Winfried Auzinger
Dirk Praetorius

Sommersemester 2012
14. Juni 2012

Übungen zur Vorlesung Computermathematik

Serie 11

Aufgabe 11.1*. Die Spaltensummennorm einer Matrix $A \in \mathbb{R}^{m\times n}$ ist durch

$$\|A\|_S := \max_{k=1,\dots,n} \sum_{j=1}^{n} |A_{jk}|$$

gegeben. Schreiben Sie eine Funktion `spaltensummennorm`, die für eine gegebene Matrix A beliebiger Dimension die Spaltensummennorm berechnet. Verwenden Sie nur skalare Arithmetik und geeignete Schleifen. Speichern Sie Ihre Datei unter `spaltensummennorm.m` ins Verzeichnis `serie11`.

Aufgabe 11.2*. Ändern Sie Ihre Funktion aus Aufgabe 11.1 dahingehend ab, dass sie ohne Schleifen auskommt und nur MATLAB-Vektorarithmetik (bzw. Vektorfunktionen) verwendet. Speichern Sie Ihre Datei unter `spaltensummennorm2.m` ins Verzeichnis `serie11`.

Aufgabe 11.3*. Schreiben Sie eine Funktion `cut`, die einen Vektor $x \in \mathbb{R}^N$ und eine Schranke $S \geq 0$ übernimmt und einen Vektor $y \in \mathbb{R}^n$ zurückgibt, aus dem alle Einträge x_j mit $|x_j| > S$ gestrichen werden. Aufruf mit $S = 1$ und $x = (0, 2, 1, 4, 5, 0, 0, 1, 2) \in \mathbb{R}^9$ liefere also $y = (0, 1, 0, 0, 1) \in \mathbb{R}^5$ zurück. Speichern Sie Ihre Datei unter `cut.m` ins Verzeichnis `serie11`.

Aufgabe 11.4*. Das Integral $\int_a^b dx$ 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 $x \in \mathbb{R}^n$ 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 $\leq n-1$ 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 f und den Vektor $x \in \mathbb{R}^n$ 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.

Aufgabe 11.5. Die Frobeniusnorm einer Matrix $A \in \mathbb{R}^{m\times n}$ ist durch

$$\|A\|_F := \Big(\sum_{j=1}^{m}\sum_{k=1}^{n} A_{jk}^2\Big)^{1/2}$$

definiert. Schreiben Sie eine Funktion `frobeniusnorm`, die für eine gegebene Matrix A die Frobeniusnorm berechnet. Verwenden Sie nur skalare Arithmetik und geeignete Schleifen. Speichern Sie Ihre Datei unter `frobenius.m` ins Verzeichnis `serie11`.

Aufgabe 11.6. Ändern Sie Ihre Funktion aus Aufgabe 11.5 dahingehend ab, dass sie ohne Schleifen auskommt und nur MATLAB-Vektorarithmetik verwendet. Speichern Sie Ihre Datei unter `frobenius2.m` ins Verzeichnis `serie11`.

Aufgabe 11.7. Gegeben sei eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$, d.h. $u_{jk} = 0$ für $j > k$, mit $u_{jj} \neq 0$ für alle $j = 1, \ldots, n$. Für gegebenes $b \in \mathbb{R}^n$ existiert dann eine eindeutige Lösung $x \in \mathbb{R}^n$ von $Ux = b$ (Warum?). Schreiben Sie eine Funktion `solveU`, die die Lösung berechnet. Dabei sollen lediglich Schleifen und Arithmetik verwendet werden. Versuchen Sie, möglichst viele Schleifen durch Verwendung von Vektorarithmetik zu umgehen. Sie können Ihre Funktion mit Hilfe des MATLAB Backslash-Operators \ verifizieren. Speichern Sie Ihre Datei unter `solveU.m` ins Verzeichnis `serie11`.

Aufgabe 11.8. Für $x > 0$ konvergiert die Folge

$$x_1 := \frac{1}{2}(1+x), \quad x_{n+1} := \frac{1}{2}\Big(x_n + \frac{x}{x_n}\Big) \quad \text{für } n \geq 1$$

gegen $\sqrt{x}$. Schreiben Sie eine Funktion `sqrt2`, die für gegebene $x > 0$ und $\tau > 0$ als Ergebnis das erste Folgenglied $y = x_n$ berechnet, für das gilt

$$\frac{|x_n - x_{n+1}|}{|x_n|} \leq \tau \quad \text{oder} \quad |x_n| \leq \tau.$$

Die Rückgabe der Funktion sei gerade der umgekehrte Vektor $(x_n, x_{n-1}, \ldots, x_1)$ der Iterierten. Speichern Sie Ihre Datei unter `sqrt2.m` ins Verzeichnis `serie11`.

Aufgabe 11.9. Schreiben Sie eine Funktion `sqrt_bisec`, die für gegebene $x > 0$ und $\tau > 0$ die Wurzel $\sqrt{x}$ mit Hilfe des Bisektionsverfahrens aus der Vorlesung approximiert, sodass der approximative Wert a die Fehlerschranke $|a - \sqrt{x}| \leq \tau$ erfüllt: Welche Funktion $f : [0, \infty) \to \mathbb{R}$ wählt man? Wie wählt man das Startintervall $[a, b]$? Schreiben Sie ein Skript `sqrt_compare`, mit dem Sie die Anzahl der Iterationen von `sqrt2` mit der Anzahl der Iterationen von `sqrt_bisec` vergleichen. Geben Sie dazu jeweils tabellarisch den Iterationsschritt, den Wert x_n und den zugehörigen

Fehler $|\sqrt{x} - x_n|$ für beide Verfahren aus. Speichern Sie Ihre Datei unter `sqrt_compare.m` ins Verzeichnis `serie11`.

Aufgabe 11.10. Modifizieren Sie das Test-Skript aus der vorausgegangenen Aufgabe so, dass anstelle der Ausgabe in der Shell die Ausgabe in eine Datei erfolgt. Diese enthalte den vollständigen LaTeX-Code für eine Tabelle der Form

Toleranz $\tau = \ldots$		
ℓ	x_ℓ	$\lvert\sqrt{x} - x_\ell\rvert$
1	$\cdots$	$\cdots$
2	$\cdots$	$\cdots$

Schreiben Sie das entsprechende Skript `sqrt_tab` und binden Sie die automatisch generierte Tabelle in ein LaTeX-Dokument `sqrt_tab.tex` ein. Beachten Sie, dass der einfache Backslash `\` in `fprintf` durch den doppelten Backslash `\\` kodiert werden muss. Dazu kann ggf. eine Zeichen-Ersetzungsfunktion hilfreich sein, siehe `help strfun`.