Dirk Praetorius, Samuel Ferraz-Leite
Christoph Überhuber

Sommersemester 2009
21. Mai 2009

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. 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, Verzweigungen** *und* **Zählschleifen** *geübt werden.*

Aufgabe 71*. 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 72*. 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 73*. 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 weder Matlab-Funktionen (z.B. `inv`, `linsolve`) noch der Backslash-Operator (für Matrizen) verwendet werden, sondern lediglich Schleifen und elementare Matlab-Arithmetik. Versuchen Sie, möglichst viele Schleifen über die Vektor-Arithmetik zu realisieren. — Sie können die Korrektheit Ihrer Funktion in Matlab mittels `x-U\b` überprüfen.

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 73.

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\b` überprüfen.

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 (sowohl zu Aufgabe 73 als auch zu Aufgabe 74) sind jetzt die Zeilenindizes, d.h. der erste Index von a_{jk} sowie der Index von b_j geeignet zu modifizieren.

Aufgabe 77. 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 78. Man schreibe eine Funktion `matrixvectorU`, die das Matrix-Vektor-Produkt $y = Ux$ mit einer oberen Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ mittels geeigneter Schleifen berechnet. Dabei soll auf die offensichtlichen Nulleinträge von U nicht zugegriffen werden, um den Rechenaufwand gering zu halten. Sie können Ihre Funktion mittels `U*x` verifizieren. Eine zufällige obere Dreiecksmatrix $U \in \mathbb{R}^{n\times n}$ können Sie beispielsweise mit `[tmp,U] = qr(rand(n));` erzeugen. Was machen diese beiden Befehle?

Aufgabe 79. Das Produkt $U = AB$ zweier oberer Dreiecksmatrizen $A, B \in \mathbb{R}^{n\times n}$ ist wieder eine obere Dreiecksmatrix. Man beweise diese Aussage zunächst mathematisch, indem man sich die Formel für das Matrix-Matrix-Produkt hinschreibe und mittels der Voraussetzung an A und B die Indizes vereinfache. Danach schreibe man eine Funktion `matrixmatrixU`, die die Produktmatrix berechnet und zurückgibt. Dabei sollen natürlich nur die nicht-trivialen Einträge von U, d.h. U_{jk} für $j \le k$, berechnet werden. Ferner soll auf die trivialen Einträge von A und B nicht zugegriffen werden, d.h. man verwende die anfangs hergeleitete Formel. — Sie können Ihre Funktion mittels `A*B` verifizieren.

Aufgabe 80. Man schreibe eine Funktion `matrixvectorL`, die das Matrix-Vektor-Produkt $y = Lx$ mit einer unteren Dreiecksmatrix L berechnet. Dabei soll auf die offensichtlichen Nulleinträge von L nicht zugegriffen werden, um den Rechenaufwand gering zu halten.