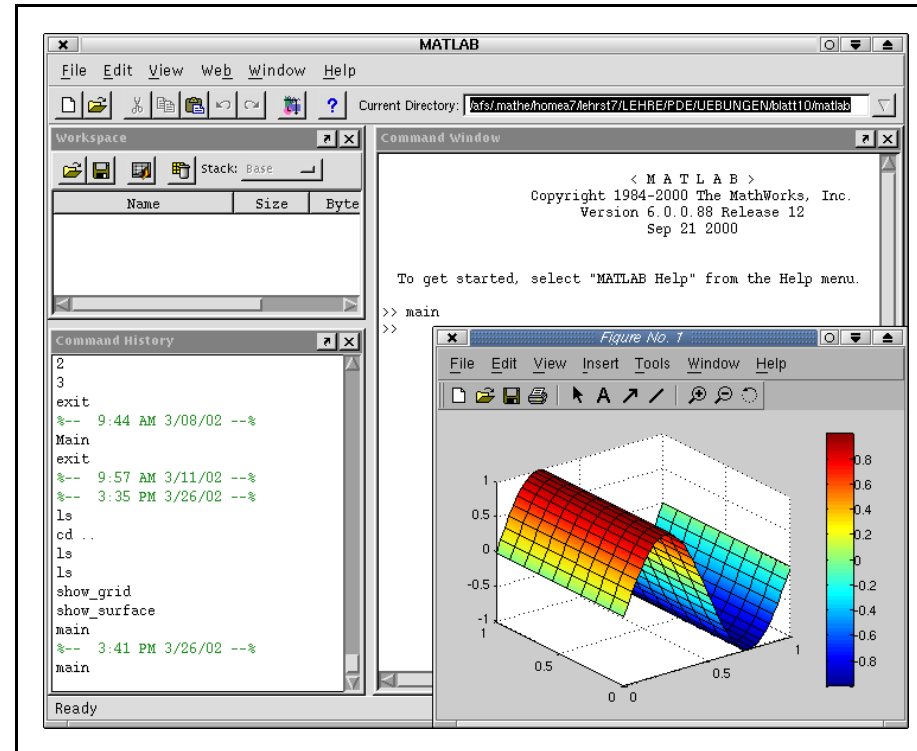


Einführung in Matlab 6

Andreas Klimke
Mittwoch, 17. April 2002





Übersicht

- 1 Was ist Matlab?
- 2 Starten und Bedienen von Matlab
- 3 Matlab-Hilfesystem und weiterführende Quellen
- 4 Zahlen, Operatoren, Funktionen
- 5 Matrizen und Matrizenoperationen
- 6 Grafik in Matlab
- 7 Programmierung



Was ist Matlab?

Softwaresystem für technische Berechnungen:

- Numerische Berechnungen
- Entwicklung von Algorithmen
- Visualisierung von Ergebnissen
- Modellierung und Simulation technischer Probleme
- Anwendungen mit graphischer Benutzeroberfläche

Beispiel: Numerische Berechnungen

Lösen eines Gleichungssystems.

```
>> A = gallery('poisson',4);  
>> f = ones(16,1);  
>> x = A\f
```

x =

```
0.8333  
1.1667  
1.1667  
0.8333  
...  
1.1667  
0.8333
```

```
>> full(A)  
ans =
```

```
4 -1 0 0 -1 0 0 0 0 0 0 0 0 0 0  
-1 4 -1 0 0 -1 0 0 0 0 0 0 0 0 0  
0 -1 4 -1 0 0 -1 0 0 0 0 0 0 0 0  
0 0 -1 4 0 0 0 -1 0 0 0 0 0 0 0  
-1 0 0 0 4 -1 0 0 -1 0 0 0 0 0 0  
0 -1 0 0 -1 4 -1 0 0 -1 0 0 0 0 0  
0 0 -1 0 0 -1 4 -1 0 0 -1 0 0 0 0  
0 0 0 -1 0 0 -1 4 0 0 0 -1 0 0 0  
0 0 0 0 -1 0 0 0 4 -1 0 0 -1 0 0  
0 0 0 0 0 -1 0 0 -1 4 -1 0 0 -1 0  
0 0 0 0 0 0 -1 0 0 -1 4 0 0 0 -1  
0 0 0 0 0 0 0 -1 0 0 0 4 -1 0 0  
0 0 0 0 0 0 0 0 -1 0 0 -1 4 -1 0  
0 0 0 0 0 0 0 0 0 -1 0 0 -1 4 -1  
0 0 0 0 0 0 0 0 0 0 -1 0 0 -1 4
```

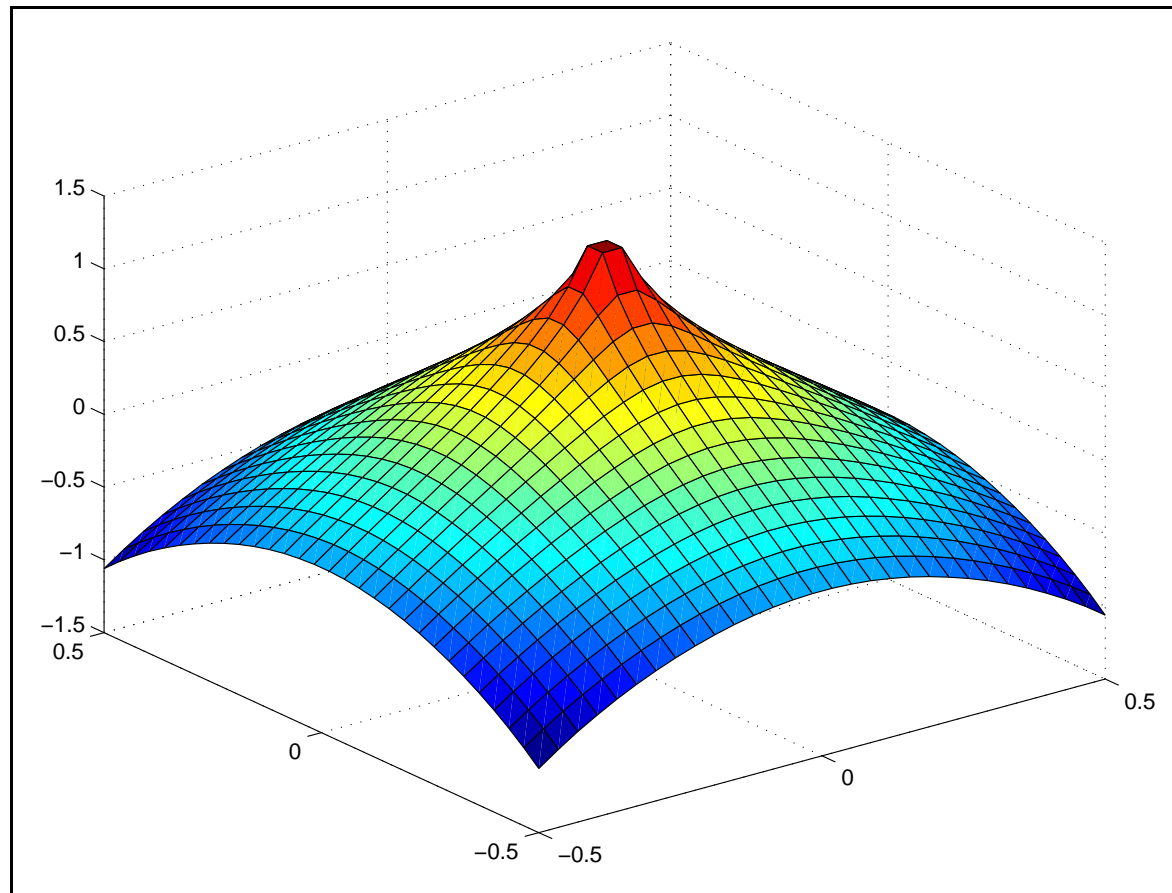
Beispiel: Entwicklung von Algorithmen

Iteratives Lösungsverfahren (PCG).

```
function [u,m] = solvePCG(A, f, u_s, C1, C2, tol, m_max)
    u = u_s; m = 0;
    r = f - A * u;
    p = C2 \ (C1 \ r);
    norm_f = norm(f);
    while( (norm(r)/norm_f > tol) & (m < m_max))
        a = A * p;
        a_dot_p = a' * p;
        lambda = (r' * p) / a_dot_p;
        u = u + lambda * p;
        r = r - lambda * a;
        inv_C_times_r = C2 \ (C1 \ r);
        p = inv_C_times_r - ((inv_C_times_r' * a) / a_dot_p) * p;
        m=m+1;
    end
```

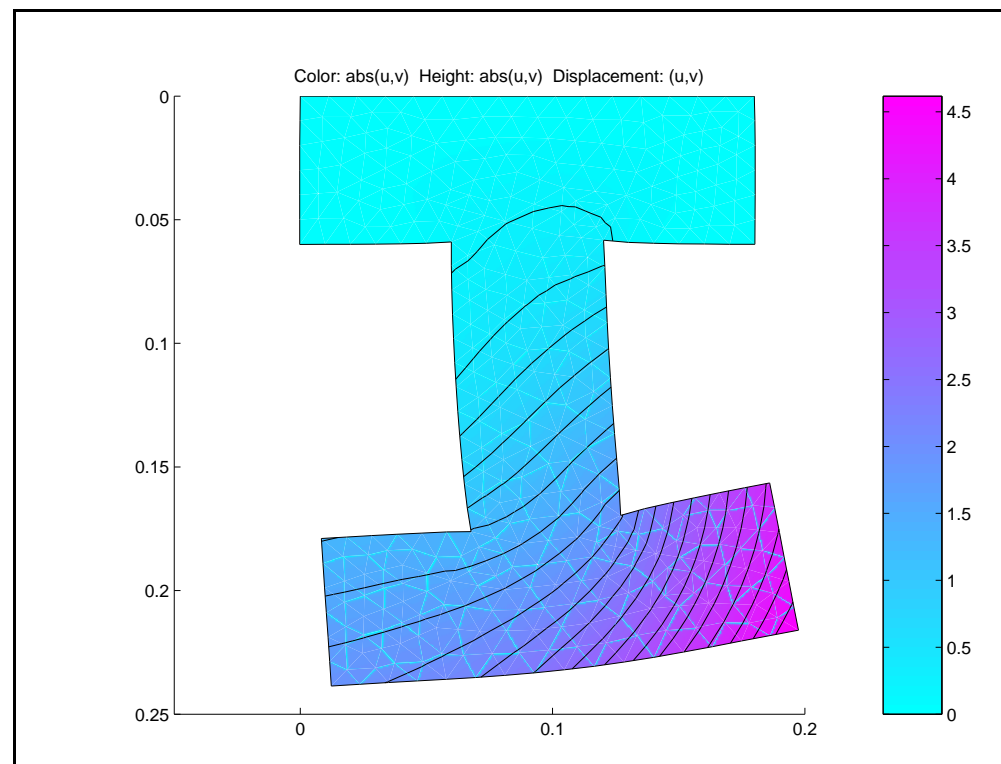
Beispiel: Visualisierung

3D-Plot der Funktion $\frac{1}{x^2+y^2}$.



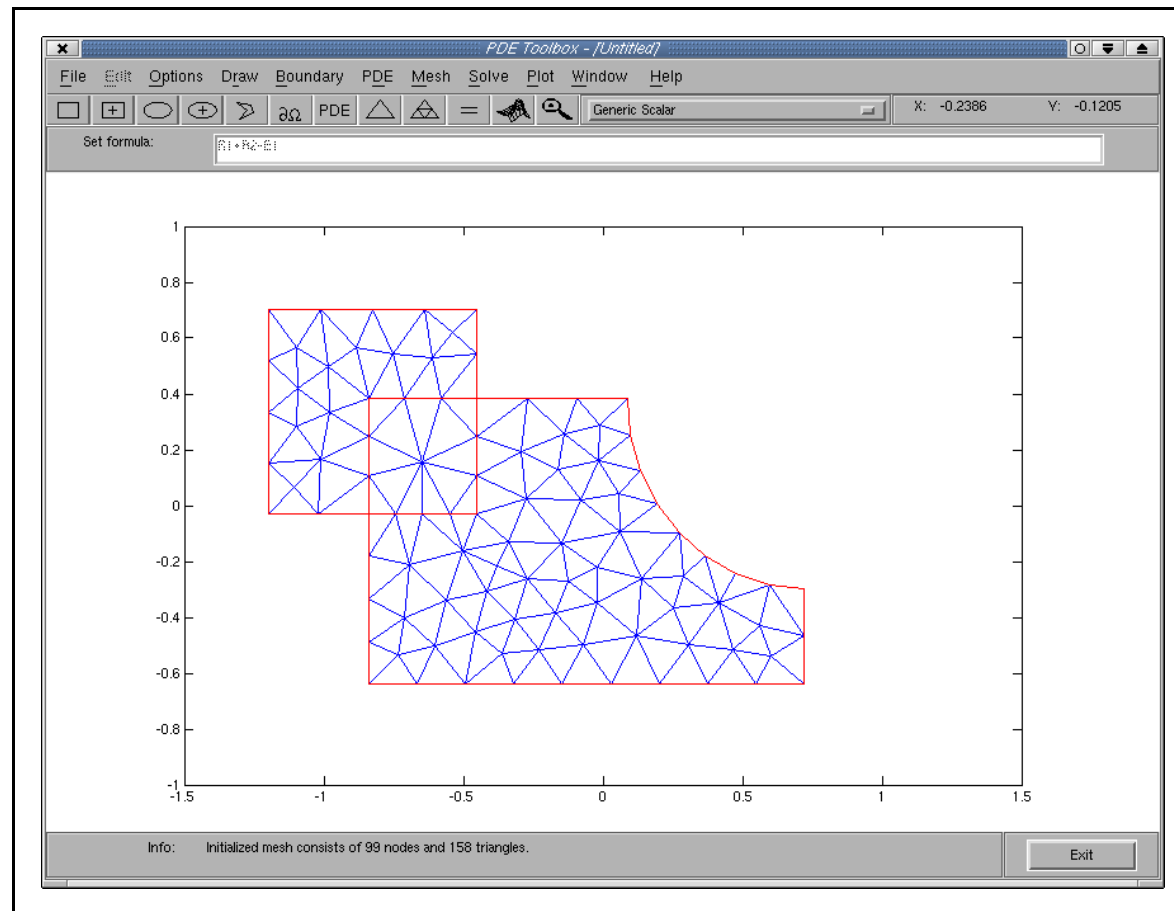
Beispiel: Modellierung und Simulation

Berechnen und Darstellen der Verschiebungsfigur eines Querschnitts unter einer Flächenlast.



Beispiel: GUI-Anwendung

Partial Differential Equations Toolbox *pde tool* von Matlab.





Matlab's Stärken

- Einfache, intuitive Bedienung
- Matrizenorientiert, damit besonders geeignet für lineare Algebra
- Vielzahl an implementierten Funktionen
- Sehr gute Grafikfähigkeiten
- Zuverlässiges, ausgereiftes System
- Leicht zu lernende, umfangreiche Programmierumgebung



Einschränkungen

- *Proprietary software*
- Lizenzen sind sehr teuer: 3000+ EUR; Studentenversion: 140 EUR (April 2002). Abhilfe: GNU Octave



Übersicht

- 1 Was ist Matlab?
- 2 Starten und Bedienen von Matlab
- 3 Matlab-Hilfesystem und weiterführende Quellen
- 4 Zahlen, Operatoren, Funktionen
- 5 Matrizen und Matrizenoperationen
- 6 Grafik in Matlab
- 7 Programmierung



Starten von Matlab

- Starten mit dem Befehl *matlab* für Matlab mit graphischer Benutzeroberfläche
- Starten mit *matlab -nojvm* für Kommandozeilenversion ohne Menüsystem

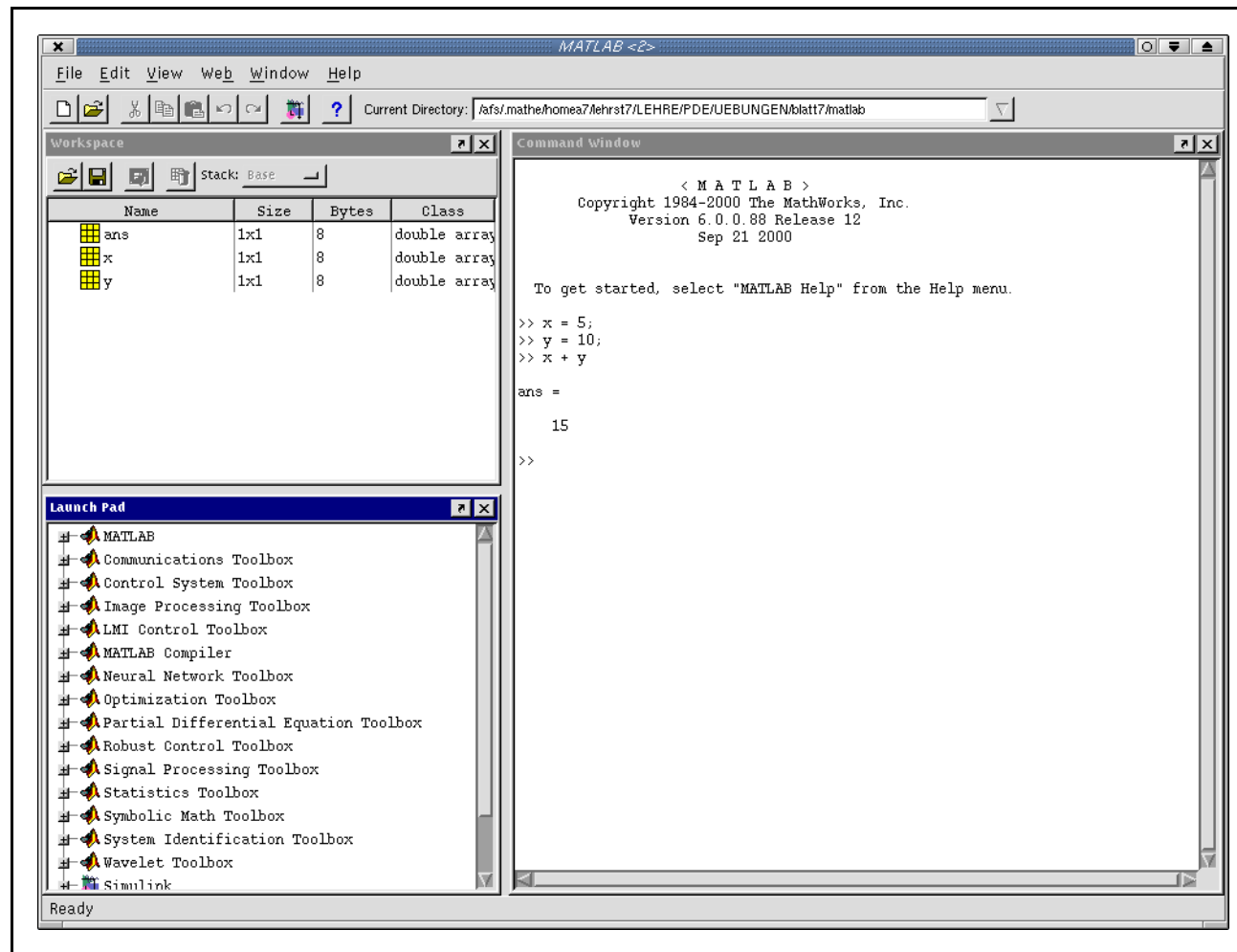


Matlab Views

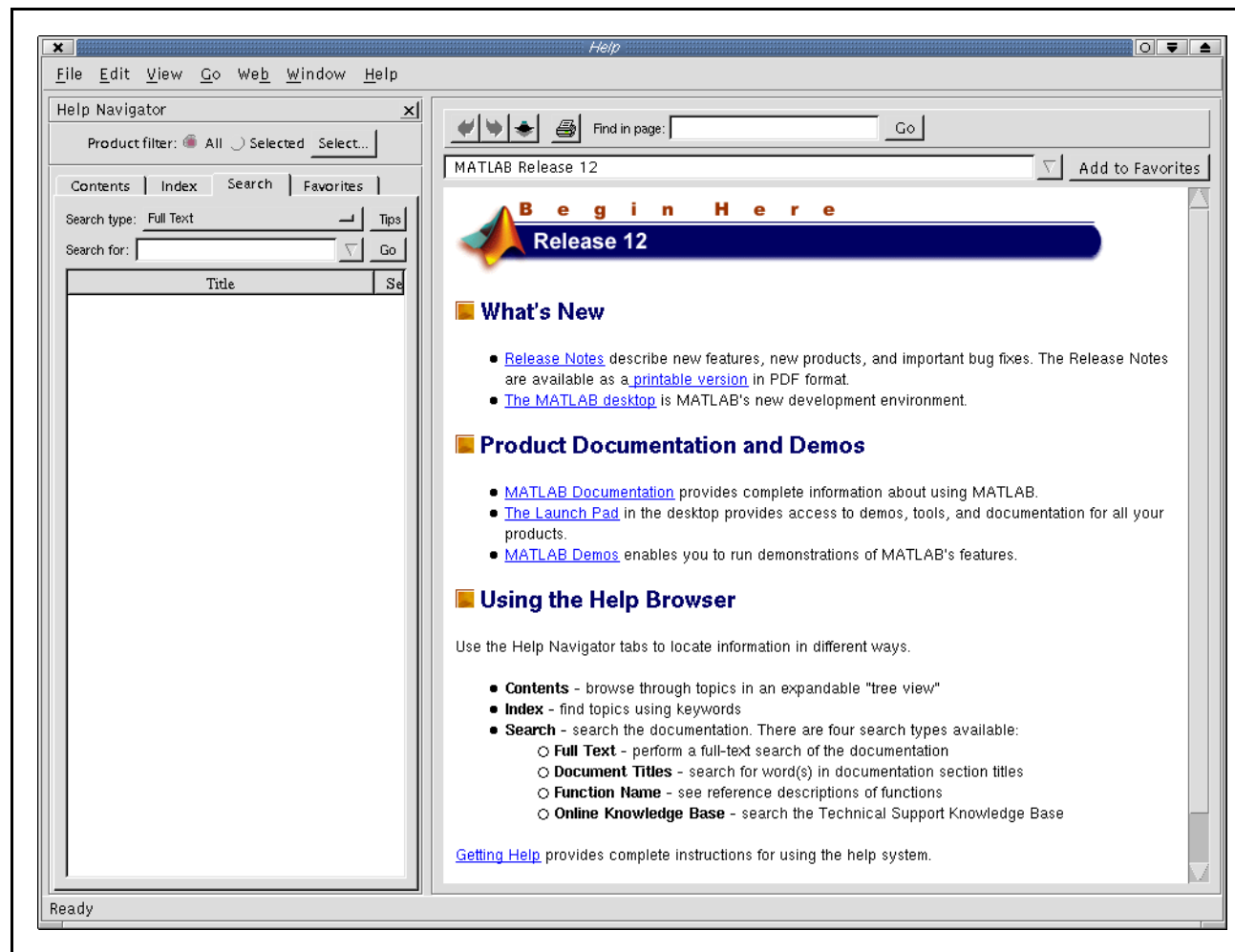
Arbeitsfenster der grafisch-interaktiven Benutzeroberfläche:

- command window
- command history window
- current directory window
- workspace
- launch pad
- help window

Command Window, Workspace, Launch Pad



Help Window



Befehlseingabe

- Prompt: >>
- Befehlswiederholung mit den Cursor-Tasten
(Einschränken der gefundenen Befehle möglich durch Angabe von Anfangsbuchstaben)
- Automatische Befehlsergänzung mit der Tabulatortaste
- Beenden mit *quit* oder *exit*

Eingabe von Daten

- Variablen

```
>> a = 2  
a =  
    2
```

- Vektoren

```
>> x = [1; 2; 3]  
x =  
    1  
    2  
    3
```

- Matrizen

```
>> A = [1,2,3;4,5,6;7,8,9]  
A =  
    1    2    3  
    4    5    6  
    7    8    9
```

Eingabe von Daten (2)

Anzeige eingegebener Daten:

```
>> whos
  Name      Size      Bytes  Class
  A         3x3         72    double array
  x         3x1         24    double array
  y         1x3         24    double array
Grand total is 15 elements using 120 bytes
>> A
A =
     1     2     3
     4     5     6
     7     8     9
```

Löschen aller Daten:

```
>> clear
```

Löschen einer Variablen:

```
>> clear <Variable>
```



Übersicht

- 1 Was ist Matlab?
- 2 Starten und Bedienen von Matlab
- 3 Matlab-Hilfesystem und weiterführende Quellen
- 4 Zahlen, Operatoren, Funktionen
- 5 Matrizen und Matrizenoperationen
- 6 Grafik in Matlab
- 7 Programmierung



Matlab-Hilfesystem

Matlab stellt eine umfangreiche Hilfe bereit.

Die wichtigsten Befehle:

Befehl	Beschreibung
help	Inhaltsverzeichnis der Hilfe-Funktion
help general	Hilfe zu allgemeinen Befehlen
help ops	Hilfe zu den Operatoren
help elmat	Hilfe zu Matrizen und deren Manipulation
help elfun	Hilfe zu den Elementarfunktionen
help matfun	Hilfe zu den besonderen Funktionen der linearen Algebra
helpwin	Hilfe im Fenster mit Mausbedienung
helpdesk	Menugesteuertes Hilfesystem
lookfor <Stichwort>	Stichwortsuche in den Hilfetexten aller Funktionen
demo	Menu zum Finden und Ausführen von Demo-Programmen



Literatur & Links

- Gramlich, Günter; Wilhelm, Werner: Numerische Mathematik mit Matlab
- Überhuber, Christoph; Katzenbeisser, Stefan: Matlab 6: Eine Einführung
- Benker, Hans: Mathematik mit Matlab. Eine Einführung für Ingenieure und Naturwissenschaftler
- Online-Tutorial: A Practical Introduction to Matlab
<http://www.math.mtu.edu/msgocken/intro/intro.html>
- Offizielle MathWorks Produktinformationen & Dokumentation
<http://www.mathworks.com/products/matlab/>
- Matlab-Einführung SS 2001 (Höllig/Hörner)
http://www.mathematik.uni-stuttgart.de/studium/infomat/Matlab_Kurs_SS01/
- Introduction to Matlab (University of Queensland)
<http://www.maths.uq.edu.au/gac/mlb/mlb.html>



Übersicht

- 1 Was ist Matlab?
- 2 Starten und Bedienen von Matlab
- 3 Matlab-Hilfesystem und weiterführende Quellen
- 4 Zahlen, Operatoren, Funktionen
- 5 Matrizen und Matrizenoperationen
- 6 Grafik in Matlab
- 7 Programmierung

Zahlen in Matlab

- Eingabe im Dezimalsystem
- Format: [Vorzeichen] Zahl [Exponent]
 - Vorzeichen: + oder -
 - Zahl: mind. 1 Ziffer, ggf. Dezimalpunkt
 - Exponent: e oder E gefolgt von + oder - und ganzer Zahl
- Speicherplatzbedarf: je 8 Bytes
- Zahlen in Matlab werden wie 1x1-Matrizen behandelt.
- Achtung: Begrenzte Genauigkeit!



Zahlen-Operatoren

- Addition: +

Beispiele:

```
>> 3\15
```

```
ans =
```

```
5
```

- Subtraktion: –

```
>> 15/3
```

```
ans =
```

```
5
```

- Multiplikation: *

- Division links nach rechts: /

```
>> 2*4+25
```

```
ans =
```

```
33
```

- Division rechts nach links: \

```
>> 2^4
```

```
ans =
```

```
16
```

- Potenz: ^

Zahlen-Elementarfunktionen

Funktion	Beschreibung	Wert
pi	die Zahl π	3.14159...
eps	relative Rechengenauigkeit	2.2204e-16
realmin	kleinste darstellbare Zahl >0	2.2251e-308
realmax	größte Darstellbare Zahl	1.7977e+308
i, j	Kennzeichnung des Imaginärteils	-

Beispiele:

```
>> sqrt(-4)
```

```
ans =
```

```
0 + 2.0000i
```

```
>> (2i)^2
```

```
ans =
```

```
-4
```

```
>> 2*pi
```

```
ans =
```

```
6.2832
```

Zahlen-Elementarfunktionen (2)

Trigonometrische Funktionen:

Funktion	Beschreibung
sin	Sinusfunktion
sinh	hyperbolischer Sinus
asin	inverser Sinus
asinh	inverser hyperbolischer Sinus

Analog: cos, tan, cot, sec, csc

Beispiele:

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

```
>> asin(1)
ans =
    1.5708
```

Zahlen-Elementarfunktionen (3)

Exponential- und Logarithmusfunktionen:

Funktion	Beschreibung
exp	Exponentialfunktion
log	natürlicher Logarithmus
log10	Logarithmus zur Basis 10
log2	Logarithmus zur Basis 2
sqrt	Quadratwurzel

Weitere Elementarfunktionen siehe:

<http://www.mathworks.de/access/helpdesk/help/techdoc/ref/ref.shtml>

Beispiele:

```
>> exp(1)
ans =
    2.7183
```

```
>> log2(8)
ans =
    3
```



Übersicht

- 1 Was ist Matlab?
- 2 Starten und Bedienen von Matlab
- 3 Matlab-Hilfesystem und weiterführende Quellen
- 4 Zahlen, Operatoren, Funktionen
- 5 Matrizen und Matrizenoperationen
- 6 Grafik in Matlab
- 7 Programmierung

Matrizen und Vektoren

- Matrizen sind die Haupt-Datenform in Matlab
- Eingabeformat:
 - Werte eingeschlossen in []
 - Zeilen durch ; oder Zeilenvorschub (Return) getrennt
 - Zeileneinträge durch , oder Leerzeichen getrennt
 - Zeilenfortsetzung durch ...
- Vektoren sind $1 \times n$ bzw. $n \times 1$ Matrizen
- Skalare sind 1×1 Matrizen (Klammern können entfallen)
- Indizierung: erster Index 1, letzter Index *end*

Spezielle Matrizen

Funktion	Beschreibung
ones(n,m)	n x m Matrix mit allen Einträgen = 1
zeros(n,m)	n x m Matrix mit allen Einträgen = 0
eye(n,m)	Hauptdiagonale = 1, sonst 0
rand(n,m)	Matrix mit Zufallswerten zwischen 0 und 1

Beispiele:

```
>> ones(2,3)
```

```
ans =
```

```
    1    1    1
    1    1    1
```

```
>> eye(3)
```

```
ans =
```

```
    1    0    0
    0    1    0
    0    0    1
```

Spezielle Vektoren

Funktion	Beschreibung
a:s:b	Vektor $[a, a+s, a+2s, a+3s, \dots, b]$
a:b	entspricht a:1:b
linspace(a,b,n)	Vektor mit n Werten, die $[a,b]$ linear unterteilen
logspace(a,b,n)	Vektor mit n Werten, die $[10^a, 10^b]$ logarithmisch unterteilen

Beispiele:

```
>> x = 1:5
```

```
x =
```

```
1     2     3     4     5
```

```
>> x = logspace(0,3,4)
```

```
x =
```

```
1     10    100   1000
```

Matrixoperationen

Matlab	Operation	Equivalent zu
$A + B$	Addition	$A + B$
$A - B$	Subtraktion	$A - B$
$A * B$	Multiplikation	AB
A / f'	rechte Division	Löst $x^T A = f^T$ nach x^T
$A \setminus f$	linke Division	Löst $Ax = f$ nach x
$A \wedge p$	Potenzieren	A^p
A'	konjugiert Transponieren	$(\overline{A})^T$
$A.'$	Transponieren	A^T
$\text{kron}(A,B)$	Kronecker Tensorprodukt	$A \otimes B$

Matrixoperationen (2)

Elementweise Operatoren

Matlab	Operation	Equivalent zu
$C = A .* B$	Hadamard-Produkt	$c_{ij} = a_{ij} * b_{ij}$
$C = A ./ B$	rechte Division	$c_{ij} = \frac{a_{ij}}{b_{ij}}$
$C = A .\ B$	linke Division	$c_{ij} = \frac{b_{ij}}{a_{ij}}$
$C = A .^ p$	Potenzieren	$c_{ij} = a_{ij}^{p_{ij}}$

Anmerkung:

Funktionen wie *sin*, *log*, *exp* können auch auf Matrizen angewendet werden. Diese werden ebenfalls **elementweise** ausgewertet.

Matrixoperationen - Beispiele

```
>> A = [1,2,3;4,5,6;7,8,9]
```

```
A =
```

```
    1    2    3
    4    5    6
    7    8    9
```

```
>> A.*B
```

```
ans =
```

```
    1    0    0
    0    5    0
    0    0    9
```

```
>> B = eye(3)
```

```
B =
```

```
    1    0    0
    0    1    0
    0    0    1
```

```
>> A'
```

```
ans =
```

```
    1    4    7
    2    5    8
    3    6    9
```

```
>> A*B
```

```
ans =
```

```
    1    2    3
    4    5    6
    7    8    9
```

```
>> asin(B)
```

```
ans =
```

```
    1.5708    0    0
    0    1.5708    0
    0    0    1.5708
```

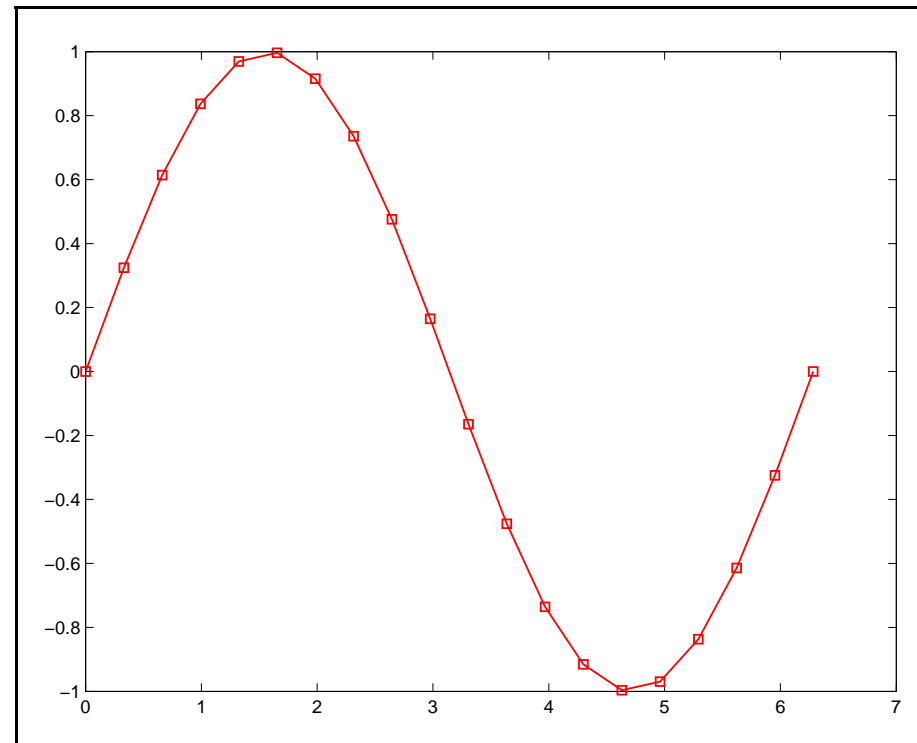


Übersicht

- 1 Was ist Matlab?
- 2 Starten und Bedienen von Matlab
- 3 Matlab-Hilfesystem und weiterführende Quellen
- 4 Zahlen, Operatoren, Funktionen
- 5 Matrizen und Matrizenoperationen
- 6 Grafik in Matlab
- 7 Programmierung

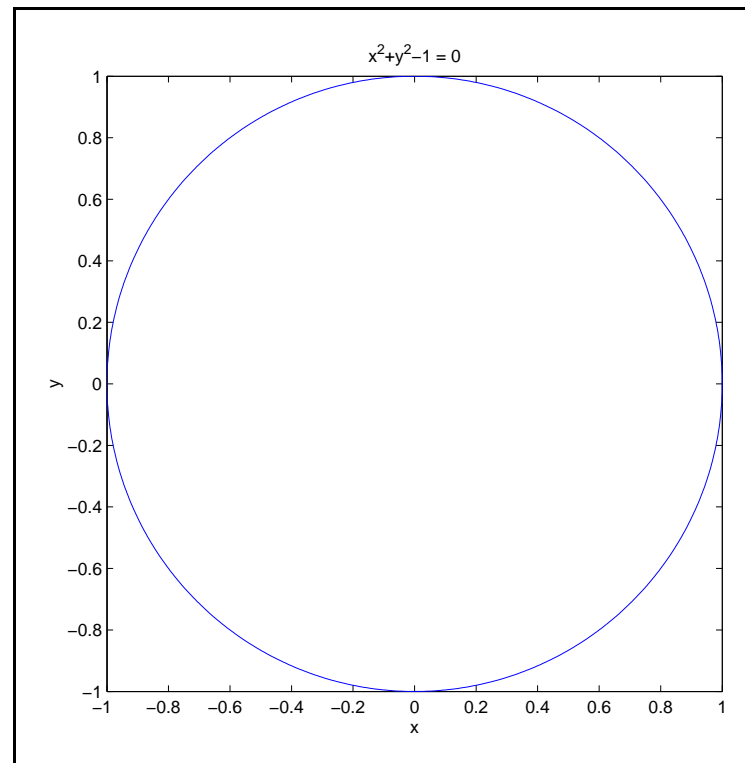
Plotten von Funktionen mit einer Veränderlichen

```
>> x = linspace(0,2*pi,20);  
>> y = sin(x);  
>> plot(x,y,'rs-')
```



Plotten von implizit definierten Funktionen in 2D

```
>> ezplot('x^2+y^2-1', [-1,1]);
```



3D-Grafik

- zum Auswerten einer Funktion $f(x, y)$ (zwei unabhängige Variablen) zunächst ein 2D-Gitter erzeugen:

```
>> x = 0:3;  
>> y = 0:3;  
>> [X,Y] = meshgrid(x,y);
```

$$X = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{pmatrix}$$

- dann Funktion auswerten:

```
>> Z = 1./(1+X.^2+Y.^2)
```

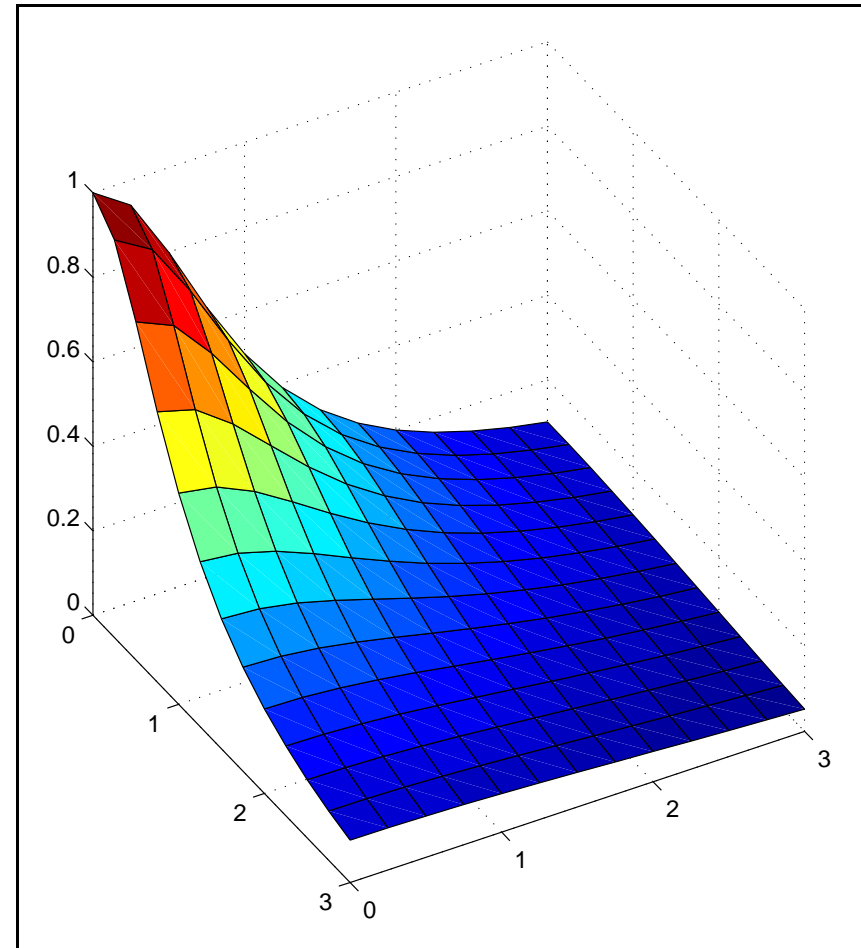
$$Z = \begin{pmatrix} 1 & 1/2 & 1/5 & 1/10 \\ 1 & 1/3 & 1/6 & 1/11 \\ 1 & 1/6 & 1/9 & 1/14 \\ 1 & 1/11 & 1/14 & 1/19 \end{pmatrix}$$

- zuletzt plotten:

```
>> surf(X,Y,Z)
```

oder

```
>> mesh(X,Y,Z)
```



Formatierung und Beschriftung der Grafiken

- Titel einfügen:

```
>> title('Verlauf der Funktion  $y=x^2$ ')
```

- Beschriftung der x- (bzw. y-) Achse:

```
>> xlabel('Zeit in [s]')
```

- Text an spezifizierten Koordinaten einfügen:

```
>> text(1,1,'Wertebereich: [0,1]')
```

- Text interaktiv mit der Maus einfügen (2D):

```
>> gtext('Funktion f')
```

- Gitter ein/ausblenden:

```
>> grid
```

- Farbwerteskala einblenden:

```
>> colorbar
```



- gleiches Achsenverhältnis einstellen:

```
>> axis square
```

- Weitere Plots in die bestehende Grafik einfügen:

```
>> hold on
```

(danach mit *hold off* wieder deaktivieren)

- Für weitere Details zu Grafik in Matlab, siehe

http://www.mathworks.com/access/helpdesk/help/techdoc/creating_plots/creating_plots.shtml



Übersicht

- 1 Was ist Matlab?
- 2 Starten und Bedienen von Matlab
- 3 Matlab-Hilfesystem und weiterführende Quellen
- 4 Zahlen, Operatoren, Funktionen
- 5 Matrizen und Matrizenoperationen
- 6 Grafik in Matlab
- 7 Programmierung

m-Files: Scripte und Funktionen

- interaktive Arbeitsweise mit Command-Prompt ungeeignet für Algorithmen und Programme, die mehrere Zeilen benötigen
- m-Files können mit einem Texteditor (z.b. Notepad unter Windows oder emacs unter Unix) erzeugt und abgespeichert werden
- Filenamen mit Kürzel *.m*
- Man unterscheidet zwei Arten von *.m* -Files:
 - Script-Files
 - Function-Files

Scripte

- Folge von gewöhnlichen Matlab-Anweisungen
- Anweisungen im Script-File werden ausgeführt bei Eingabe des File-Namen ohne Kürzel *.m*.
- Man kann auch andere *.m*-Files von einem Script aus aufrufen.
- **Beispiel:**
 - Filename: *main.m*
 - Programm-Code:

```
disp('Auflösen eines linearen Gleichungssystems')
A = input('Bitte Matrix A eingeben (z.B. [1 2; 3 0]): ')
f = input('Nun die rechte Seite f eingeben (z.B. [1;3]): ')
disp('Der Lösungsvektor lautet: ')
x = A\f
disp('Bitte irgendeine Taste drücken...')
pause
disp('Fertig!')
```



– Ausführen in Matlab:

```
>> main
```

– Ausgabe:

Auflösen eines linearen Gleichungssystems

Bitte Matrix A eingeben (z.B. [1 2; 3 0]): [1 2; 3 0]

A =

```
    1    2
    3    0
```

Nun die rechte Seite f eingeben (z.B. [1;3]): [1; 3]

f =

```
    1
    3
```

Der Lösungsvektor lautet:

x =

```
    1
    0
```

Bitte irgendeine Taste drücken...

Fertig!

- Variablen in einem Script-File sind global
- Variablen des Workspace können verwendet werden
- Kontrollfluss-Befehle für Scripte:

Matlab-Funktion	Beschreibung
disp	String ausgeben
input	Auf Eingabe vom Benutzer warten und in Variable speichern (siehe Beispiel <i>main.m</i>)
keyboard	Bearbeitung des Script-Files unterbrechen und Benutzer die Kontrolle übertragen (Rückkehr zum Programm mit <i>return</i>)
pause	auf Tastendruck warten
pause(n)	n Sekunden warten

Funktionen

- dienen zum Erstellen eigener Funktionen, die wie andere Matlab-Kommandos verwendet werden können
- somit Erweiterung des Matlab-Funktionenvorrats
- Variablen sind lokal
- damit ein File als Function-File erkannt wird, muss es mit dem Schlüsselwort *function* beginnen
- Allgemeine Form eines Function-Files:

```
function [out_1, ..., out_n] = name(in_1, ..., in_m)
    <irgendwelche Anweisungen>
```

● **Beispiel 1:**

– File-Name: *month.m*

– Programm-Code:

```
function [monthName, days] = month(number)
% MONTH Determine month name and number of days/month.
% [M,D] = MONTH(N) ermittelt die Anzahl der Tage D und
% den Monatsnamen M für eine gegebene Monatsnummer.

MONTH = { 'Januar', 'Februar', 'März', 'April', ...
          'Mai', 'Juni', 'Juli', 'August', 'September', ...
          'Oktober', 'November', 'Dezember' };
DAYS = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];
if (number < 1 | number > 12)
    error('Fehler: Monatsnummer muss zw. 1 und 12 liegen!')
end
monthName = MONTH{number};
days = DAYS(number);
```



– Ausführen in Matlab:

```
>> [M,D] = month(10)
```

– Ausgabe:

M =

Oktober

D =

31

● **Beispiel 2:**

- File-Name: *f.m*
- Beschreibung:
definiert die stückweise zusammengesetzte Funktion

$$f(x) = \begin{cases} x^2 & x < 0 \\ x & x \geq 0 \end{cases}$$

- Programm-Code:

```
function y = f(x)
% F    Stückweise zusammengesetzte Funktion
y1 = x^2 * (x < 0);
y2 = x * (x >= 0);
y = y1+y2;
```

- Aufruf in Matlab:

```
>> f(-2)
```

- Ausgabe:

```
ans = 4
```

Eigenschaften von Funktionen

- File- und Funktionsname müssen übereinstimmen
- das Schlüsselwort *function* muss in der ersten Zeile stehen
- die Kommentarzeilen nach der ersten Zeile werden bei Aufruf von *help <Funktionsname>* angezeigt
- ein Function-File kann mehrere Unterfunktionen enthalten, die von der Hauptfunktion aufgerufen werden können. Die Unterfunktionen können sich auch gegenseitig aufrufen
- ein Function-File kann andere Function-Files oder Scripte aufrufen
- das Function-File muss kein Ein- oder Ausgabeargument haben
- jedes Function-File hat einen eigenen, lokalen Workspace für die Variablen, mit dem Schlüsselwort *global* können jedoch globale Variablen deklariert werden

Vergleichsoperatoren und Logische Operatoren

- Vergleichsoperatoren:

Operator	Beschreibung	Math. Symbol
<	kleiner	<
<=	kleiner gleich	≤
>	größer	>
>=	größer gleich	≥
==	gleich	=
~=	ungleich	≠

- Logische Operatoren:

Operator	Beschreibung
&	logisches UND
	logisches ODER
~	logisches NICHT

Beispiele

- Vergleich zweier Zahlen:

```
>> x = 3;
```

```
>> x == 3
```

```
ans =
```

```
1
```

```
>> x > 1
```

```
ans =
```

```
1
```

```
>> x == 2
```

```
ans =
```

```
0
```

- Vergleich von Matrizen:

```
>> x = [ 1 2 4 ];
```

```
>> y = [ 0 2 3 ];
```

```
>> x == y
```

```
ans =
```

```
0 1 0
```

- Logisches ODER:

```
>> x = [ 0 2 2 ];
```

```
>> y = [ 0 1 0 ];
```

```
>> x | y
```

```
ans =
```

```
0 1 1
```

- Logisches UND:

```
>> x = [ 0 2 2 ];
```

```
>> y = [ 0 1 0 ];
```

```
>> x & y
```

```
ans =
```

```
0 1 0
```

Steuerstrukturen

- Steuerung des Programms durch **Schleifen** und **Verzweigungen**
- Schleifen werden gebildet, um eine Gruppe von Befehlen mehrfach auszuführen
- Verzweigungen werden eingefügt, um eine Gruppe von Befehlen nur unter bestimmten Bedingungen auszuführen (Fallunterscheidung)
- In Matlab gibt es 4 Möglichkeiten:
 - *for*-Schleife
 - *while*-Schleife
 - Verzweigung mit *if*
 - Verzweigung mit *switch* (hier nicht behandelt)

die for-Schleife

- Syntax:

```
for <Variable x> = <Matrix M>  
    <Anweisungen>  
end
```

- Anweisungen in der for-Schleife ausführen, und zwar so oft, wie die Matrix M Spalten hat
- In jedem Durchlauf wird x gleich der nächsten Spalte der Matrix M gesetzt
- x ist in jedem Durchlauf entweder ein Spaltenvektor oder ein Skalar (wenn die Matrix M nur **eine Zeile** hat)
- können geschachtelt werden

for-Schleife: Beispiele

- Berechnung von 5!
(Anmerkung:
effizienter wäre die Verwendung
des Befehls *factorial(5)*)

```
>> x = 1;  
>> for k = 1:5  
    x = x * k;  
end  
>> x  
x =  
    120
```

- Erzeugen der Hilbert-Matrix der Ordnung 3:

$$H_3 = \begin{pmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{pmatrix}$$

```
>> n = 3;  
>> for k = 1:n  
    for l = 1:n  
        H(k,l) = 1/(k+l-1);  
    end  
end  
>> H  
H =  
    1.0000    0.5000    0.3333  
    0.5000    0.3333    0.2500  
    0.3333    0.2500    0.2000
```

die while-Schleife

- Syntax:

```
while <Testausdruck>  
    <Anweisungen>  
end
```

- die Gruppe der Anweisungen wird ausgeführt, solange der Testausdruck zu WAHR ($\neq 0$) ausgewertet wird
- die Auswertung des Testausdrucks erfolgt jeweils **bevor** Verarbeitung der Anweisungen
- ist der Testausdruck FALSCH ($= 0$), wird im Programm mit dem ersten Befehl nach *end* fortgefahren

while-Schleife: Beispiel

- Beispiel: Anzeigen aller Primzahlen zwischen 100 und 200

```
>> x = 101;           x =  
                        101  
>> while ( x<200 )   x =  
    if (isprime(x))   103  
        x            x =  
    end              107  
    x = x + 2;       109  
>> end               x =  
                        113  
                        ...  
x =  
    199
```

Verzweigung mit if

- Syntax:

```
if <Testausdruck>  
    <Anweisungen>  
end
```

oder:

```
if <Testausdruck>  
    <Anweisungen>  
else  
    <Anweisungen>  
end
```

oder:

```
if <Testausdruck 1>  
    <Anweisungen>  
elseif <Testausdruck 2>  
    <Anweisungen>  
...  
elseif <Testausdruck n>  
    <Anweisungen>  
end
```

- der Testausdruck wird ausgewertet (WAHR oder UNWAHR), dann erfolgt Abarbeitung der Anweisungen entsprechend

Verzweigung mit if: Beispiel

- Test, ob Zahl gerade oder ungerade ist:

```
>> n = 50;  
>> if ( rem(n,2) == 0 )  
        disp('n ist gerade')  
    else  
        disp('n ist ungerade')  
    end
```

```
n ist gerade
```