Einführung in das Programmierung mit Matlab, Teil 2/4

Thomas Dunst

Mathematisches Institut
Universität Tübingen

(e–mail: progtutor@na.uni-tuebingen.de)

10. Oktober 2012

Wiederholung:

Was haben wir gestern gemacht:

- Allgemeines zu MATLAB
- Anweisungen
  - Operatoren (arithmetische Op. z.B. +,-, relationale Op. z.B. <,>, logische Op. z.B. &)
  - Kontrollanweisungen (Schleifen, Verzweigungen)
  - Ausgabe von Variablen

```
1  % Beispiel das Alles beeinhaltet:
2  summe = 0; % Anfangswert fuer Summation
3  j = 0; % Anfangswert fuer Iterationsvariable
4  while j<10
5    j = j+1;
6    summe = summe + 2 * j + 1;
7  end
8  disp('Ergebnis: ');
9  disp(summe);
```

Wiederholung:

Was haben wir gestern gemacht:

- Allgemeines zu MATLAB
- Anweisungen
  - Operatoren (arithmetische Op. z.B. +,-, relationale Op. z.B. <,>, logische Op. z.B. &)
  - Kontrollanweisungen (Schleifen, Verzweigungen)
  - Ausgabe von Variablen

```
1  % Beispiel das Alles beeinhaltet:
2  summe = 0; % Anfangswert fuer Summation
3  % Hier eignet sich eine FOR Schleife besser!
4  for j=1:1:10
5     summe = summe + 2 * j + 1;
6  end
7  disp('Ergebnis: ');
8  disp(summe);
```

Gliederung

Lösungen für die Aufgaben des 1. Übungsblattes

Vektoren
- Erzeugung
- Zugriffsarten auf Vektoren
- wichtige Operationen auf Vektoren
- Beispiel: Mittelwertberechnung
- Operatoren auf Vektoren

Matrizen
- Erzeugung
- Zugriffsarten auf Matrizen
- wichtige Operationen auf Matrizen
- Beispiel: Matrizenmultiplikation
- Operatoren auf Matrizen

Funktionen und Skripte
- Definition einer Funktion
- Auslagern von Code-Blöcken in Funktion
- Beispiel: Funktion
- wichtige Funktionen in Matlab

Lösungen für die Aufgaben des 1. Übungsblattes:

Aufgabe: 4

```
1  for n = 1 : 10
2    n
3  end
```

Aufgabe: 5

```
1  for n = 0 : 10
2    2 * n + 1
3  end
```

Alternative

```
for n = 1 : 2: 21
  n
end
```

Aufgabe 6: siehe Beispiel auf Folien des 1. Tages.

Aufgabe 7/8:

```
1  n = 10      % Summationsende
2  q = 0.5     % zu entwickelnde Zahl
3  summe = 0 % Initialisierung der Summe
4  for i = 0 : n
5    summe = summe + q^i
6  end
```

Lösungen für die Aufgaben des 1. Übungsblattes:

Aufgabe 9:

```
1  S = 0; % Initialisieren der Summe
2  n = 0; % Initialisieren von n
3  while (abs(S - 2)>=0.01)
4          S = S + (1/2)^n
5          n=n+1;
6  end
7  n-1
```

Ausgabe:

```
S =  1
S =  1.5000
S =  1.7500
S =  1.8750
S =  1.9375
S =  1.9688
S =  1.9844
S =  1.9922
n =  7
```

Variablen I

MATLAB ist ein auf Matrizen basierendes Werkzeug. Alle Daten die in MATLAB eingegeben werden, werden als Matrix oder mehrdimensionales Array abgespeichert.

- In MATLAB sind Skalare und Vektoren nur der Spezialfall von Matrizen.
- Skalare sind (1×1)-Matrizen
- Vektoren der Dimension n sind $(1 \times n)$ oder $(n \times 1)$-Matrizen.
- MATLAB numeriert die Elemente beginnend mit 1 nicht mit 0!
- Mit dem Befehl `who` bzw. `whos` kann man rausfinden, welche Variablennamen schon vergeben wurden.

Vektoren: Erzeugung

- `a = [1, 2, 3, 4] % Zeilenvektor der Dim. 4`
- `b = [1 2 3 4]  % identisch zu a`
- `c = [1; 2]  % Spaltenvektor der Dim. 2`
- `d = 1:0.1:2  % Spaltenvektor der Dim. 11`
- `a = [] % leerer Vektor, d.h. Dim. von a ist 0`

Ausdruck kann zum Löschen einer Variable (hier Vektor a) benutzt werden.

Zugriffsarten auf Vektoren

- als Ganzes:

```
g = a % Vektor g ist nun identisch
      % zu Vektor a
```

- Elementweise:

```
g = a(1) % g ist Skalar und enthaelt
         % erstes Element von a
i = 2;
g = a(i) % g enthaelt i-tes (2.) Element von a
g = a(end) % g enthaelt letztes Element von a
```

  i ist Index. i indiziert Element von a.

- Bereichswahl:

```
g = a(1:3) % g ist Vektor der Dim. 3 und ent-
           % haelt ersten drei Elemente von a
```

wichtige Operationen auf Vektoren

Wichtig: Indizes können nur natürliche Zahlen ohne Null sein!

- Transponieren (’):

```
% c sei ein Spaltenvektor
d = c' % ist ein Zeilenvektor
e = d' % ist ein Spaltenvektor
```

- Länge eines Vektors ermitteln `length(a)`, z.B. für Schleifen

```
n = length(a) % Dim. des Vektors bestimmen
for i=1:n
  a(i)        % i-tes Element von a
end           % anzeigen.
```

Beispiel: Mittelwertberechnung

Berechnung des Mittelwertes $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ für $n = 4$ und die Werte $x_1 = 3, x_2 = 7, x_3 = 5, x_4 = 1$.

Beispiel: Mittelwertberechnung

Berechnung des Mittelwertes $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ für $n = 4$ und die Werte $x_1 = 3, x_2 = 7, x_3 = 5, x_4 = 1$.

```
1  x = [3 7 5 1]; % Vektor (1 x 4) mit Werten
2  % Mittelwert berechnen
3  s = 0                  % Summe initialisieren.
4  for i = 1 : 4
5     s = s + x(i)      % aktuelle Summe ausgeben
6  end
7  mw = s / 4             % Mittelwert
```

Ausgabe:

```
s =  0
s =  3
s =  10
s =  16
mw =  4
```

Operatoren auf Vektoren

Operatoren wirken auf Vektoren anders als auf Skalare. Unter anderem existieren folgende Operatoren auf Vektoren. Die Vektoren müssen dazu die gleiche Dimension haben.

- Zuweisungsoperator: `=`
  `a = b`: $a(i) = b(i)$ für alle i.
- Arithmetische Operatoren: `+` `-`
  `a + b`: $a(i) + b(i)$ für alle i.
- Arithmetische Operatoren: `*`
  hat 2 Bedeutungen:
  1. a, b sind Vektoren:
     siehe Übungen.
  2. a ist Skalar, b ist Vektor
     `a * b`: $a * b(i)$ für alle i.
- relationale Operatoren: `==` `~=`
  `a == b`: $a(i) = b(i)$ für alle i.
  `a ~= b`: $a(i) \neq b(i)$ für mindestens ein i.

(Weitere Operatoren sind definiert, werden aber nicht besprochen.)

Matrizen: Erzeugung

- `A = [1 2 3 4; 5 6 7 8] % 2 x 4 - Matrix`
- `B = [1 5; 2 6; 3 7; 4 8]' % identisch zu A`

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

- `B(2,:) = [3 6 9 12] % untere Zeile von B neu.`

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 6 & 9 & 12 \end{pmatrix}$$

- `A = [] % a leere Matrix, d.h. Dim. von A ist 0`

Ausdruck kann zum Löschen einer Matrix benutzt werden.

Matrizen: Erzeugung II

- `I = eye(n) % n x n - Einheitsmatrix`
- `A = zeros(n,m) % n x m - Matrix mit Nullen`
- `B = ones(n,m) % n x m - Matrix mit Einsen`
- ```
  v = [1 2 3 4] % Definition eines Vektors\\
  D = diag(v)   % macht aus Vektor v\\
                % eine Diagonalmatrix
  ```

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$
  ```

Zugriffsarten auf Matrizen

- als Ganzes:

```
G = A % Matrix  G ist nun identisch
      % zu Matrix A
```

- Elementweise:

```
g = A(3,1) % g ist Skalar und enthaelt Element
      % aus dritter Reihen erster Spalte von A
i = 2, j = 2;
g = A(i,j) % g enthaelt 2. Diagonal-Element
           % von A
g = A(end,1) % g enthaelt erstes Element
             % letzter Reihe von A
```

- Bereichswahl:

```
G = A(2:4,2:4) % G ist 3 x 3 Matrix und ent-
               % haelt Untermatrix von A
v = A(:,2) % v ist 2. Spaltenvektor von A.
```

wichtige Operationen auf Matrizen

- Transponieren (’):

```
D = C’ % D ist transponierte von C
E = D’ % E = C
```

- Größe einer Matrix ermitteln: `size(A)`

```
size(A)
[n, m] = size(A) % n Spalten, m Zeilen
```

liefert die Ausgabe:

```
ans = 2 4
n = 2
m = 4
```

Beispiel Matrizenmultiplikation

Für eine gegebene $(m \times n)$-Matrix A und eine $(n \times r)$-Matrix B ist die Matrix-Multiplikation folgendermaßen definiert:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \qquad \forall i = 1, \ldots, m \quad j = 1, \ldots, r$$

Beispiel Matrizenmultiplikation

Für eine gegebene $(m \times n)$-Matrix A und eine $(n \times r)$-Matrix B ist die Matrix-Multiplikation folgendermaßen definiert:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

```
 1  % A sei gegeben
 2  % B sei gegeben
 3  for ii = 1 : m     % ueber alle Zeilen von c
 4    for jj = 1 : r   % ueber alle Spalten von c
 5      c(ii,jj) = 0;    % Summe erstmal Null.
 6      for k = 1 : n
 7  % Aufsummieren der Eintraege der
 8  % Spalten A und Zeilen B.
 9        c(ii,jj) = c(ii,jj) + A(ii,k) * B(k, jj)
10      end
11    end
12  end
```

Operatoren auf Matrizen I

Es gelten die üblichen Rechenregeln für Matrix-Matrix-, Matrix-Vektor- und Skalar-Matrix-Operationen:

- Zuweisungsoperator: `=`
  `A = B`: $A(i,j) = B(i,j)$ für alle i, j.
- Arithmetische Operatoren: `+` `-`
  `A + B`: $A(i,j) + B(i,j)$ für alle i, j.
- Arithmetischer Operator: `*`
  hat viele Bedeutungen, die den gewohnten Rechenregeln entsprechen:
  1. A, B sind Matrizen:
     Matrixmultiplikation. Siehe Beispiel.
  2. a ist Skalar, B ist Matrix
     `a * B`: $a * B(i,j)$ für alle i, j.
  3. A ist $(m \times n)$- Matrix, b ist $(n \times 1)$ - Vektor
     Matrix-Vektor-Multiplikation.

Operatoren auf Matrizen II

- relationale Operatoren: `==` `~=`
  `A == B`: $A(i,j) = B(i,j)$ für alle i, j.
  `A ~= B`: $A(i,j) \neq B(i,j)$ für mindestens ein i, j.

(Weitere Operatoren sind definiert, werden aber nicht besprochen.)

Variablen II

- Löschen von Variablen: (sinnvoll wenn Variable andere Dimension erhalten soll)

```
clear A; % Loescht Variable A
clear all; % Loescht alle Variablen
```

- Die Variable `ans` nimmt Ergebnis einer Berechnung auf, wenn keine Ziel-Variable angegeben wurde.

```
2 * 3
ans = 6
2 * 3; % keine Ausgabe wegen ";"
A = 2 * 3 % Ergebnis wurde Variable A zugewiesen
A = 6
```

Funktionen und Skripte

MATLAB unterscheidet zwischen Skripten und Funktionen:

- Skripte sind Textdateien, die Anweisungen enthalten, die man genauso gut im Command Window einzeln eintippen könnte.
- Funktionen sind auch Textdateien mit folgenden Unterschieden zu Skriptdateien:
  - Funktionendateien enthalten spezielle Anweisungen: In der ersten Zeile steht als erstes das Schlüsselwort `function` und weiter hinten in der Zeile der zum Dateinamen gleichlautende Funktionenname.
  - Funktionen können so angelegt werden, dass sie beim Aufruf ein oder mehrere Funktionsargumente erwarten, z.B. `f(x)`
  - Funktionen können so angelegt werden, dass sie nach ihrer Beendigung Rückgabewerte an den Aufrufer liefern.
  - Es besteht die Möglichkeit, über spezielle Kommentare eine Hilfe für die Funktion in Matlab zu importieren.

Definition einer Funktion

Eine eigene Funktion wird in MATLAB in folgender Weise definiert.
Bsp:

```
 1  function [X, Y] = funktionsname(A,B,C)
 2  % Ein paar Worte was die Funktion macht
 3  %
 4  % Input:
 5  % Matrix A: Bedeutung
 6  % Matrix B: Bedeutung
 7  % Matrix C: Bedeutung
 8  % Output:
 9  % Matrix X: Bedeutung
10  % Matrix Y: Bedeutung
11
12      :
13      :
14  end
```

Definition einer Funktion II

- Der Dateiname der im obigen Bsp. definierten Funktion namens `funktionsname` lautet `funktionsname.m`
- die Funktion liefert als Rückgabewert in einem Feld 2 Variablen passenden Typs zurück: `X, Y`.
- Die Funktion erwartet beim Aufruf 3 Variablen passenden Typs: `A, B, C`.
- Die zurückgegebenen Variablen enthalten die Werte, die diese Variablen am Ende der Ausführung der Funktion hatten. (D.h. die Rückgabewerte müssen nicht speziell irgendwo gesetzt werden oder speziell zurückgegeben werden.)
- Erwartet die Funktion keine Argumente, bleibt die runde Klammer hiner dem Funktionsnamen leer `()`.
- Gibt die Funktion keine Argumente zurück bleibt die eckige Klammer hinter `function` leer `[]`.

Auslagern von Code-Blöcken in eine Funktion

Das Vorgehen:

- Anlegen einer Daten mit dem Namen der zu erzeugenden Funktion (plus `.m` als Endung).
- Erzeugen eines Funktionsgerüsts wie im Beispiel.
- Kopieren des Code-Blocks in die erzeugte Datei.
- Anpassen des Codes, so dass er evtl. übergebene Argumente verarbeitet.
- Anpassen des Codes, so dass er evtl. geforderte Rückgabewerte zurückgibt.
- Testen...
- Achtung: Die übergebenen Datentypen müssen zusammenpassen. Erwartet die Funktion z.B. einen Skalar, erhält aber eine Matrix, beschwert sich Matlab erst, wenn etwa eine Matrixmultiplikation wegen falscher Dimensionen der Beteiligten fehlschlägt.

Beispiel Funktion

Beispiel-Funktion in Datei bsp02.m

```
 1  function [a, b] = bsp02(X, Y)
 2  % Hilfetext...
 3  %
 4  % Input:
 5  % Matrix X, Y   ...
 6  % Output:
 7  % Matrix a, b   ...
 8
 9  a =   X * Y   % Rueckgabevariable a belegen
10  b =   X + Y   % Rueckgabevariable b belegen
11  a = []        % a doch leeren.
12
13  % b enthaelt X + Y
14  % a ist leerer Vektor
15  end
```

Beispiel-Skript

```
1  g1 = 3;
2  [c,d] = bsp02(g1, 2);
3  c  % sollte leer sein
4  d  % sollte 5 ( = 2 + 3) sein.
```

wichtige Funktionen in Matlab

Funktionen, die man in Matlab oft benutzt.

- Trigonometrische Funktionen `sin, cos, tan`
- andere mathematische Funktionen `exp, log, abs, sqrt`
  (Exponentialfunktion, Logarithmusf., Absolutbetragsf., Wurzelf.)
- Verwaltungsfunktionen `length, size`
  Größe von Daten/Variablen bestimmen.
- Ein/Ausgabe-Funktionen `disp, load, save`
  einfache Bildschirmausgabe, laden / speichern von Variablen.
- andere Ein/Ausgabe-Funktionen `fprintf, sprintf`
  mächtigere Ausgabe auf Bildschirm oder in Datei, Zeichenketten erzeugen. (wird nicht besprochen)
- grafische Funktionen `plot`
  Funktionen plotten. (3. Teil der Veranstaltung)
- mächtige Funktionen `lu, qr, norm, max`
  Marix-Zerlegungen, Norm ber., max. (wird nicht besprochen)