Samuel Ferraz-Leite, Dirk Praetorius
Christoph Überhuber

Sommersemester 2008
21. Mai 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` *interpretiert werden können. In den folgenden Aufgaben sollen im wesentlichen* ***Arithmetik*** *und* ***Zählschleifen*** *geübt werden.*

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

**Aufgabe 72*.** 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);
```

Geben Sie in der Matlab-Shell den Befehl `help for` ein, um sich über die Funktionsweise der for-Schleife zu informieren. 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 73*.** Nicht jede Matrix $A \in \mathbb{R}^{n\times n}$ hat eine normalisierte LU-Zerlegung $A = LU$, d.h.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \ell_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \ell_{n1} & \dots & \ell_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \dots & 0 & u_{nn} \end{pmatrix}.$$

Wenn aber $A$ eine normalisierte LU-Zerlegung besitzt, so gilt

$$u_{ik} = a_{ik} - \sum_{j=1}^{i-1} \ell_{ij} u_{jk} \quad \text{für } i = 1, \ldots, n, \quad k = i, \ldots, n,$$

$$\ell_{ki} = \frac{1}{u_{ii}} \left( a_{ki} - \sum_{j=1}^{i-1} \ell_{kj} u_{ji} \right) \quad \text{für } i = 1, \ldots, n, \quad k = i+1, \ldots, n,$$

$$\ell_{ii} = 1 \quad \text{für } i = 1, \ldots, n,$$

wie man leicht über die Formel für die Matrix-Matrix-Multiplikation zeigen kann. Alle übrigen Einträge von $L, U \in \mathbb{R}^{n \times n}$ sind Null. Man schreibe eine Funktion `computeLU`, die die LU-Zerlegung von $A$ berechnet und zurückgibt. Dazu überlege man, in welcher Reihenfolge man die Einträge von $L$ und $U$ berechnen muss, damit die angegebenen Formeln wohldefiniert sind (d.h. alles was benötigt wird, ist bereits zuvor berechnet worden). Verwenden Sie bei der Implementierung lediglich die MATLAB-Arithmetik sowie geeignete Schleifen. Sie können ihren Funktion in MATLAB mittels `lu` verifizieren.

**Aufgabe 74*.** Man schreibe eine Funktion `solveLU`, die die Lösung $x$ des linearen Gleichungssystem $Ax = b$ berechnet. Dabei soll 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 dann nämlich $Ax = LUx = Ly = b$. Verwenden Sie bei der Implementierung lediglich die MATLAB-Arithmetik sowie geeignete Schleifen.

**Aufgabe 75.** Man kann den Speicheraufwand in Aufgabe 74 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 76.** Die Determinante einer Matrix $A \in \mathbb{R}^{n \times n}$ kann über die normalisierte LU-Zerlegung aus Aufgabe 73 berechnet werden. 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. Sie können Ihre Funktion mit der Matlab-Funktion `det` verifizieren. Die Berechnung der LU-Zerlegung können sie in Matlab mittels `lu` überprüfen.

**Aufgabe 77.** 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 mit und ohne Schleifen.

**Aufgabe 78.** Die Fibonacci-Folge ist definiert durch $x_0 := 0, x_1 := 1$ und $x_{n+1} = x_n + x_{n-1}$. Man schreibe eine Matlab-Funktion `x = fibonacci(k)`, die zu gegebenem Index $k$ die Teilfolge $x_0, x_1, \ldots, x_k$ bis zum $k$-ten Folgenglied zurückgibt.

**Aufgabe 79*.** Man schreibe eine Matlab-Funktion `kurvendiskussion`, die die (lokalen) Extrema eines Polynoms dritten Grades $p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ 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. Beachten Sie die Sonderfälle, dass $p$ eine Gerade oder eine Parabel ist.

**Aufgabe 80.** Man schreibe eine Matlab-Funktion `matrixtype(A)`, die ausgibt, ob eine gegebene quadratische Matrix $A \in \mathbb{R}^{n \times n}$ diagonal (d.h. $A_{ij} = 0$ für $i \neq j$), symmetrisch, ein magisches Quadrat und/oder eine allgemeine Matrix ist. Unter einem magischen Quadrat versteht man eine Matrix, deren Zeilen- und Spaltensummen sowie die Summe der Diagonaleinträge stets denselben Wert annehmen. Weiters soll ausgegeben werden, ob die Matrix $A$ regulär ist. Sie können magische Quadrate in Matlab mit dem Befehl `magic` zum Testen Ihrer Funktion erstellen.