Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2008/09
10. Dezember 2008

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

Die Aufgaben mit Stern (*) sind bis zur nächsten Übung vorzubereiten und werden dort abgeprüft. Die übrigen Aufgaben dienen nur Ihrer Übung und mir als zusätzliche Grundlage für den Prüfungsstoff in den schriftlichen Tests. Kopieren Sie bitte den Source-Code in ein Unterverzeichnis `serie08` *Ihres Home-Verzeichnisses. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit* `matlab` *auf der* `lva.student.tuwien.ac.at` *interpretiert werden können. In den folgenden Aufgaben sollen* **Arithmetik, Zählschleifen** *und* **rekursive Funktionen** *geübt werden.*

Aufgabe 71*. Man schreibe eine Matlab-Funktion `[phi,psi]=bogenmass(theta)`, die einen im Gradmaß gegebenen Winkel θ ins Bogenmaß umrechnet. Es wird sowohl der Wert ϕ als auch der reduzierte Wert $\psi := \phi - 2k\pi \in [0, 2\pi)$ zurückgegeben, wobei $k \in \mathbb{N}$ geeignet gewählt ist. Verwenden Sie die Matlab-Funktion `mod` zur Berechnung von ψ. Speichern Sie den Source-Code unter `bogenmass.m` in das Verzeichnis `serie08`.

Aufgabe 72*. Man schreibe eine Funktion `maxcount`, die von einem Vektor $x \in \mathbb{R}^n$ das Maximum zurückliefert und die Anzahl, wie oft dieses im Vektor vorkommt. Realisieren Sie die Funktion ein Mal mit und ein Mal ohne Verwendung von Schleifen. Speichern Sie den Source-Code unter `maxcount.m` in das Verzeichnis `serie08`.

Aufgabe 73*. Die Fibonacci-Folge ist definiert durch $x_0 := 0, x - 1 := 1$ und $x_{n+1} = x_n + x_{n-1}$. Man schreibe eine rekursive Funktion `fibonacci`, die zu gegebenem Index n das Folgenglied x_n zurückgibt. Man schreibe weiters eine nicht rekursive Funktion `fibonacci2`, die dasselbe leistet, wobei die Berechnung aber über geeignete Schleifen realisiert wird. Welche der beiden Funktionen ist effizienter und warum? Speichern Sie den Source-Code unter `fibonacci.m` in das Verzeichnis `serie08`.

Aufgabe 74*. Gegeben seien eine Matrix $A \in \mathbb{R}^{n \times n}$,

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix},$$

und eine rechte Seite $b \in \mathbb{R}^n$. Man löse das Gleichungssystem $Ax = b$ mit dem *Gauß'schen Eliminationsverfahren*. Dies ist gerade das Vorgehen, wenn man ein lineares Gleichungssystem händisch löst:

- Zunächst bringt man die Matrix A auf obere Dreiecksform, in dem man die Unbekannten eliminiert. Gleichzeitig modifiziert man die rechte Seite b.
- Das entstandene Gleichungssystem mit oberer Dreiecksmatrix A löst man mit Aufgabe 41.

Im ersten Eliminationsschritt zieht man geeignete Vielfache der ersten Zeile von den übrigen Zeilen ab und erhält dadurch eine Matrix der Form

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2} & \dots & a_{nn} \end{pmatrix},$$

Im zweiten Eliminationsschritt zieht man nun geeignete Vielfache der zweiten Zeile von den übrigen Zeilen ab und erhält eine Matrix der Form

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a_{n3} & \dots & a_{nn} \end{pmatrix},$$

Nach $n-1$ Eliminationsschritten erhält man also eine obere Dreiecksmatrix A. Man berücksichtige, dass auch die rechte Seite $b \in \mathbb{R}^n$ geeignet modifiziert werden muss und mache sich das Vorgehen zunächst an einem Beispiel mit $A \in \mathbb{R}^{2\times 2}$ sowie $A \in \mathbb{R}^{3\times 3}$ klar. Man schreibe eine MATLAB-Funktion `gauss`, die die Lösung von $Ax = b$ berechnet. — Sie können die Korrektheit Ihrer Funktion in MATLAB mittels $x - A\backslash b$ überprüfen. Speichern Sie den Source-Code unter `gauss.m` in das Verzeichnis `serie08`.

Aufgabe 75. Das Gauß'sche Eliminationsverfahren scheitert, falls im k-ten Schritt $a_{kk} = 0$ gilt, auch wenn das Gleichungssystgem $Ax = b$ eine eindeutige Lösung x besitzt. Deshalb kann man das Verfahren um eine sogenannte *Pivot-Suche* erweitern:

- Im k-ten Schritt wählt man aus $a_{kk}, \dots, a_{nk}$ das betragsgrößte Element a_{pk}.
- Dann vertauscht man die k-te und die p-te Zeile von A (und b).
- Schließlich führt man den Eliminationsschritt aus wie zuvor.

Man schreibe eine Matlab-Funktion `gausspivot`, die die Lösung von $Ax = b$ wie angegeben berechnet. (Man kann übrigens mathematisch beweisen, dass das Gauss-Verfahren mit Pivot-Suche genau dann durchführbar ist, wenn das Gleichungssystem $Ax = b$ eine eindeutige Lösung besitzt. Einen Beweis dazu sehen Sie in der Vorlesung zur Numerischen Mathematik.) Sie können die Korrektheit Ihrer Funktion in Matlab mittels `x-A\b` überprüfen.

Aufgabe 76. Wenn man im Gauß-Verfahren mit Pivot-Suche die Zeilen im Eliminationsschritt *wirklich* vertauscht (d.h. Speicher kopiert), führt dies auf unnötig viele Operationen und entsprechend lange Laufzeit des Programms. Es empfiehlt sich daher, die Vertauschung nur *virtuell* durchzuführen: Man startet mit einem Buchhaltervektor $\pi = (1, \dots, n)$. Im Vertauschungsschritt vertauscht man lediglich $\pi(p)$ mit $\pi(k)$. Im Source-Code sind jetzt die Zeilenindizes, d.h. der erste Index von a_{jk} sowie der Index von b_j geeignet zu modifizieren.

Aufgabe 77. Man schreibe eine Funktion `merge`, die zwei aufsteigend sortierte Felder a und b so vereinigt, dass das resultierende Feld c ebenfalls aufsteigend sortiert ist, z.B. soll also Aufruf mit $a = (1, 3, 3, 4, 7)$ und $b = (1, 2, 3, 8)$ als Ergebnis $c = (1, 1, 2, 3, 3, 3, 4, 7, 8)$ liefern.

Aufgabe 78. Man schreibe eine rekursive Funktion `mergesort`, die ein Feld a aufsteigend sortiert und das sortierte Feld zurückgibt:

- Hat a Länge ≤ 2, so wird das Feld a explizit sortiert.
- Hat a Länge > 2, halbiere man a in zwei Teilfelder a_1 und a_2, rufe `mergesort` für a_1 und a_2 auf und vereinige die sortierten Teilfelder mittels `merge` aus Aufgabe 77.

Aufgabe 79. Programmieren Sie eine Funktion `computeLU`, die die LU-Zerlegung einer Matrix wie in Aufgabe 57 beschrieben berechnet. Verwenden Sie möglichst wenig Schleifen und benutzen Sie stattdessen wenn möglich Matlab-Arithmetik.

Aufgabe 80. Man schreibe eine Funktion `solveLU`, die die Lösung x des linearen Gleichungssystems $Ax = b$ berechnet. Dabei soll die folgende Lösungsstrategie verwendet werden:

(1) Berechne die LU-Zerlegung von A.

(2) Löse $Ly = b$ nach y.

(3) Löse $Ux = y$ nach x.

Es gilt nämlich $Ax = LUx = Ly = b$.