

Learning MATLAB by doing MATLAB

Christian Mehl*

Andreas Steinbrecher†

31. Oktober 2001

Dieses kleine MATLAB-Tutorial setzt auf "Learning by Doing". Gib jeweils die hinter > angegebenen Befehle im MATLAB Command Window ein und beobachte, was bei der Ausgabe passiert.

1. Variablen, Vektoren, Matrizen

```
>a=7          a wird als Skalar interpretiert (oder 1 x 1-Matrix)
>b=[1,2,3]    nach Komma: neues Element in derselben Zeile, also hier  $b \in \mathbb{R}^{1,3}$ 
>c=[1+2,3,3]
>d=[7 7 2]    Leerzeichen haben gleiche Bedeutung wie Kommata
>e=[7 a 2]
>f=[1;2;3;4]  Semikolon: Beginn einer neuen Zeile, also hier  $f \in \mathbb{R}^{4,1}$ 
>g=f(2)       fragt das 2. Element des Spaltenvektors  $f$  ab
>E=[1 2 3;2 1 3] MATLAB unterscheidet zwischen Groß- und Kleinschreibung
>h=E(1,2)     fragt das (1,2)-Element von  $E$  an
>E
>F(3,4)=7     MATLAB interpretiert  $F$  zunächst als  $3 \times 4$ -Matrix.
              Nicht benannte Elemente werden auf Null gesetzt.
>F(4,3)=2     Jetzt brauchen wir eine 4. Zeile!
>F(1,2)=3     Belegt das (1,2)-Element von  $F$  mit 3.
>F(1:2,3:4)=[1 3;2 7] 1 : 2 bedeutet Zeile 1 bis Zeile 2; 3 : 4 bedeutet Spalte 3 bis Spalte 4
>who          gibt aus, welche Variablen belegt sind
>clear a b    löscht die Variablen  $a$  und  $b$ 
>who
>clear       löscht alle Variablen
>help clear  gibt Hilfe zum Befehl clear
>A=[1 2];    Semikolon am Ende unterdrückt die Ausgabe
>A
>pi
>A=eye(3)      $3 \times 3$ -Einheitsmatrix
>b=[1 2 3]
>B=diag(b)    Diagonalmatrix
>C=diag([1 7 8])
>D=diag([1;7;8]) geht auch mit Spaltenvektoren
>E=ones(4)     $4 \times 4$ -Matrix mit Einsen
>F=ones(2,3)   $2 \times 3$ -Matrix mit Einsen
>G=zeros(4)
>H=zeros(2,3)
>I=[A B;zeros(3) A] Blockmatrix
>C
>C'          Transponierte von  $C$ 
>w(3)=5     MATLAB interpretiert  $w$  als Zeilenvektor. Das 3. Element wird gleich 5 gesetzt.
>x=0:1/3:2   Zeilenvektor mit Einträgen von 0 bis 2 in 1/3-Schritten
>for i=1:10   $i$  läuft von 1 bis 10. (MATLAB macht erst etwas, wenn end kommt!)
y(i)=2*i     Der Index  $i = 0$  ist hier nicht möglich, da Vektoren kein '0.Element' haben!
end
>for i=1:10
z(i)=2*i;
end
>z
```

*Fachbereich Mathematik, Technische Universität Berlin, D-10635 Berlin, Germany, (mehl@math.tu-berlin.de)

†Fachbereich Mathematik, Technische Universität Berlin, D-10635 Berlin, Germany, (steinbrecher@math.tu-berlin.de)

2. Einfache Operationen

```
>clear
>A=[1 2 3;2 1 0]
>B=[2 2;1 0;0 1]
>C=[0 1 0;5 1 3]
>size(A)           Gibt Anzahl der Zeilen und Spalten von A als Zeilenvektor aus.
>[m,n]=size(A)    So bekomme ich sie einzeln.
>b=[2 1 3]
>x=[2;1;3]
>A*B              Matrixmultiplikation
>A*C              Fehler, da Dimensionen nicht passen!
>A
>C
>A*C'              $A \cdot C^T$ 
>diag(A*C')       liefert die Diagonale der Matrix
>who              ans (für "answer") ist die unbenannte Ausgabevariable
>D=A+C           Matrixaddition
>E=A+B           Fehler, da Dimensionen nicht passen!
>E=A-B'           $E = A - B^T$ 
>g=A*x           Matrix mal Vektor
>g=A*b           Fehler!
>A
>b
>B
>f=b*B           Zeilenvektor mal Matrix
>C=[1 2 3]'
```

3. Matrixmanipulationen

```
>clear
>A=[1 2;3 4]
>A(3,2)=7         Hinzunahme einer dritten Zeile!
>A(1:2,2)        (1:2,2): 1. bis 2. Element der 2. Spalte
>A(3,1:2)        (3,1:2): 1. bis 2. Element der 3. Zeile
>B(3:4,3)=[5;6]  MATLAB erzeugt eine Matrix B bei der das 3. und 4. Element
                  der 3. Spalte 5 bzw 6 ist. Alle anderen Einträge sind Null.
>C(4:5,4)=A(1:2,2)
>B(:,3)          3. Spalte von B
>d=C(1,:)        1. Zeile von C
>E=[1 2 3;4 5 6;7 8 9;10 11 12]
>E(1:2:4,3)      (1:2:4,3): Sucht in der 3. Spalte vom 1. bis 4. Element
                  jedes 2. Element heraus.
>F=[1 2 3 4 5;6 7 8 9 10;11 12 13 14 15]
>G=F(1:2:3,1:2:5) Sucht in der 1., 3. und 5. Spalte jeweils vom
                  1. bis 3. Element jedes 2. Element heraus.
>H=[1 3;9 11]
>H^(-1)          Die Inverse.
>inv(H)          Ebenfalls die Inverse.
>det(H)          Die Determinante.
>b=[99 100 101]
>F(1,1:3)=b
>A
>A(1,1:3)=b      Das (2,3)- und das (3,3)-Element werden neu hinzugefügt.
```

4. Unterprogramme, m-Files

Für die folgenden Unterprogramme musst du Dateien mit dem jeweiligen Namen `file.m` anlegen (z.B. mit deinem Lieblingseditor) und abspeichern. Doch Achtung: Diese m-files müssen in dem dem Verzeichnis liegen, von dem aus MATLAB gestartet wurde.

```
% Das ist mein erstes m-File.           % Hinter % folgt Kommentar.
% Es heißt test1.m und berechnet die Summe aller Elemente einer Matrix A.
function summe=test1(A)
[m,n]=size(A);
summe=0;                                % Initialisierung
for i=1:m                                % i läuft von 1 bis m
    for j=1:n                              % j läuft von 1 bis n
        summe=summe+A(i,j);
    end
end
```

Bitte unter test1.m abspeichern.

```
>clear
>help test1
>A=[1 2;3 4]
>s=test1(A)
```

```
% Das Programm test2.m berechnet die Summe und das Produkt aller Elemente einer Matrix A,
% sowie die Spur (d.h. die Summe aller Diagonalelemente) wenn die Matrix quadratisch ist.
function [summe,prod,spur]=test2(A)
[m,n]=size(A);
summe=0;
prod=1;
spur=0;
for i=1:m
    if (m==n)                               % wenn m = n, dann ...
        spur=spur+A(i,i);
    end
    for j=1:n
        summe=summe+A(i,j);
        prod=prod*A(i,j);
    end
end
end
```

Bitte unter test2.m abspeichern.

```
>clear, help test2           Man kann auch mehrere Befehle in eine Zeile schreiben, durch Kommata getrennt.
>A=[1 2;3 4]
>test2(A)                    Nur das erste Output-Argument wird ausgegeben.
>[su,pr,sp]=test2(A);
>su,pr,sp
>B=[1 2 3;4 5 6]
>[su,pr,sp]=test2(B)
```

Unterprogramme können auch mehrere Variablen zur Eingabe (z.B. `function ausgabe=test3(A,B,C)`) oder weder Eingabe noch Ausgabe erfordern (z.B. `function []=test4()`, Aufruf mit `test4`). Neben der 'for'-Schleife gibt es natürlich noch eine 'while'-Schleife. Nutze `help while` für Informationen.

5. Grafiken

```
>for i=1:10, x(i)=i/10; y(i)=x(i)^2; z(i)=sqrt(x(i)); end
>plot(x,y)
>plot(x,z)
>clf
>plot(x,y)
>hold on
>plot(x,z)
>plot(x,2*z,'r')
>plot(x,y+z,'g*')
>hold off
>plot(x,y-z,'k+')
>help plot
>title('Meine Grafik')
>xlabel('x-Achse')
>ylabel('y-Achse')
>axis([0,20,-5,50])
>box
>grid
>clf
>subplot(3,2,1)
>plot(x,y)
>subplot(3,2,2)
>plot(x,z,'k')
>subplot(3,2,5)
>plot(x,z+y,'mo')
>hold on
>plot(x,z,'k')
>subplot(3,2,1)
>plot(x,z,'k')
>subplot(3,2,4)
>title('leer')
>subplot(3,1,2)
>plot(y)
>orient tall
>help orient
>print -dps test1.ps
>help print
>!ls
>!ghostview test1.ps
>demo
```

Der Plot hat $3 \cdot 2 = 6$ Subplots. Der 1. Subplot wird angesprochen.

Jetzt gibt es nur noch drei Subplots, einer pro Zeile. Der 2. wird angesprochen.

Erzeugt ein ps-file test1.ps dieses Plots.

MATLAB gibt alles was hinter einem '!' steht an Unix weiter.

'ls' ist ein Unix-Befehl, der alle Dateien des Verzeichnisses auflistet.