ulm university universität ulm

WiMa-Praktikum (Matlab 1/9)

Einführung in LaTeX und Matlab

Prof. Dr. Stefan Funken, Dipl.-Ing. Christoph Erath
11. Mai 2009

Warum Matlab?

Was ist Matlab?

Matlab

- ist ein Softwarepaket für numerische Berechnungen und zur Visualisierung;
- wurde in den 1970er Jahren zur Unterstützung von Kursen der Linearen Algebra und numerischen Analysis entwickelt.

Was kann Matlab?

Matlab bietet

- eine einfache Syntax basierend auf dem Matrix-Datentyp;
- ein breites Spektrum mathematischer Funktionen und Algorithmen aus verschiedenen Anwendungsbereichen;
- eine plattformübergreifende Programmiersprache;
- einfach zu bedienende Visualisierungsmöglichkeiten.

Start von Matlab im KIZ Linux Pool

Nachdem eine Shell geöffnet wurde

```
$ option matlab
$ matlab &
```

Alternativ kann auch der konsolen Modus von Matlab benutzt werden

```
$ option matlab
$ matlab -nodisplay

                          < M A T L A B (R) >
                Copyright 1984-2008 The MathWorks, Inc.
                       Version 7.6.0.324 (R2008a)
                           February 10, 2008


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>>
```

Grundrechenarten in Matlab

Matlab kann wie ein Taschenrechner zum Addieren, Subtrahieren, Multiplizieren, Dividieren und Potenzieren verwendet werden.
Achtung! \ liefert die Linksinverse, / die Rechtsinverse, „Was unten steht, dadurch wird geteilt!“

```
>> 1+2
ans =
     3
>> 9*3
ans =
     21
>> 8/2
ans =
     4
>> 8\2
ans =
    0.2500
>> 2^3
ans =
     8
```

- Die Grundrechenarten sind durch die Zeichen $+, -, *, /, \backslash, \hat{}$ (potenzieren) definiert.

Grundrechenarten in Matlab

Auswertungsreihenfolge:

```
>> 2+3-1
ans =
     4
>> 2*4+1
ans =
     9
>> 1+3*2^2
ans =
    13
>> 1+(3*2)^2
ans =
    37
```

- Bei den Operatoren gilt die übliche Auswertungsreihenfolge:
  Potenzieren vor Punktrechnung vor Strichrechnung.
  Auswertungsreihenfolgen können durch Klammerung geändert werden.

Aber Achtung!!!

Matlab rechnet mit Maschinenzahlen, d.h. endliche Genauigkeit!
Es gilt $4 \cdot 13860^4 - 19601^4 + 2 \cdot 19601^2 = 1$ (überschlagen Sie die letzte Stelle), aber Matlab liefert

```
>> 4*13860^4-19601^4+2*19601^2
ans =
     2
>> x = 13860; y=19601;
>> u = 4 * x * x * x * x + 1;
>> v = y * y -1;
>> tmp = u - v * v
tmp =
     0
```

... und auch dies ist nicht richtig, oder?

```
>> floor(114)  % rundet in Richtung -oo
ans =
     114
>> floor(1.14*100)
ans =
     113
```

Elementare Funktionen

Es gibt eine Vielzahl elementarer Funktionen in Matlab:

`exp`	Exponentialfunktion zur Basis e
`log`	Logarithmus Funktionen
`sqrt`	Wurzelfunktionen
`sin, cos, tan`	Trigonometrische Funktionen
`asin, acos, atan`	Inverse der trigonometrischen Funktionen
`abs, sign`	Betragsfunktion bei skalarem Argument bzw. Signum (Vorzeichen)
`round, floor, ceil`	runden, abrunden, aufrunden
`mod, rem`	Modul, Divisionsrest

```
>> sin(pi)
ans =
   1.2246e-16
>> cos(pi)
ans =
    -1
...
```

```
...
>> exp(1)
ans =
    2.7183
>> sqrt(-1)
ans =
        0 + 1.0000i
```

Hilfeseite: `>> help elfun`.

Konstanten in Matlab

In Matlab sind einige spezielle Zahlen definiert:

`inf, -inf`	$\pm\infty$
`NaN`	Not a number, nicht definierter Ausdruck, z.B. `0/0`
`pi`	Kreiszahl π
`eps`	relative Genauigkeit von Gleitpunktzahlen
`i, j`	imaginäre Einheit

```
>> pi*sqrt(-1)
ans =
          0 + 3.1416i
>> 0/0
ans =
   NaN
>> 1/0
ans =
   Inf
>> 1+eps
ans =
    1.0000
```

Variablen

- In Matlab werden Variablen durch Zuweisungen ohne vorherige Deklaration angelegt.
- Variablennamen können aus Buchstaben, Ziffern und dem Zeichen _ bestehen, das erste muss ein Buchstabe sein.
- Matlab unterscheidet zwischen Groß- und Kleinschreibung bei Variablennamen (case-sensitive).

- In einem Workspace definierte Variablen können mit den Funktionen `who` und `whos` angezeigt werden.
- Durch Variablendefinition können vorhandene Matlab Funktionen und Variablen überschrieben werden.
- Mit `clear <Variablenname>` bzw. `clear` kann eine Variable bzw. alle Variablen im Workspace gelöscht werden.
- Vorsicht mit den Variablen `i` und `j`:

```
>> i=2
i =
     2
>> pi*i
ans =
    6.2832
>> clear i
>> pi*i
ans =
        0 + 3.1416i
```

Elementares Rechnen in Matlab

Beispiel: Berechne zu einem Kreisradius r die Fläche und den Umfang des Kreises und den Umfang eines flächengleichen Quadrates.

- Fläche eines Kreises: A_Kreis$= r^2\pi$
- Umfang eines Kreises: U_Kreis$= 2r\pi$
- Umfang eines flächengleichen Quadrates: U_Quadrat$= 4l = 4 \cdot \sqrt{\text{A_Kreis}}$

```
>> r=3
r =
     3
>> A_Kreis=r^2*pi
A_Kreis =
   28.2743
>> U_Kreis=2*r*pi
U_Kreis =
   18.8496
>> U_Quadrat=4*sqrt(A_kreis)
U_Quadrat =
   21.2694
```

- Variablen werden durch Zuweisungen eines Wertes mit „=" definiert.
- Namen müssen mit einem Buchstaben anfangen und dürfen Buchstaben, Zahlen und den Unterstrich enthalten. Dabei wird Groß- und Kleinschreibung berücksichtigt.

Komplexe Zahlen

- Komplexe und reellwertige Zahlen können in Matlab gleichzeitg ohne besondere Deklaration verwendet werden.
- Real- und Imaginärteil einer Zahl können mit den Funktionen `real` bzw. `imag` bestimmt werden.
- Der Betrag einer komplexen Zahl kann mit `abs` bestimmt werden.
- Die Funktion `angle` bestimmt das Argument einer komplexen Zahl.

```
>> z=2*exp(i*pi/3)
z =
   1.0000 + 1.7321i
>> x=real(z)
x =
    1.0000
>> y=imag(z)
y =
    1.7321
...
```

```
...
>> r=abs(z)
r =
     2
>> phi=angle(z)
phi =
    1.0472
>> phi/(2*pi)*360
ans =
    60
```

Rechnen mit Matrizen und Vektoren - Beispiel

```
>> A=[1 2 3
      4 5 6]
A =
     1     2     3
     4     5     6
>> B=[1,2;3,4;5,6]
B =
     1     2
     3     4
     5     6
>> x=[1 2 3]
x =
     1     2     3
>> y=[1;2;3]
y =
     1
     2
     3
```

- Matrizen und Vektoren können in Matlab duch Angabe der Elemente in eckigen Klammern definiert werden.
- Dabei werden die Werte zeilenweise angegeben, Elemente einer Zeile werden durch Komma oder Leerzeichen voneinander getrennt, verschiedene Zeilen werden durch Semikolon oder Zeilenumbruch getrennt.
- Vektoren werden als Matrizen definiert, wobei die Zeilen- oder Spaltendimension 1 ist.

Matrixindizierung

Auf Komponenten von Matrizen/Vektoren kann mit dem ( ) Operator zugegriffen werden. Dazu können die Elemente auf zwei verschiedene Arten indiziert werden:

über Zeilen- und Spaltenindizes

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & & a_{2,n} \\ \vdots & & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

über Indizes der Elemente

$$A = \begin{pmatrix} a_{1} & a_{m+1} & \cdots & a_{(n-1)m+1} \\ a_{2} & a_{m+2} & & a_{(n-1)m+2} \\ \vdots & & \ddots & \vdots \\ a_{m} & a_{2m} & \cdots & a_{mn} \end{pmatrix}$$

Achtung: Bei nur einem Index werden die Elemente, im Gegensatz zur Eingabe, spaltenweise nummeriert.

```
 >> A=[1 2 3; 4 5 6]
A =
     1     2     3
     4     5     6
>> A(2, 2)
ans =
     5
>> A(2)
ans =
     4
```

Spezielle Vektoren

Ebenso gibt es Funktionen für häufig verwendete Vektortypen:

- Sequenz von Zahlen beginnend mit a bis maximal b mit festem Inkrement c: `a:b:c` bzw. `a:c` für das Inkrement 1.
- lineare Unterteilung eines Intervalls $[a; b]$ in $n - 1$ Teilintervalle: `linspace(a,b,n)`
- logarithmische Unterteilung eines Intervalls $[10^a; 10^b]$ in $n - 1$ Teilintervalle: `logspace(a,b,n)`

```
>> 1.0:.5:3.0
ans =
    1.0000    1.5000
2.0000    2.5000    3.0000
>> 1:-.5:-1.0
ans =
    1.0000    0.5000
0   -0.5000   -1.0000
>> 1:4
ans =
    1    2    3    4
>> linspace(2, 6, 5)
ans =
    2    3    4    5    6
>> logspace(1, 4, 4)
ans =
    10    100    1000    10000
```

Matrixindizierung

Mit dem Klammeroperator ( ) kann ...

```
>> A=[1 2 3; 4 5 6]
A =
     1     2     3
     4     5     6
```

... auf einzelne Elemente ...

- Element 2 in 2.Zeile.
- Element 2 der Matrix.

```
>> A(2, 2)
ans =
     5
>> A(2)
ans =
     4
```

... oder auf Teilmatrizen zugegriffen werden.

- Elemente 1, 2 und 4 der Matrix.
- Elemente 2 und 3 aus 1.Zeile.

```
>> A([1,2,4])
ans =
     1     4     5
>> A(1, [2,3])
ans =
     2     3
```

Matrixindizierung

Wichtig ist hier : und end ...

```
>> A=[1 2 3; 4 5 6]
A =
     1     2     3
     4     5     6
```

... auf Teile kann zugegriffen werden ...

- Elemente 1 bis 3 in 1.Zeile.

```
>> A(1, [1,2,3])
ans =
     1   2   3
>> A(1, 1:3)
ans =
     1   2   3
>> A(1, 1:size(A,2))
ans =
     1   2   3
>> A(1, 1:end)
ans =
     1   2   3
>> A(1, :)
ans =
     1   2   3
```

Verändern und Zusammensetzen von Matrizen

Über den Zugriff auf Komponenten einer Matrix kann diese ausgelesen und verändert werden:

```
>> A(1, 2:3)=zeros(1,2)
A =
     1     0     0
     4     5     6
```

Matrizen können aus Teilmatrizen passender Größe zusammengesetzt werden. Nützlich ist auch die Funktion `blkdiag`, die Matrizen entlang der Diagonalen anordnet.

```
>> A=[1:4; eye(1,2), zeros(1,2)]
A =
     1     2     3     4
     1     0     0     0
>> blkdiag([1 2; 3 4],eye(2))
ans =
     1     2     0     0
     3     4     0     0
     0     0     1     0
     0     0     0     1
```

Mit `reshape` können die Dimensionen einer Matrix verändert werden:

```
>> B=reshape(1:4, 2, 2)
B =
     1     3
     2     4
```

Verändern und Zusammensetzen von Matrizen

Mit dem Operator [] können Zeilen oder Spalten von Matrizen und Vektoren gelöscht werden.

- Lösche alle Zeilen aus Spalte 2 bis 3 von

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

```
>> A(:,2:3)=[]
A =
     1     4
     1     0
```

Um eine Matrix zu transponieren bzw. die komplex konjugierte zu bestimmen gibt es die Operatoren `.'` und `'`:

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

```
>> B'
ans =
     1     2
     3     4
```

Mit den Funktionen `fliplr` bzw. `flipud` kann die Reihenfolge der Spalten bzw. Zeilen der Matrix vertauscht werden:

- `fliplr` behält die Zeilen bei und vertauscht die Spalten in rechts/links Richtung.
- `flipud` behält die Spalten bei und vertauscht die Zeilen nach oben/unten.

```
>> fliplr(B)
ans =
     3     1
     4     2
>> flipud(B)
ans =
     2     4
     1     3
```

Matrix-Abmessungen

Die Abmessungen einer Matrix kann mit der Funktion `[z,s]=size(A)` ermittelt werden, dabei gibt z die Anzahl Zeilen und s die Anzahl Spalten an.

```
>> A=[1 2 3; 4 5 6];
>> [z, s]=size(A)
z =
     2
s =
     3
```

Mit `length(A)` kann man die Länge eines Vektors ermitteln.

```
>> x=1:5;
>> length(x)
ans =
     5
```

Die Funktion `numel(A)` gibt die Anzahl der Elemente von `A` zurück, mit `isempty(A)` kann man ermitteln, ob die Matrix leer ist. (Der Rückgabewerte der Funktion `isempty(A)` ist ein `logical` mit den Werten 1 für *true* und 0 für *false*)

```
>> numel(A)
ans =
     6
>> isempty(A)
ans =
     0
>> isempty([])
ans =
     1
```

Teilmatrizen

Weitere Möglichkeiten auf Teile von Matrizen zuzugreifen sind z. B.:

Zugriff auf die Diagonalelemente mit `diag`. Die Funktion `diag` angewand auf einen Vektor erzeugt eine Diagonalmatrix:

$$A = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix},$$

$$\mathrm{diag}([8;5;3]) = \begin{pmatrix} 8 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

```
>> A=magic(3);
>> diag(A)
ans =
     8
     5
     2
>> diag(diag(A))
ans =
     8     0     0
     0     5     0
     0     0     2
```

Die Funktionen `triu` und `tril` liefern die obere bzw. untere Dreiecksmatrix:

```
>> triu(A)
ans =
     8     1     6
     0     5     7
     0     0     2
>> tril(A)
ans =
     8     0     0
     3     5     0
     4     9     2
```

Operatoren für Matrizen

- In Matlab sind Operatoren zum Rechnen mit Matrizen, Vektoren und Skalaren definiert.
- Operationen zwischen zwei Matrizen / Vektoren:
  +, −, ∗ zum Addieren, Subtrahieren, Multiplizieren.

```
>> A=[0,1;4,9]
A =
     0     1
     4     9
>> B=[1,1;1,1]
B =
     1     1
     1     1
>> C=A+B
C =
     1     2
     5    10
>> D=A-B
D =
    -1     0
     3     8
>> E=A*B
E =
     1     1
    13    13
```

```
>> x=[1 2 3]
x =
     1     2     3
>> y=[1 1 1]
y =
     1     1     1
>> a=x+y
a =
     2     3     4
>> b=x-y
b =
     0     1     2
>> c=x*y'
c =
     6
```

Operatoren für Matrizen und Skalare

- Skalare Multiplikation und Division mit den Operatoren * und /:

```
>> 3*[1 1; 0 1]
ans =
     3     3
     0     3
>> [1 1; 0 1]/3
ans =
    0.3333    0.3333
         0    0.3333
```

Multiplikation/Division zwischen Matrizen/-Vektoren und Skalaren wird elementweise durchgeführt.

- Potenzieren mit ^:

```
>> [1 1; 0 1]^2
ans =
     1     2
     0     1
```

Matrix wird quadriert:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$

- Addition und Subtraktion mit + und −:

```
>> [1 1; 0 1]+3
ans =
     4     4
     3     4
```

Additon/Subtraktion mit Skalaren wird elementweise durchgeführt

Komponentenweise Operationen für Matrizen/Vektoren

- Komponentenweise Multiplikation und Division: skalare Multiplikation mit den Operatoren `.*` und `./`:

```
>> A=[1 2; 3 4];
>> B=[1 2; 1 2];
>> A.*B
ans =
     1     4
     3     8
>> A./B
ans =
     1     1
     3     2
```

Elemente von A und B mit gleichem Index werden multipliziert/dividiert.

- Komponentenweises Potenzieren mit `.^`:

```
>> A.^B
ans =
     1     4
     3    16
```

Elemente von A werden jeweils mit dem Element aus B das denselben Index besitzt potenziert.

Spezielle Matrizen

Für häufig verwendete Matrizen gibt es Funktionen mit denen diese Matrizen erzeugt werden können:

- Einheitsmatrix bzw. -vektor:
  `eye(n)`, `eye(n,m)`
- Einsmatrix- bzw. -vektor:
  `ones(n)`, `ones(n,m)`
- Nullmatrix- bzw. -vektor:
  `zeros(n)`, `zeros(n,m)`
- Zufallsmatrix bzw. -vektor:
  `rand(n,m)`, `randn(n,m)`
- Magisches Quadrat:
  `magic(n)`

```
>> eye(2,3)
ans =
     1     0     0
     0     1     0
>> ones(3)
ans =
     1     1     1
     1     1     1
     1     1     1
>> zeros(1,4)
ans =
     0     0     0     0
>> rand(3)
ans =
    0.4898    0.7094    0.6797
    0.4456    0.7547    0.6551
    0.6463    0.2760    0.1626
>> magic(3)
ans =
     8     1     6
     3     5     7
     4     9     2
```

Spezielle Matrizen

- Diagonalmatrix zu einem Vektor mit Diagonalelementen oder Vektor der Diagonalelemente einer Matrix: `diag(x)`, `diag(A)`

Mit `gallery` können noch weitere spezielle Matrixformen erzeugt werden.
Hilfeseite: `>> help elmat`.

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> d=diag(A)
d =
     1
     5
     9
>> diag(d)
ans =
     1     0     0
     0     5     0
     0     0     9
>> diag(diag(A))
ans =
     1     0     0
     0     5     0
     0     0     9
>> gallery ('jordbloc' ,3 ,2)
ans =
     2     1     0
     0     2     1
     0     0     2
```

Lösen von Gleichungssystemen

Zum Lösen von linearen Gleichungssystemen ist in Matlab der \-Operator definiert. (Matlab löst das Gleichungssystem intern mit einer QR-Zerlegung falls die Matrix invertierbar ist, andernfalls wird ein least-squares Problem gelöst. Genaueres dazu spätestens in Numerik1.)

Bsp.: Löse Ax=b für

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$$
$$b = (-1, -1)^T$$

```
>> A=[1  2  ;0  1];
>> b=[-1  -1]';
>> x=A\b
x =
      1
     -1
>> A*x
ans  =
     -1
     -1
```

Mit dem \-Operator kann das Gleichungssystem auch für mehrere Vektoren b auf einmal gelöst werden. Dies kann z.B. dazu verwendet werden eine Inverse zu berechnen.
Bsp.: Löse

$$A \cdot X = I$$

für A wie oben und I = Einheitsmatrix.

```
>> I=eye(2);
>> X=A\I
X =
      1     -2
      0      1
>> A*X
ans  =
      1      0
      0      1
```

Funktionen für skalare Kenngrößen einer Matrix

$$\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

Funktionen, mit denen Matrizen charakterisiert werden, sind

die Spur `trace`

$$\begin{aligned} \mathrm{tr}(A) &= \sum_{i=1}^{n} a_{ii} \\ &= 8 + 5 + 2 = 15, \end{aligned}$$

```
>> trace(A)
ans =
    15
```

den Rang `rank`, also die Dimension des Bildraums einer Matrix

$$\dim(\mathrm{Bild}(A)),$$

```
>> rank(A)
ans =
     3
```

die Determinante `det`

$$\begin{aligned} \det(A) &= \sum_{\sigma \in S_n} \mathrm{sign}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)} \\ &= 8 \cdot 5 \cdot 2 + 1 \cdot 7 \cdot 4 + 6 \cdot 3 \cdot 9 \\ &\quad -4 \cdot 5 \cdot 6 - 9 \cdot 7 \cdot 8 - 2 \cdot 3 \cdot 1 = -360, \end{aligned}$$

```
>> det(A)
ans =
  -360
```

(wird nicht so berechnet!!)

Weitere Kenngrößen von Vektoren und Matrizen

$$\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

Die Inverse `inv` einer quadratischen Matrix:

$$A^{-1}$$

```
>> inv(A)
ans =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
```

Die Norm eines Vektors oder einer Matrix:

$$\|x\| = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}} = \sqrt{1+4+4} = 3,$$

$$\|A\| = \max_{\|x\|=1} \|Ax\|;$$

```
>> x=[1 2 2];
>> norm(x)
ans =
     3
>> norm(A)
ans =
    15
```

die Kondition `cond`

$$\kappa(A) = \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|}$$

```
>> cond(A)
ans =
    4.3301
```

Weitere Kenngrößen von Vektoren und Matrizen

$$\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

die Summe der Elemente eines Vektors

$$\mathrm{sum}(a) = 8 + 1 + 6 = 15$$

```
>> a=A(1,:)
a =
     8     1     6
>> sum(a)
ans =
    15
```

die Summe der Elemente in jeder Spalte einer Matrix

$$\begin{aligned} \mathrm{sum}(A) &= (8+3+4, 1+5+9, 6+7+2) \\ &= (15, 15, 15) \end{aligned}$$

```
>> sum(A)
ans =
    15    15    15
```

maximales bzw. minimales Element eines Vektors:

$$a = (8, 1, 6)$$

```
>> min(a)
ans =
     1
>> max(a)
ans =
     8
```

Einfache Skripte

- Matlab Befehle können in Textdateien mit Endung `.m` gespeichert und im Workspace durch Eingabe des Dateinamens (ohne Endung) ausgeführt werden. Dazu kann der Matlab Editor `edit` oder jeder andere Texteditor benutzt werden.

Kegel.m

```
% Berechnung des Volumens
% eines Kegels
r=3
h=5
V=1/3*r^2*h
```

```
>> Kegel
r =
     3
h =
     5
V =
    15
>>
```

- Zeilen, die mit einem % beginnen, werden als Kommentarzeilen behandelt.
- Beim Aufruf im Workspace werden alle Skripte im aktuellen Verzeichnis und im Suchpfad berücksichtigt.
- Mit `Edit <Dateiname>` wird der Matlab-Texteditor aufgerufen, `type <Skriptname>` zeigt den Inhalt eines m-Files an.
- Die Funktion `what` listet alle m-Files im aktuellen Verzeichnis auf.

Matlab Hilfe im Command Window

In Matlab gibt es ein umfassendes Hilfe-System um Informationen zu allen Funktionen zu bekommen. Es gibt verschiedene Möglichkeiten die Hilfe in Matlab zu nutzen:

- `help` oder `help <Thema>`
  Zeigt eine Übersicht über Hilfethemen oder über ein Thema bzw. einer Funktion im Command Window an;
- `lookfor <Text>`
  Sucht in den Kurzbeschreibungen der Funktionen nach `<Text>` ;

```
>> help sin
 SIN    Sine of argument in radians.
    SIN(X) is the sine of the elements of X.

    See also asin, sind.


    Reference page in Help browser
       doc sin

>> lookfor lookfor

LOOKFOR Search all M-files for keyword.
```

Matlab Hilfefenster

- `helpbrowser`

  Öffnet das graphische Hilfesystem;

- `doc <Thema>`

  Öffnet die Hilfe zum Thema oder zum Funktionsnamen im graphischen Hilfefenster zu einem Thema.

```
>> doc help
```