Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Wintersemester 2009/10
09. Dezember 2009

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 8

Die Aufgaben mit Stern (*) sind bis zur Übung in der kommenden Woche vorzubereiten. Kopieren Sie die Source-Codes vor der Übung auf Ihren Account auf der `lva.student.tuwien.ac.at` *und überprüfen Sie, ob diese von* `matlab` *auf der* `lva.student.tuwien.ac.at` *interpretiert werden können. In den folgenden Aufgaben sollen* **Arithmetik, Verzweigungen** *und* **Zählschleifen** *geübt werden.*

Aufgabe 8.1*. Was macht die folgende Matlab-Funktion?

```
function [y,j] = f(x)
j = 1;
absy = abs(x(1));
for k = 2:length(x)
  absx = abs(x(k));
  if absx > absy
    absy = absx;
    j = k;
  end
end
y = x(j);
```

Welchen Wert haben die Variablen `j`, `k`, `absx`, `absy` und `x` jeweils vor dem `end` in der vorletzten Zeile, wenn man die Funktion mittels

```
[y,j] = f([1,-3,3,2,4,-4])
```

aufruft? Schreiben Sie einen alternativen Source-Code, der anstelle der Schleife geeignete Matlab-Funktionen verwendet.

Aufgabe 8.2*. Gegeben sei eine obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ mit $u_{jj} \neq 0$ für alle $j = 1,\dots,n$. Für gegebenes $b \in \mathbb{R}^n$ existiert dann eine eindeutige Lösung $x \in \mathbb{R}^n$ von $Ux = b$. Schreiben Sie eine Funktion `solveU`, die die Lösung berechnet. Dabei sollen lediglich Schleifen und elementare Matlab-Arithmetik verwendet werden. Versuchen Sie, möglichst viele Schleifen über die Vektor-Arithmetik zu realisieren. Speichern Sie den Source-Code im Verzeichnis `serie08` unter `solveU.m`.

Aufgabe 8.3*. Programmieren Sie die LU-Zerlegung einer Matrix wie in Aufgabe 6.6 beschrieben. Schreiben Sie zunächst eine Funktion `computeLUfor` in der Sie die Implementierung durch Schleifen realisieren. Schreiben Sie eine zweite Lösung `computeLU` in der Sie Schleifen möglichst vermeiden und durch geeignete Matlab-Arithmetik ersetzen. Speichern Sie die Source-Codes im Verzeichnis `serie08`.

Aufgabe 8.4*. Schreiben Sie 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 $Ax = LUx = Ly = b$.

Aufgabe 8.5. Man kann den Speicheraufwand in Aufgabe 8.4 minimieren, indem man die Einträge der Matrix A geeignet durch die Einträge der Matrizen L und U überschreibt. Ferner kann man den Vektor b bei geeignetem Vorgehen, zunächst durch y und schließlich durch x überschreiben. Dadurch wird insgesamt kein zusätzlicher Speicher benötigt. Realisieren Sie dieses Vorgehen.

Aufgabe 8.6. Die Determinante einer Matrix $A \in \mathbb{R}^{n\times n}$ kann über die normalisierte LU-Zerlegung aus Aufgabe 8.3 berechnen. Es gilt nämlich $\det(A) = \det(L)\det(U) = \det(U) = \prod_{j=1}^{n} u_{jj}$. Schreiben Sie eine Funktion `detLU`, die die Determinante einer Matrix A mittels der normalisierten LU-Zerlegung berechnet.

Aufgabe 8.7. Schreiben Sie eine Funktion `rundung`, die für eine gegebene Zahl $x \in \mathbb{R}$ die Zahl $n \in \mathbb{Z}$ zurückliefert, die x am nächsten liegt. Falls x genau in der Mitte zwischen zwei ganzen Zahlen liegt, werde die größere zurückgegeben.

Aufgabe 8.8. Gegeben seien eine Matrix $A \in \mathbb{R}^{n\times n}$ und eine rechte Seite $b \in \mathbb{R}^n$. Lösen Sie 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 8.2.

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. Berücksichtigen Sie, dass auch die rechte Seite $b \in \mathbb{R}^n$ geeignet modifiziert werden muss und machen Sie sich das Vorgehen zunächst an einem Beispiel mit $A \in \mathbb{R}^{2\times 2}$ sowie $A \in \mathbb{R}^{3\times 3}$ klar. Schreiben Sie eine Matlab-Funktion `gauss`, die die Lösung von $Ax = b$ berechnet.

Aufgabe 8.9. Das Gauß'sche Eliminationsverfahren scheitert, falls im k-ten Schritt $a_{kk} = 0$ gilt, auch wenn das Gleichungssystem $Ax = b$ 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.

Schreiben Sie 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.)

Aufgabe 8.10. 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 $\pi = (\mathtt{1}, \dots, \mathtt{n})$. Im Vertauschungsschritt vertauscht man lediglich $\pi(p)$ mit $\pi(k)$. Im Source-Code (sowohl zu Aufgabe 8.2 als auch zu Aufgabe 8.8) sind jetzt die Zeilenindizes, d.h. der erste Index von a_{jk} sowie der Index von b_j geeignet zu modifizieren.