Einführung in MATLAB (MATrix LABoratory)

Allgemein

- MATLAB ist eine Software zur
  - numerischen Lösung mathematischer Probleme
  - grafische Darstellung von Ergebnissen
  - prototyping von Algorithmen
  - ...

Allgemein

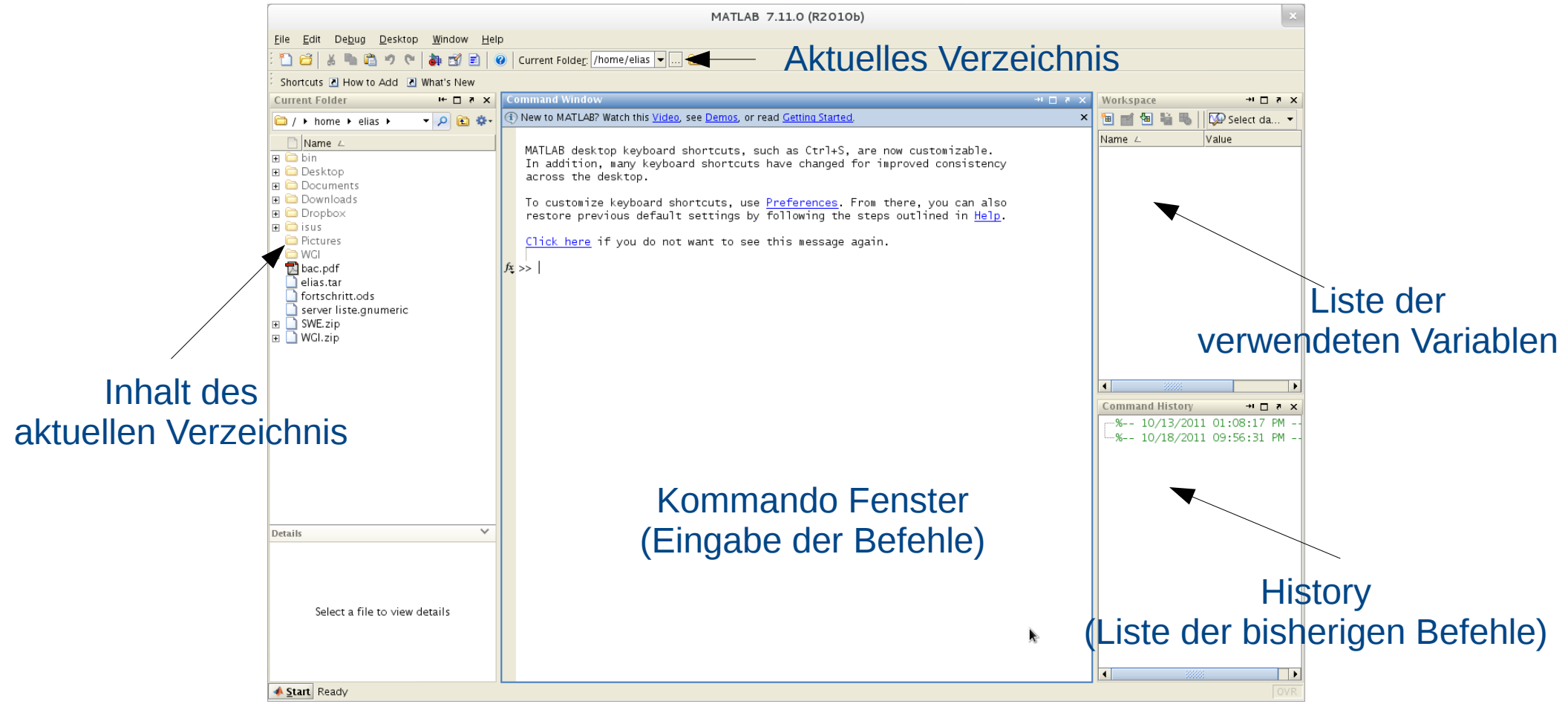
- MATLAB bietet
  - eine Hochsprache für technische Probleme
  - eine grafische Entwicklungsumgebung (Desktop)
  - Interaktive Tools zum Erlernen und Lösen von mathematischen Problemen
  - mathematische Funktionen für Lineare Algebra, Statistik, Analysis, Optimierung usw.
  - 2D- und 3D-Visualisierungsfunktionen
  - Einen Builder für grafische Oberflächen
  - Anbindung für Programmiersprachen, wie C/C++, Fortran, Java

Allgemein

- Anwendungen
  - Matrixberechnung
  - numerische Analysen
  - Beschaffung, Untersuchung und Analyse von Daten
  - Datenvisualisierung (grafische Darstellung)
  - Modellierung und Simulation
  - ...

Start

- beim Start erscheint der MATLAB-Desktop



MATLAB-Sprache

- MATLAB ist eine Ausdruckssprache
- Syntax
  - <Variable = Ausdruck>
    - Auswertung des Ausdrucks wird der Variable zugeordnet
  - <Ausdruck>
    - Auswertung wird automatisch der Variable ans (answer) zugewiesen
- MATLAB ist case-sensitiv, d.h. Groß- und Kleinschreibung wird beachtet!
  - z.B. A ist nicht gleich a

MATLAB-Sprache

- Beispiel
  - `>> 3+4/5`

    ```
    ans =

        3.8000
    ```
  - `>> a=3+4/5`

    ```
    a =

        3.8000
    ```

Kommando Fenster

- Recall
  - Abrufen voriger Befehlszeilen mit der Pfeil-Rauf und Pfeil-Runter – Taste
  - Bearbeiten dieser Befehle
  - Ausführen des neuen Befehls mit [Enter]-Taste
- Hilfe
  - help … Listet alle Hilfe Themen auf
  - help <themenname> … listet die Funkionsnamen zum Thema
  - help <funktionsname> … zeigt Informationen über die gewünschte Funktion an
  - helpdesk … Hilfebrowser

Variablen

who, whos … Listet alle aktuellen Variablen im Arbeitsbereich auf

clear <variablenname> … löscht die angegebene Variable aus dem Arbeitsbereich

clearall … löscht alle Variablen im Arbeitsbereich

MATLAB-Sprache

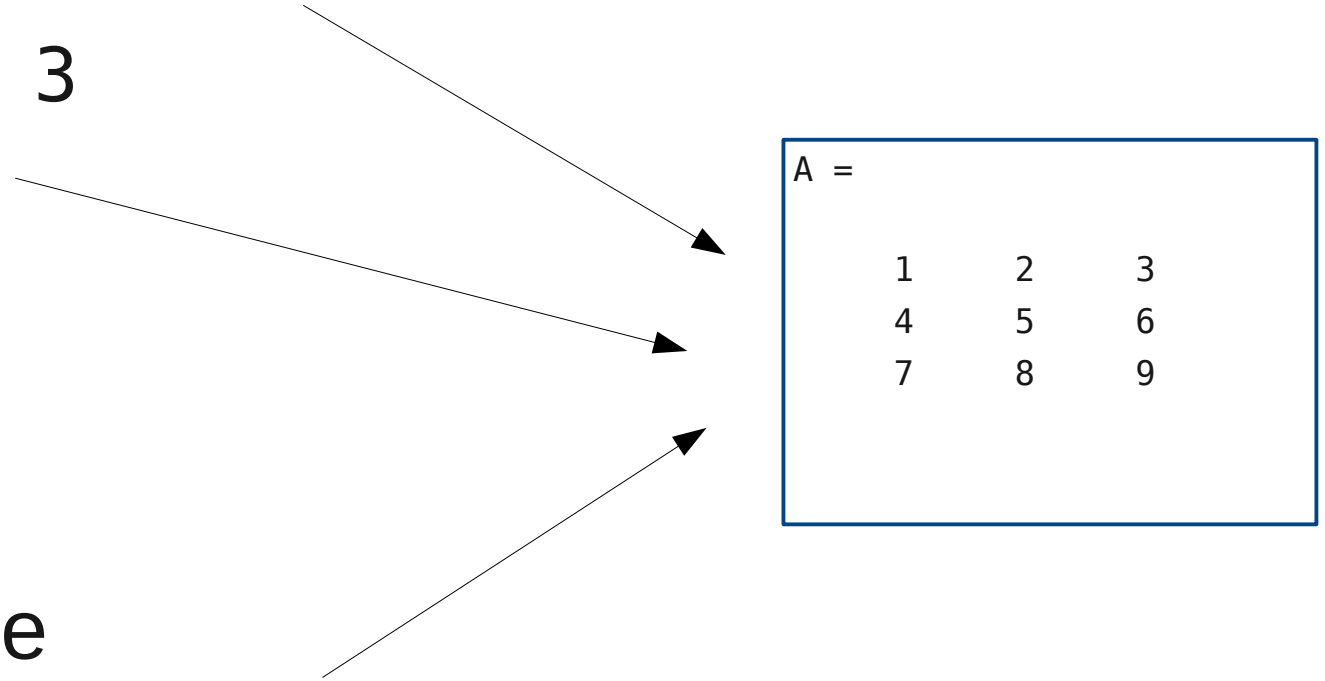
- alle Variablen in MATLAB repräsentieren Matrizen
- Skalar wird als 1x1-Matrix betrachtet
- Matrizen mit einer Zeile oder einer Spalte sind Vektoren

Eingabe von Vektoren

- Zeilenvektor
  - `V = [ 1 2 3 4]`, bzw. `V = [1,2,3,4]`
- Spaltenvektor
  - `V = [1;2;3;4]`
- Doppelpunkt-Notation
  - v = [0:0.1:1] → v = 0.0 0.1 0.2 0.3 0.4 … 0.9 1.0
  - [Anfang:Inkrement:Ende]
    - Ohne Angabe von Inkrement wird um 1 erhöht

Eingabe von Matrizen

- Zeilenweise
  - `A = [1 2 3; 4 5 6; 7 8 9]`
  - `A = [1 2 3`
`4 5 6`
`7 8 9]`



- Spaltenweise
  - `A = [[1;4;7] [2;5;8] [3;6;9]]`

Erzeugen von Matrizen

- eye() ... Einheitsmatrix
- zeros() ... Matrix mit Nullen
- ones() ... Matrix mit Einsen
- rand() ... zufällig generierte Matrix
- ...

Erzeugen von Matrizen

- rand(n)
  - nxn-Matrix, mit zufällig generierten Elementen zwischen 0 und 1
- rand(m,n)
  - mxn-Matrix, mit zufällig generierten Elementen zwischen 0 und 1
- sprand()
  - schwach besetzte Matrix
- sprandsym()
  - symmetrische schwach besetzte Matrix

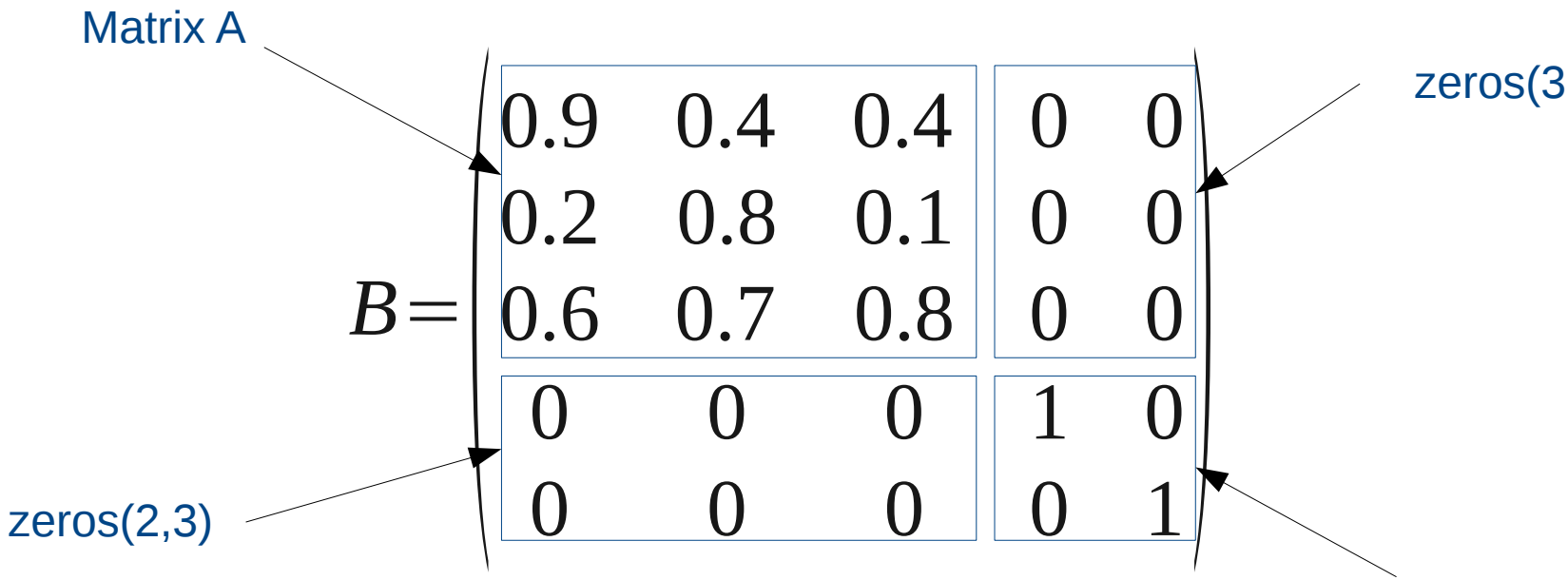
zusammengesetzten Matrizen

- Beispiel: A ist eine 3x3-Matrix
- A = [0.9 0.4 0.4;0.2 0.8 0.1;0.6 0.7 0.8]

$$A = \begin{pmatrix} 0.9 & 0.4 & 0.4 \\ 0.2 & 0.8 & 0.1 \\ 0.6 & 0.7 & 0.8 \end{pmatrix}$$

zusammengesetzten Matrizen

- Beispiel-Fortsetzung
- B = [A,zeros(3,2);zeros(2,3),eye(2)]



B ist eine 5x5 Matrix

Zugriff auf Matrix-Elemente

$$A = \begin{pmatrix} 0.9 & 0.4 & 0.4 \\ 0.2 & 0.8 & 0.1 \\ 0.6 & 0.7 & 0.8 \end{pmatrix}$$

- A(m,n)
  - m … Zeile
  - n … Spalte
- Beispiel
  - A(2,3) = 0.1
  - A(2,2:3) = [0.8 0.1]
  - A(:,2:3) = [0.4 0.4; 0.8 0.1; 0.7 0.8]

Rechenoperationen

- Allgemein

  \+ ... Addition

  \- ... Subtraktion

  * ... Multiplikation

  / … Division

  ^ ... Exponent

Rechenoperationen

- Matrizen & Vektoren

  .* … Elementweise Multiplikation

  ./ … Elementweise Division

  \ ... Division („left matrix divide“ → Lösung von Gleichungssystemen)

  ‘ ... Transponierte (AT)

  .^ … Elementweise Potenzen

Vergleichsoperatoren

< ... kleiner als

> ... größer als

<= ... kleiner oder gleich als

>= ... größer oder gleich als

== ... gleich

~= ... nicht gleich, ungleich

Beachte

- “=“ zur Zuweisung
- “==“ zum Vergleichen

Vergleichsoperatoren

- Vergleich wahr → Antwort = 1
- Vergleich nicht wahr → Antwort = 0
- Anwendung bei Skalaren
  - Beispiele
    - 2<7 → ans = 1
    - 2>7 → ans = 0
    - 2==7 → ans = 0
    - 2==2 → ans = 1

Vergleichsoperatoren

- Anwendung bei Matrizen: Vergleich der einzelnen Elemente zweier Matrizen
- Beispiel: A=rand(4), B=triu(A)

$$A = \begin{pmatrix} 0.2 & 0.2 & 0.7 & 0.6 \\ 0.1 & 0.1 & 0.4 & 0.4 \\ 0.6 & 0.7 & 0.9 & 0.3 \\ 0.5 & 0.1 & 0.5 & 0.2 \end{pmatrix} \qquad B = \begin{pmatrix} 0.2 & 0.2 & 0.7 & 0.6 \\ 0 & 0.1 & 0.4 & 0.4 \\ 0 & 0 & 0.9 & 0.3 \\ 0 & 0 & 0 & 0.2 \end{pmatrix}$$

B = obere Dreiecksmatrix von A

Vergleichsoperatoren

- Beispiel-Fortsetzung: A==B

$$A = \begin{pmatrix} 0.2 & 0.2 & 0.7 & 0.6 \\ 0.1 & 0.1 & 0.4 & 0.4 \\ 0.6 & 0.7 & 0.9 & 0.3 \\ 0.5 & 0.1 & 0.5 & 0.2 \end{pmatrix} \qquad B = \begin{pmatrix} 0.2 & 0.2 & 0.7 & 0.6 \\ 0 & 0.1 & 0.4 & 0.4 \\ 0 & 0 & 0.9 & 0.3 \\ 0 & 0 & 0 & 0.2 \end{pmatrix}$$

$$ans = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Funktionen

$$[A, b] = functionname(x, y, ...)$$

- Variable Parameteranzahl
- Liste als Ergebnis
  - Bei Zuweisung auf eine Variable, wird das erste Element zugewiesen.
- Eingebaute Hilfe
  - help <Funktionsname>

Skalarfunktionen

sin(a) ... Sinus

cos(a) ... Cosinus

tan(a) ... Tangens

sqrt(a) ... Wurzel

...

- Beispiel
  - sqrt(9) → ans = 3

Skalarfunktionen

- manche Funktionen auch anwendbar auf Matrizen
- diese Funktion wird dann Elementweise ausgeführt
- Beispiel
  - A=rand(2), sqrt(A)

Vektorfunktionen

- Bestimmte Funktionen hauptsächlich für Vektoren geeignet
- Beispiele

  max(v) ... Maximum

  min(v) ... Minimum

  sum(v) ... Summe

  ...

Vektorfunktionen

- auch auf Matrizen anwendbar
- Matrix wird Spaltenweise durchgearbeitet
- Ergebnisse werden in einem Zeilenvektor dargestellt

Vektorfunktionen

- Unterschied von max(A) und max(max(A))

$$A = \begin{pmatrix} 2 & 8 \\ 1 & 9 \end{pmatrix}$$

$$max(A) = \begin{pmatrix} 2 & 9 \end{pmatrix}$$

ein Zeilenvektor mit den größten Elementen in den Spalten wird ausgegeben

$$max(max(A)) = 9$$

das größte Element wird ausgegeben (Skalar)

Matrixfunktionen

size(A) ... Größe

eig(A) ... Eigenwerte und Eigenvektoren

inv(A) ... Inverse

rank(A) ... Rang

diag(A) … Matrixdiagonale

tril(A) … extrahiert die untere Dreiecks-Matrix

triu(A) … extrahiert die obere Dreiecks-Matrix

...

Matrixfunktionen

- Beispiel
  - eig(A) erstellt einen Spaltenvektor mit den Eigenwerten von A
  - A=rand(3)

$$A = \begin{pmatrix} 0.2897 & 0.7271 & 0.5681 \\ 0.3412 & 0.3093 & 0.3704 \\ 0.5341 & 0.8385 & 0.7027 \end{pmatrix} \rightarrow eig(A) = \begin{pmatrix} 1.5354 \\ -0.1967 \\ -0.0370 \end{pmatrix}$$

Grafiken

- MATLAB kann
  - 2-D oder 3-D Grafen von Kurven
  - 3-D Netz-Flächengrafiken
  - farbige 3-D Flächengrafiken
  - usw.
- erstellen

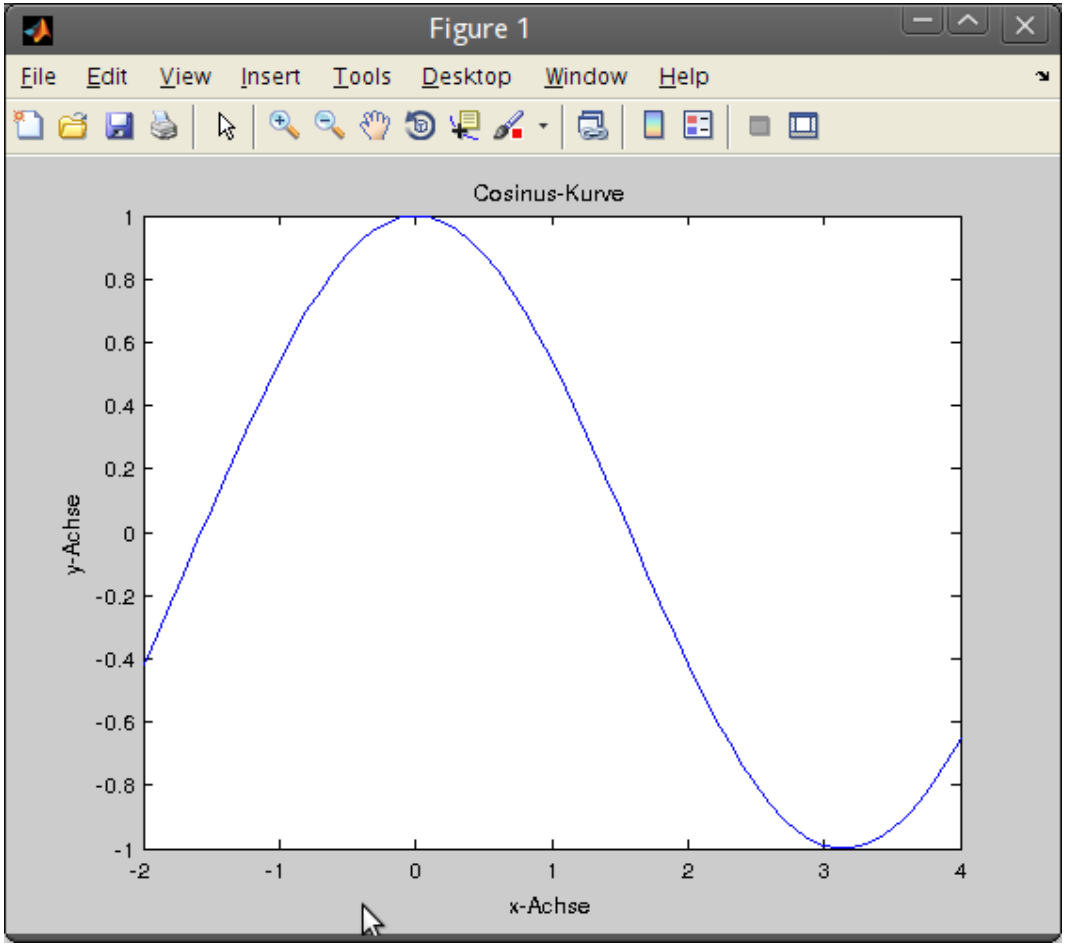
Grafiken 2D

- plot(x,y) erstellt eine lineare xy-Graphik
- x und y sind Vektoren gleicher Länge
- Befehl figure öffnet ein zweites Graphikfenster
- Graphiken können beschriftet werden
  - title(‘Cosinus-Kurve‘)
  - xlabel(‘x-Achse‘), ylabel('y-Achse')

Graphiken 2D

- Beispiel

```
x = -2 : 0.1 : 4
y = cos(x)
plot(x,y)
title('Cosinus-Kurve')
ylabel('y-Achse')
xlabel('x-Achse')
```



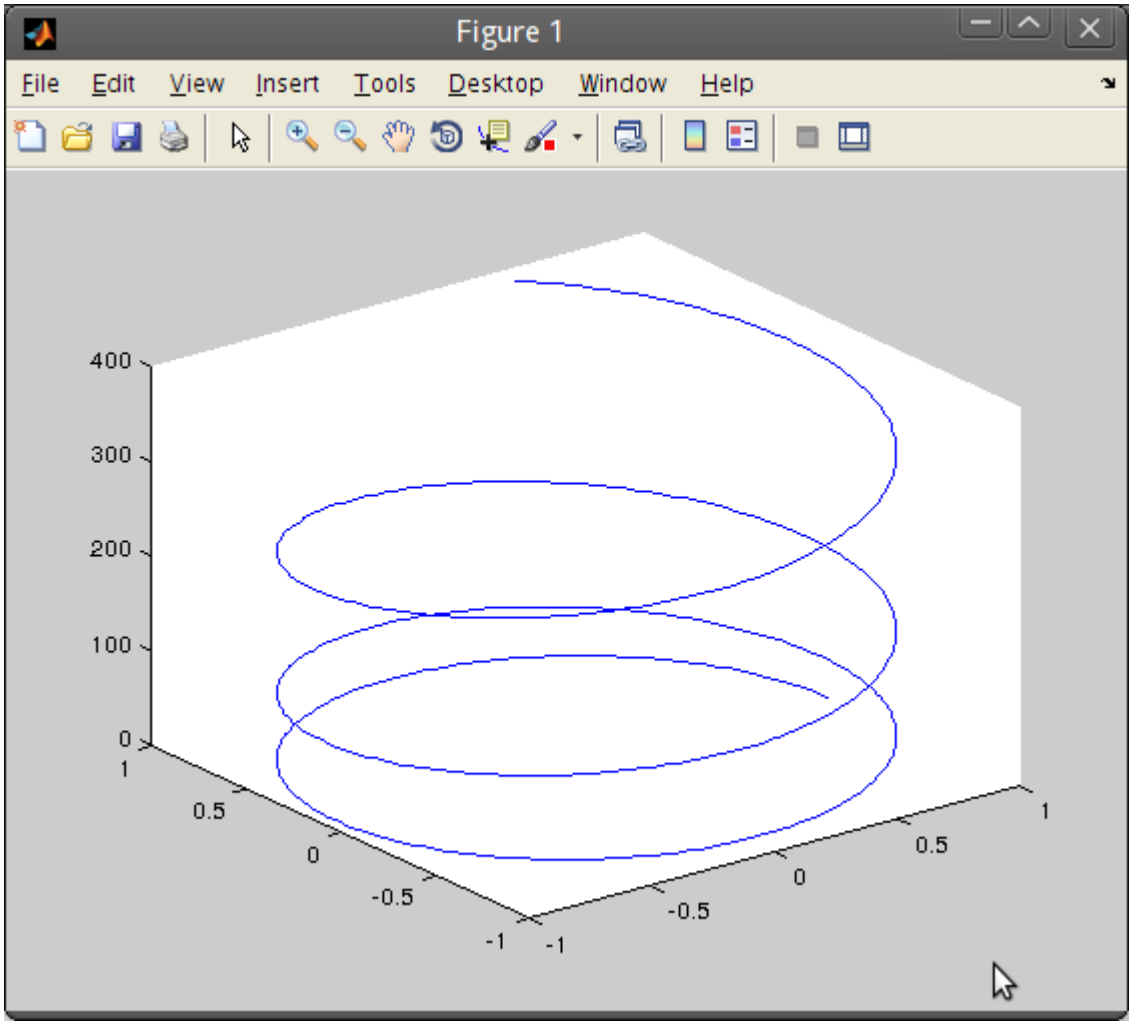
Grafiken 3D

- plot3(x,y,z) erstellt Kurven in einem 3-dimensionalen Raum
- x, y und z sind Vektoren gleicher Länge
- Beschriftung der Graphiken wie bei 2-dimensionalen Darstellungen

Grafiken 3D

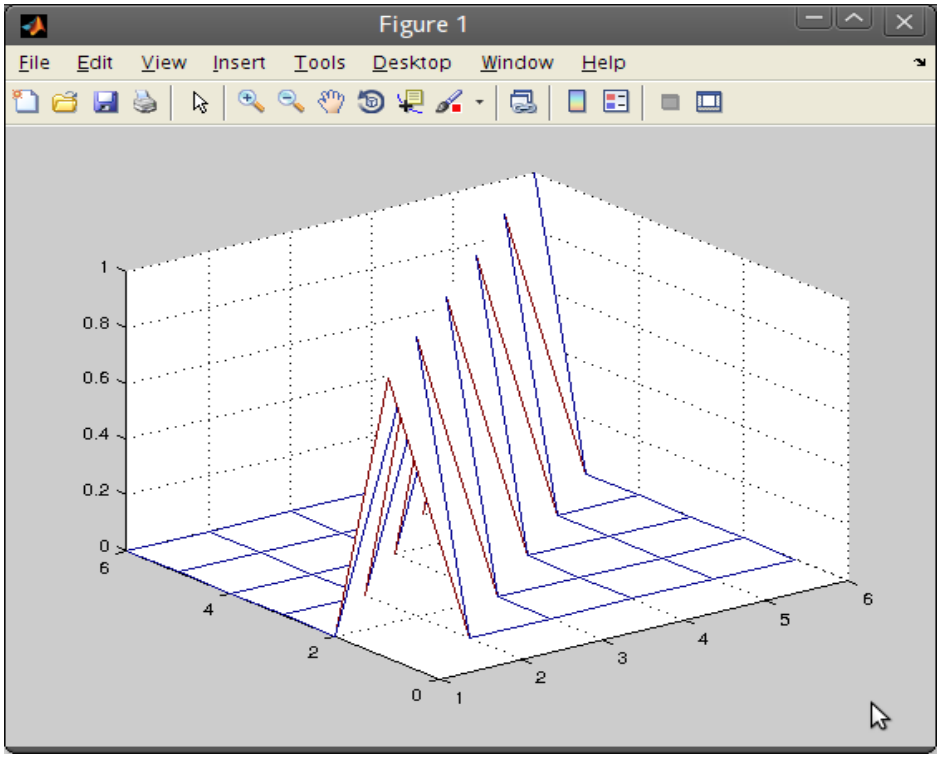
- Beispiel

```
t = 0.02 : 0.03 : 20
x = cos(t)
y = sin(t)
z = t.^2
plot3(x,y,z)
```



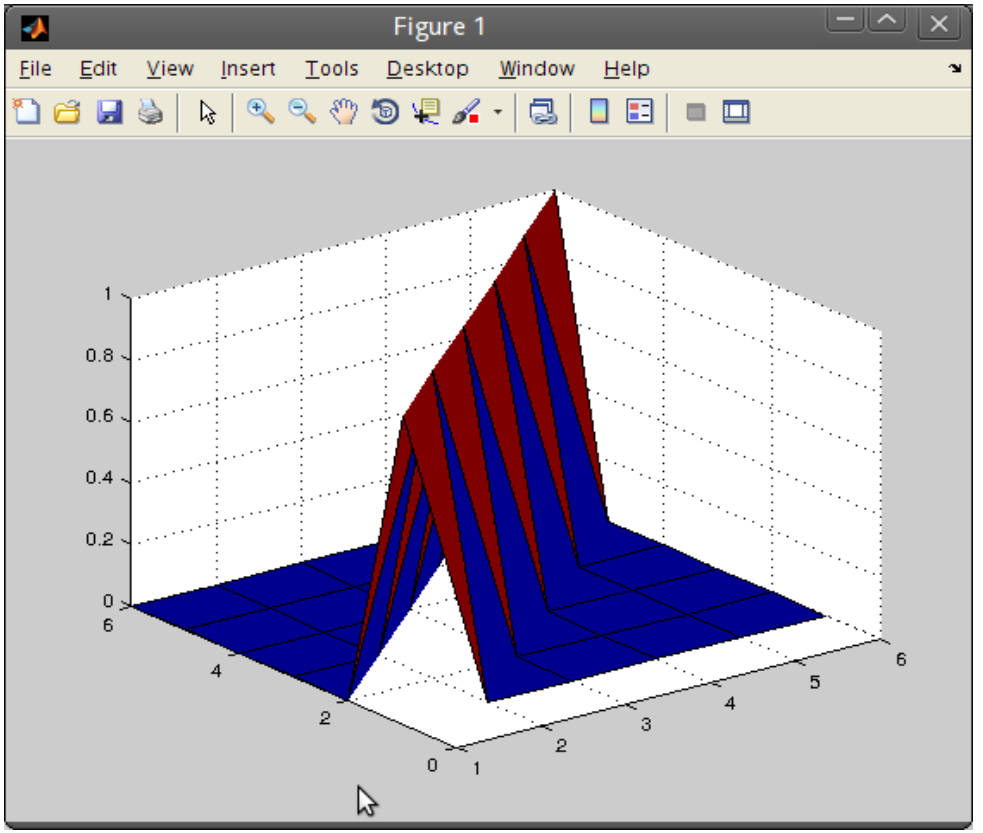
Grafiken 3D-Netzflächen

- mesh(z) erstellt eine 3-dimensionale Ansicht der Elemente der Matrix z
- Beispiel: z=eye(6), mesh(z)



Grafiken farbliche 3D-Flächen

- surf(z) erstellt eine farbliche 3D-Ansicht der Elemente der Matrix z
- Beispiel: surf(z)



MATLAB Skripte

- *.m (M-Dateien)
- Skriptdatei = Sequenz von „normalen“ MATLAB-Befehlen
  - z.B. test.m, Eingabe von test führt alle Befehle dieser Datei aus
- Variablen in Skriptdateien sind global
  - Werte von gleichen Variablen in der aktuellen MATLAB-Sitzung werden geändert

MATLAB Funktionsdatei

- Funktionsdatei = neu erstellte problem-spezifische Funktion
  - hat gleichen Stellenwert wie die üblichen Funktionen von MATLAB
  - Variablen sind lokal, können aber als global deklariert werden
- Aufbau (untitled.m)

```
function [ output_args ] = untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here



end
```

MATLAB Funktionsdatei

- Eigene Skripte und Funktionen nur aus dem aktuellen Arbeitsverzeichnis verfügbar

  cd <Verzeichnis> … Verzeichniswechsel

  edit <Funktionsname> … öffnet die Funktionsdatei im Editor (auch für eingebaute Funktionen)
- Aufruf wie eingebaute Funktion
- help <Funktionsname> … gibt selbst definierte Hilfe-Meldung aus
  - Beispiel

```
UNTITLED2 Summary of this function goes here
    Detailed explanation goes here
```

Schleifen

- For-Schleife
  - beginnt mit Ausdruck for und endet mit end
- Beispiel

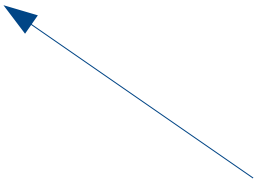
```
x=[ ]
for i=1:n
    x=[x, i^2]
end
```

Weist pro Iteration der Variable i den nächsten Wert aus der Liste zu

Schleifen

- While-Schleife
  - beginnt mit Ausdruck while <vergleich> und endet mit end
- Beispiel

```
n=0; a = 10000;
while 2^n < a
    n=n+1;
end
```



solange der Vergleich wahr ist, werden die Anweisungen in der Schleife ausgeführt

If-Anweisung

- beginnt mit Ausdruck if <vergleich> und endet mit end
- Beispiel

```
if n < 0
    do something ...
elseif n > 1
    do something else …
else
    do something different ...
end
```

Ausgabeunterdrückung

- ; … Semikolon am Ende der Befehlszeile unterdrückt die Ausgabe

Zeichenketten (String)

- Zeichenketten sind in MATLAB mit einfachen Anführungszeichen umgeben
- Beispiel
  - s = ‘Das soll eine Zeichenkette sein‘
    - der eingegebene Text wird der Variablen s zugeordnet
- mit dem Befehl disp() kann der String am Bildschirm ausgegeben werden

Fehlermeldungen

- die Funktion error() gibt eine Fehlermeldung aus
- Beispiel

  error(‘Das ist eine Fehlermeldung‘)
- Ausgabe in MATLAB:

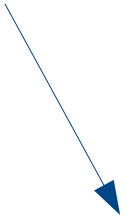
  ??? Das ist eine Fehlermeldung

Dateneingabe

input() … fordert den User auf Daten einzugeben

- Beispiel

```
daten = input('Geben Sie die Groesse ein: ')
```



nach Betätigen der [Enter]-Taste erhält die Variable daten den eingegeben Wert

- Strings müssen mit einfachen Anführungszeichen (') eingegeben werden

Zeitmessung

- Laufzeit (elapsed time) erhält man durch die Stoppuhr-Zeiten
  - tic (startet den Timer)
  - toc (gibt die Laufzeit zurück)
- Beispiel

```
A=rand(3,3)
b=rand(1,3)
tic; x=A/b; toc
„Elapsed time is 0.175977 seconds.“
```

Darstellungsformat (Output-Format)

- Folgende Befehle können die Ausgabe am Bildschirm kontrollieren
  - format short ... Fixpunkt mit 4 Dezimalstellen
  - format long ... Fixpunkt mit 14 Dezimalstellen
  - format hex ... Hexadezimal
  - ...

Hardcopy

- diary <dateiname>
- Beispiel

```
diary test.m
c=rand(2,1)
d= rand(3,1)
diary off
```

in test.m werden alle Befehle mit ihren Ergebnissen eingetragen

diary off beendet diesen Vorgang

nach Beendigung kann die Datei test.m geändert und gedruckt werden

Grafiken Hardcopy

- Befehl print druckt aktuelle Graphik aus
- print <dateiname> speichert die aktuelle Graphik unter dem gewünschten Dateinamen
- existiert Dateiname, wird Datei überschrieben
- außer wenn –append
  - print –append <dateiname> fügt aktuelle Graphik in der Datei dateiname hinzu

Grafiken Hardcopy

- keine Angabe der Extension beim Dateinamen → Standard-Extension wird angehängt
  - z.B. wenn die Default-Einstellung PostSript verwendet → dateiname.ps
- Standard-Einstellungen können geändert werden
  - Beispiel: print –deps –f1 graphik1 speichert die Graphik f1 in die Datei graphik1.eps

MATLAB Erweiterungen

- Erweiterungen werden in „Toolboxes“ zusammengefasst
- Beispiele
  - Image Processing Toolbox
  - Symbolic Math Toolbox
  - Statistics Toolbox
  - Neural Networks Toolbox
  - Parallel Computing Toolbox
  - usw.

MATLAB Erweiterungen

- Eigene Erweiterungen können in verschiedenen Programmiersprachen geschrieben werden (MEX)
- MATLAB kann in eigene Programme eingebunden werden (Engine Library)

MATLAB Bezugsquellen

- TU-Wien
  - Mitbelegen
  - Kosten: 13,90€ für ein Jahr
- Studenten-Labor Lenaugasse
  - Account-Formular ausfüllen

MATLAB Alternativen

- andere frei verfügbare Systeme
- i.A. geringere Funktionalität als MATLAB
- Vorsicht, nicht immer kompatibel!
- GNU Octave
  - Link: www.gnu.org/software/octave
- Scilab
  - Link: www.scilab.org