Dirk Praetorius
Christoph Überhuber

Wintersemester 2006/07
01. Dezember 2006

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 7

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 `serie07` *Ihres Home-Verzeichnisses. Überprüfen Sie bitte vor der Übung, ob Ihre Source-Codes mit* `matlab` *auf der* `cad.zserv.tuwien.ac.at` *interpretiert werden können. In den folgenden Aufgaben sollen im wesentlichen **Verzweigungen** und **Zählschleifen** geübt werden, um sich mit Matlab vertraut zu machen.*

Aufgabe 57*. Starten Sie den Matlab-Interpreter, indem Sie in einer Shell den Befehl `matlab` eingeben. Informieren Sie sich über den Gebrauch der Verzweigung und der Zählschleife in Matlab, z.B. indem Sie in der Matlab-Shell die Befehle `help if` und `help for` eingeben. Was bedeutet der Befehl

```
vector = 14:-1:1;
```

im Unterschied zu

```
vector = 14:-1:1
```

Wie werden Vektoren und Matrizen in Matlab indiziert?

Aufgabe 58*. 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);
```

Klären Sie allfällige Matlab-Befehle mit Hilfe der `help`-Funktion. 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? Sie können Ihre Überlegungen verifizieren, indem Sie die Funktion `f` in einer Datei `f.m` abspeichern und im selben Verzeichnis Matlab starten.

Aufgabe 59*. Man schreibe eine Matlab-Prozedur `kurvendiskussion`, die die (lokalen) Extrema eines kubischen Polynoms $p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ berechnet und ausgibt, ob es sich um ein Minimum oder ein Maximum handelt. Die Koeffizienten sollen als Vektor $a \in \mathbb{R}^4$ an die Funktion übergeben werden. Zur Ausgabe verwenden Sie den Befehl `fprintf`, der wie `printf` in C verwendet wird. Beachten Sie die Sonderfälle, dass p eine Gerade oder eine Parabel ist. Speichern Sie den Source-Code unter `kurvendiskussion.m` ins Verzeichnis `serie07`.

Aufgabe 60*. Man schreibe eine Matlab-Funktion `bubblesort`, die den Sortier-Algorithmus aus Aufgabe 26 realisiert. Speichern Sie den Source-Code unter `bubblesort.m` ins Verzeichnis `serie07`. Sie können Ihren Code mit Hilfe des Matlab-Befehls `sort` verifizieren.

Aufgabe 61. Man schreibe eine Prozedur `vielfache`, die alle ganzzahligen Vielfachen der Zahl $k \in \mathbb{N}$, die $\leq n_{\max} \in \mathbb{N}$ sind, am Bildschirm ausgibt. Die Ausgabe erfolge zeilenweise in der Form

```
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
```

beispielsweise für den Fall $k = 5$ und $n_{\max} = 19$.

Aufgabe 62. 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.

Aufgabe 63. 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 35.

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 Eliminationsschrfitt 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 64. 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 Nunmerischen Mathematik.) Sie können die Korrektheit Ihrer Funktion in Matlab mittels `x-A\b` überprüfen.

Aufgabe 65. 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 35 als auch zu Aufgabe 63) sind jetzt die Spaltenindizes, d.h. der erste Index von a_{jk} sowie der Index von b_j geeignet zu modifizieren.