matrix.dat` speichert. Verwenden Sie `%1.16e` für die Matrixkoeffizienten! (Warum macht das Sinn?) Optional, habe die Funktion den String `name` als Eingabeparameter, wobei die Matrix in der ASCII Datei `name.dat` gespeichert werde. Um Ihren Code zu verifizieren, schreiben Sie ein MATLAB Skript, welches eine Zufallsmatrix $A \in \mathbb{R}^{M \times N}$ generiert und sie in einer ASCII Datei `A.dat` speichert. Laden Sie die Matrix via `B = load('A.dat')` und checken Sie, ob $A$ und $B$ übereinstimmen.