

INTELLIGENTE DATENANALYSE IN MATLAB

Einführung in MATLAB

Überblick

- Was ist MATLAB?
 - ▣ Abkürzung für MATrix LABoratory.
 - ▣ Numerikprogramm für das Rechnen mit großen Zahlenfeldern (Matrizen und Vektoren).
 - ▣ Interpretersprache.
 - ▣ Werkzeug zur Erzeugung von graphischen Darstellungen und Oberflächen.
- Was ist MATLAB nicht?
 - ▣ MATLAB ist (eigentlich) kein Arithmetikprogramm für symbolisches Rechnen.

Überblick

□ Warum MATLAB?

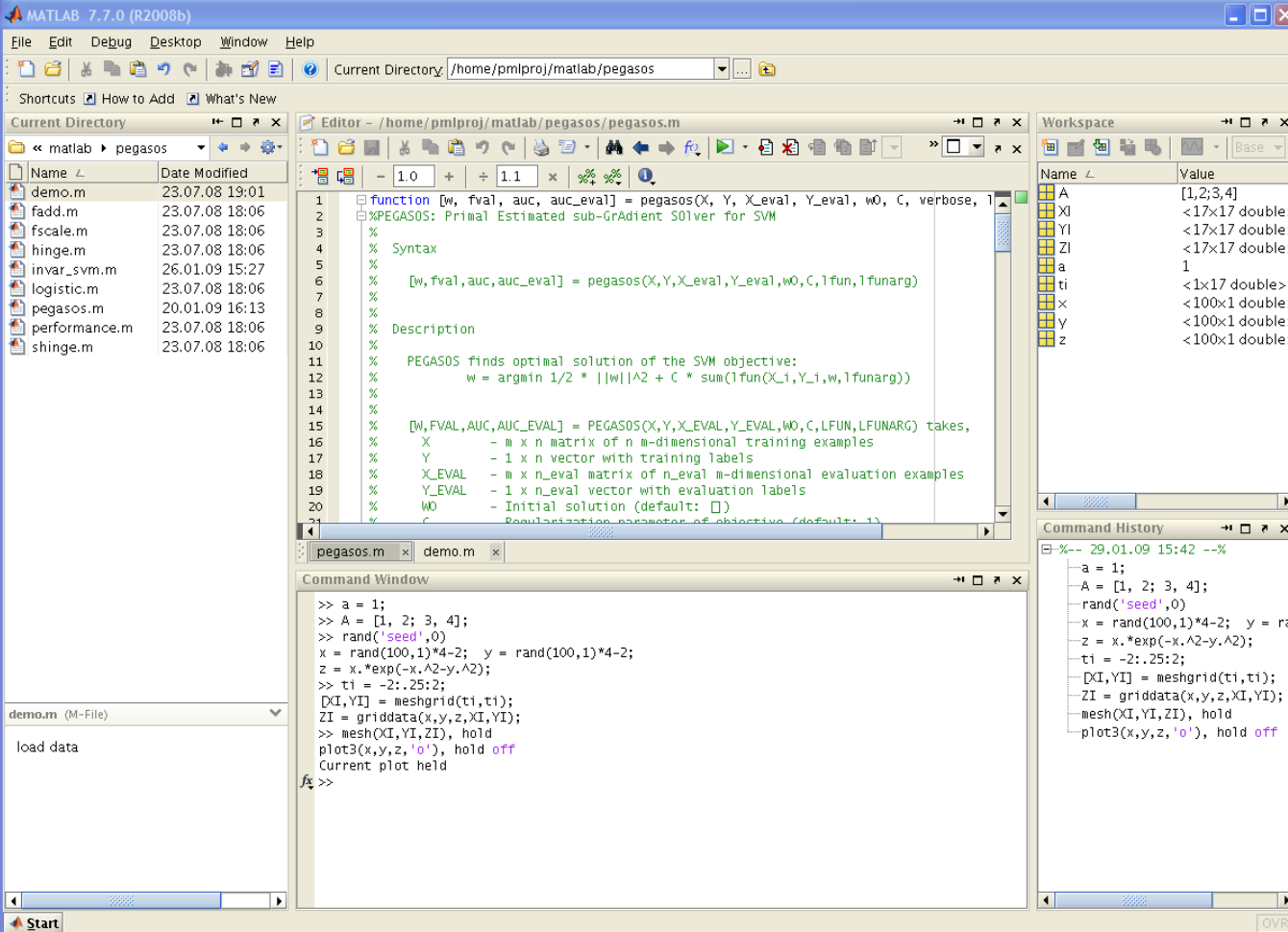
- Einfache Syntax basierend auf dem Matrix-Datentyp.
- Breites Spektrum mathematischer Funktionen und Algorithmen aus verschiedenen Anwendungsbereichen.
- Plattformübergreifende Programmiersprache.
- Einfach zu bedienende Visualisierungsmöglichkeiten.
- Demos: <http://www.mathworks.de/products/matlab/demos.html>

□ Alternativen?

- Octave, Scilab
- R

Das Programm

Arbeitsfenster



The image shows the MATLAB 7.7.0 (R2008b) interface. The main window is the Editor, displaying the file `pegasos.m`. The workspace on the right shows variables `A`, `XI`, `YI`, `ZI`, `a`, `ti`, `x`, `y`, and `z`. The command window at the bottom shows the execution of the `demo.m` script, which initializes variables and calls the `pegasos` function.

Editor - /home/pmlproj/matlab/pegasos/pegasos.m

```

1 function [w, fval, auc, auc_eval] = pegasos(X, Y, X_eval, Y_eval, w0, C, verbose, 1)
2 %PEGASOS: Primal Estimated sub-Gradient Solver for SVM
3 %
4 % Syntax
5 %
6 % [w,fval, auc, auc_eval] = pegasos(X,Y,X_eval,Y_eval,w0,C,1fun,1funarg)
7 %
8 %
9 % Description
10 %
11 % PEGASOS finds optimal solution of the SVM objective:
12 %     w = argmin 1/2 * ||w||^2 + C * sum(1fun(X_i,Y_i,w,1funarg))
13 %
14 %
15 % [W,FVAL,AUC,AUC_EVAL] = PEGASOS(X,Y,X_EVAL,Y_EVAL,W0,C,LFUN,LFUNARG) takes,
16 % X - m x n matrix of n m-dimensional training examples
17 % Y - 1 x n vector with training labels
18 % X_EVAL - m x n_eval matrix of n_eval m-dimensional evaluation examples
19 % Y_EVAL - 1 x n_eval vector with evaluation labels
20 % W0 - Initial solution (default: [])
21 % C - Regularization parameter of objective (default: 1)

```

Workspace

Name	Value
A	[1,2,3,4]
XI	<17x17 double>
YI	<17x17 double>
ZI	<17x17 double>
a	1
ti	<1x17 double>
x	<100x1 double>
y	<100x1 double>
z	<100x1 double>

Command History

```

-- 29.01.09 15:42 --%
>> a = 1;
>> A = [1, 2; 3, 4];
>> rand('seed',0)
>> x = rand(100,1)*4-2; y = rand(100,1)*4-2;
>> z = x.*exp(-x.^2-y.^2);
>> ti = -2:.25:2;
>> [XI,YI] = meshgrid(ti,ti);
>> ZI = griddata(x,y,z,XI,YI);
>> mesh(XI,YI,ZI), hold
>> plot3(x,y,z,'o'), hold off

```

Command Window

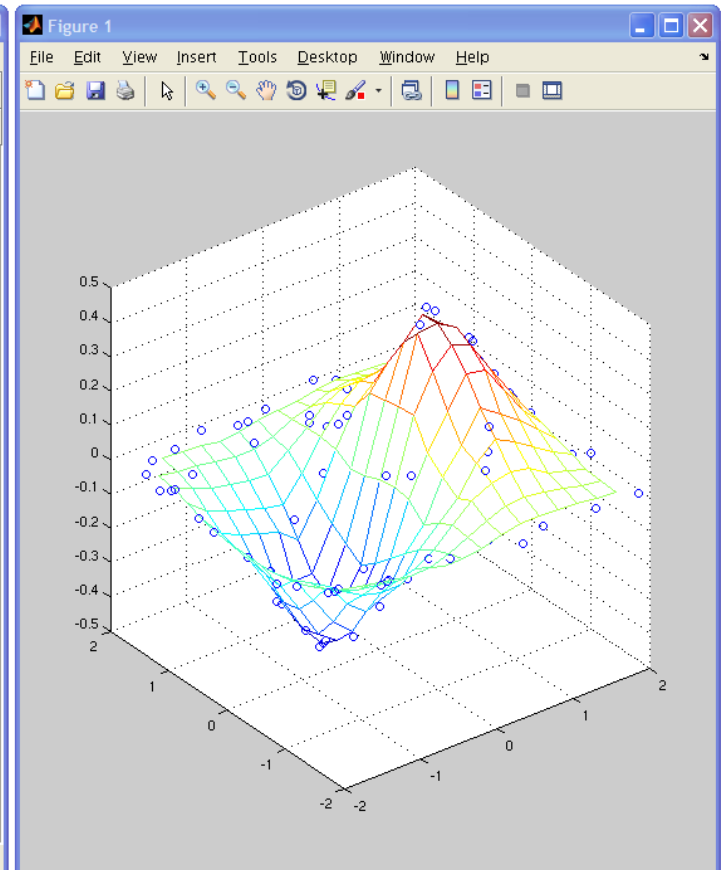
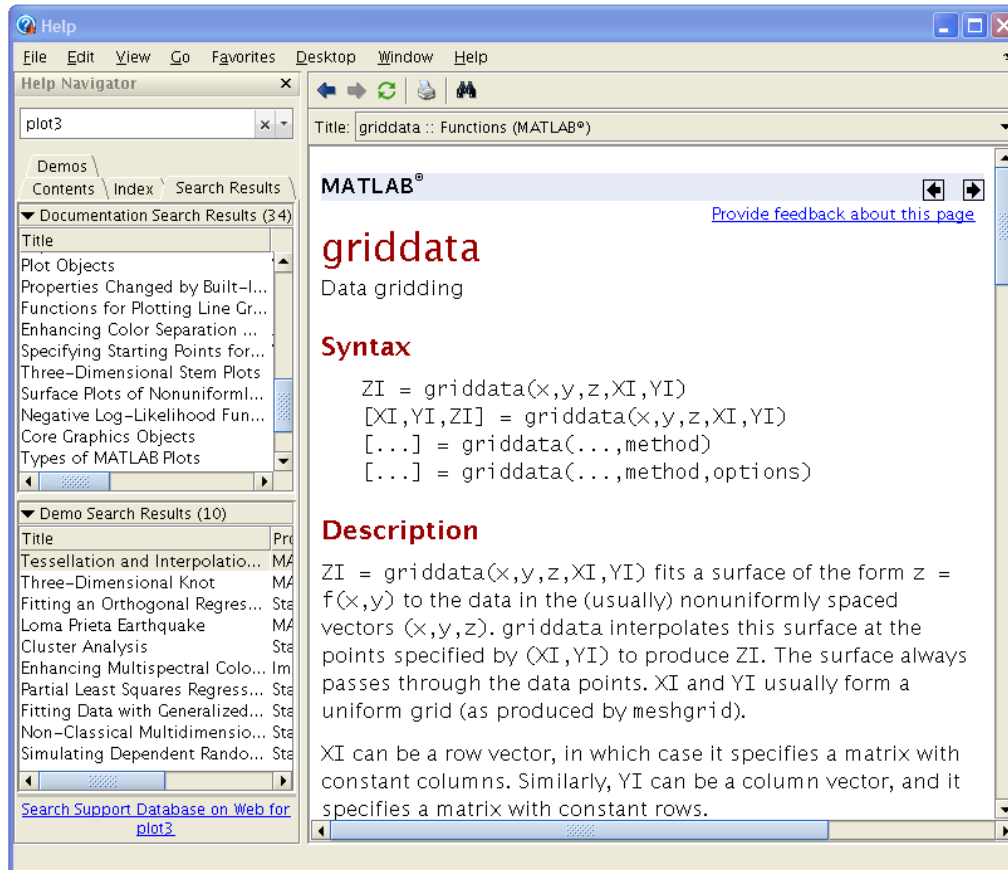
```

>> demo
>> a = 1;
>> A = [1, 2; 3, 4];
>> rand('seed',0)
>> x = rand(100,1)*4-2; y = rand(100,1)*4-2;
>> z = x.*exp(-x.^2-y.^2);
>> ti = -2:.25:2;
>> [XI,YI] = meshgrid(ti,ti);
>> ZI = griddata(x,y,z,XI,YI);
>> mesh(XI,YI,ZI), hold
>> plot3(x,y,z,'o'), hold off
>>

```

Das Programm

Hilfe und Plot-Fenster



Das Programm

Hilfe



□ Folgende Befehle:

`help`

Listet alle Hilfe-Themen auf

`help topic`

Hilfe zu einem speziellen Thema

`help command`

Hilfe zu einem speziellen Befehl

`Helpwin`

Öffnet MATLAB-Hilfefenster

`Lookfor keyword`

Sucht in allen MATLAB-Dateien nach
einem speziellen Stichwort

□ Online-Hilfe.

Das Programm

Befehle in MATLAB



- Skriptsprache: Eingabe erfolgt durch einen ASCII-Text (.m-File) oder über die Tastatur.
- Skripte mit Parameterübergabe heißen Funktionen.
- Befehl wird nach dem Enter-Zeichen sofort ausgeführt.
- Durch Befehl wird gleichnamiges .m-File gestartet welches wiederum eine Folge von Befehlen enthält.
 - ▣ Jedes selbstgeschriebene Programm erweitert MATLAB um einen neuen Befehl.
- Grundlegende Funktionen sind eingebaut.

Das Programm

Datenstruktur in MATLAB



- Standard-Variablen sind mehrdimensionale Felder (Skalare, Vektoren, Matrizen usw.).
- Zahlen sind i.d.R. Fließkommazahlen doppelter Präzision, Ausnahmen:
 - Strings sind 1 x N-Felder vom Typ Character (ASCII-Zeichen).
 - In der Bildverarbeitung werden oft Felder von vorzeichenlosen 8-Bit-Ganzzahlen verwendet.
 - Wahr/Falsch-Felder als Ergebnis logischer Operationen.
- Dimensionierung zu keiner Zeit erforderlich, aber empfehlenswert.

Das Programm

Datenstruktur in MATLAB



- Cell-Array: Feld mit beliebigem Inhalt z.B. Skalar, Matrix, String, Cell-Array usw.
- Sparse Matrix: Verhält sich beim Rechnen wie normale Felder, benötigt bei dünn besetzten Feldern deutlich weniger Speicherplatz.
- Strukturen: Ähnlich Struct-Element in anderen Programmiersprachen.

Das Programm

Arbeitsbereich

- Alle unter dem Prompt „>>“ oder in Skripten erzeugten Variablen liegen im „Arbeitsbereich“.
- Funktionen haben ihren eigenen, lokalen Arbeitsbereich.
- Abfrage der im Arbeitsbereich befindlichen Variablen:

```
>> who
```

```
Your variables are:
```

```
a      b      b2     c
```

- Detailliertere Aufstellung: `whos`
- Löschen des Arbeitsbereiches: `clear`

Daten-/Befehlseingabe

Skalare

□ Eingabe von Skalaren:

```
>> a=2
```

```
a=
```

```
2
```

```
>> a=sqrt(-16)
```

```
a=
```

```
0 + 4.0000i
```

□ Ein Semikolon (;) unterdrückt die Ausgabe des Ergebnisses:

```
>> a=1.2345 * 2;
```

□ Nachträgliche Abfrage:

```
>> a
```

```
a=
```

```
2.4690
```

Daten-/Befehlseingabe

Vektoren

□ Eingabe von Vektoren:

```
>> b=[2,4,6,8]
```

```
b=
```

2 4 6 8

Dies ist ein 1x4-Vektor, also ein Zeilenvektor.

□ Im Unterschied dazu ist

```
>> b2=[2;4;6;8]
```

```
b2=
```

2
4
6
8

ein 4x1-Vektor, also ein Spaltenvektor.

Daten-/Befehlseingabe

Vektoren



- Doppelpunkt erzeugt Zeilenvektoren:

```
>> b=2:2:8
```

```
b=
```

```
      2      4      6      8
```

Parameter sind Anfang, Schrittweite und Ende der Folge.

- Parameter sind Anfang, Ende und Anzahl:

```
>> b2=linspace(1,3,5)
```

```
b2 =
```

```
      1.00      1.50      2.00      2.50      3.00
```

Daten-/Befehlseingabe

Matrizen

□ Eingabe von Matrizen:

```
>> A=[1 2 3; 4 5 6; 7 8 0]
```

```
A=
```

1	2	3
4	5	6
7	8	0

Dies ist ein 3x3-Matrix.

□ Transposition erfolgt mit Apostroph:

```
>> A2 =[1 2 3; 4 5 6; 7 8 0]'
```

```
A2=
```

1	4	7
2	5	8
3	6	0

Daten-/Befehlseingabe

Laden und Speichern

<code>save</code>	Speichert gesamten Arbeitsbereich unter <code>matlab.mat</code> im aktuellen Verzeichnis.
<code>save fname</code>	Speichert gesamten Arbeitsbereich unter <code>fname.mat</code> im aktuellen Verzeichnis.
<code>save fname A b</code>	Speichert nur die Variablen <code>A</code> und <code>b</code> unter <code>fname.mat</code> im aktuellen Verzeichnis.
<code>save A.xyz A -ascii</code>	Speichert Variable <code>A</code> unter <code>A.xyz</code> als ASCII-Tab.
<code>load</code>	Lädt alle Variablen aus <code>matlab.mat</code> in den Arbeitsbereich.
<code>load fname</code>	Lädt alle Variablen aus <code>fname.mat</code> in den Arbeitsbereich.
<code>load A.xyz</code>	Lädt die ASCII-Tabelle <code>A.xyz</code> in die Variable <code>A</code> .

Variablen

Indizierung



A=1	2	3
4	5	6
7	8	0

□ Lineare Indizierung:

>> A(6) liefert die 8: spaltenweise Zählung

□ Indizierung über Zeile und Spalte:

>> A(2, 3) liefert die 6: Name(Zeile, Spalte)

□ Indizierung mit Vektoren:

>> A([1, 3], 2) Index als Vektor

>> A(2, :) liefert die gesamte 2. Zeile

>> A(:, 3) liefert die gesamte 3. Spalte

Variablen

Manipulation von Matrizen

- Änderung von Werten erfolgt durch Zuweisung:

```
>> A(4,4)=28
```

```
A =
```

1	2	3	0
4	5	6	0
7	8	0	0
0	0	0	28

- Matrixgröße wird zur Laufzeit automatisch angepasst.
- Informationen über Matrizen (Variablen):

<code>who, whos</code>	Welche Variablen gibt es? Größe?
<code>size(A)</code>	Dimensionen einer Matrix
<code>length(v)</code>	Länge eines Vektors
<code>numel(v)</code>	Anzahl der Elemente eines Vektors

Variablen

Matrizen automatisch erzeugen

□ Weitere Befehle zum Erzeugen spezieller Matrizen:

<code>zeros</code>	Matrix gefüllt mit Nullen
<code>ones</code>	Matrix gefüllt mit Einsen
<code>eye</code>	Einheitsmatrix
<code>rand</code>	Matrix mit zufälligen Elementen (gleichverteilt in $[0,1]$)
<code>randn</code>	Matrix mit zufälligen Elementen (normalverteilt)

□ Beispiel:

```
>> x=rand(1,4)
>> plot(x)
```

Variablen

Besondere Variable



- Variablen welche beim Start von MATLAB vorhanden sind (können nachträglich überschrieben werden):

<code>pi</code>	3.14159...
<code>i, j</code>	imaginäre Einheit
<code>inf</code>	unendlich
<code>NaN</code>	“not a number“
<code>eps</code>	Fließkomma-Rechengenauigkeit (IEEE: 2^{-52})
<code>realmax</code>	größte Fließkommazahl (IEEE: $2^{1024} - 1$)
<code>realmin</code>	kleinste Fließkommazahl (IEEE: 2^{-1022})
<code>computer</code>	Computertyp und Betriebssystem

Operationen

Arithmetische Ausdrücke



□ Verknüpfungszeichen für Matrizen:

+	Addition
-	Subtraktion
*	Matrix-Multiplikation
^	Matrix-Potenzierung
/	Matrix-Division
\	Matrix-Linksdivision
\	komplex-konjugierte Transposition

□ Elementweise Verknüpfungen:

. *	elementweise Multiplikation
. ^	elementweise Potenzierung
. /	elementweise Division

Operationen

Beispiele

Eingabe von Werten

```
>> clear
>> x=[-1 0 2]'
```

x=	-1	0	2
----	----	---	---

```
>> A=[1 2 3;4 5 6;7 8 0]
```

A=	1	2	3
	4	5	6
	7	8	0

```
>> c=3;
>>
```

Matrixoperationen

```
>> B=A'
B=
```

1	4	7
2	5	8
3	6	0

```
>> C=A+B
C=
```

2	6	10
6	10	14
10	14	0

```
>> D=A*B
D=
```

14	32	23
32	77	68
23	68	113

```
>>
```

Operationen

Beispiele

Vektoroperationen I

```
>> x
x=
    -1
     0
     2

>> y=x-1
y=
    -2
    -1
     1

>> x'*y
ans=
     4

>>
```

Vektoroperationen II

```
>> x*y'
ans=
     2     1    -1
     0     0     0
    -4    -2     2

>> y*x'
ans=
     2     0    -4
     1     0    -2
    -1     0     2

>> pi * x
ans=
   -3.1416
         0
    6.2832
```

Operationen

Matrix-„Division“



□ „Linksdivision“ in MATLAB:

$$X = A \setminus B \quad \text{bedeutet} \quad X = \text{inv}(A) * B$$

Hier ist X die Lösung von $A * X = B$.

- Linksdivision definiert, wenn A genauso viele Zeilen hat wie B .
- Falls die Matrix A quadratisch ist, wird sie nach dem Gauß'schen Eliminationsverfahren berechnet.
- Hat A mehr oder weniger Spalten als B , dann ist X die Lösung des über- bzw. unterbestimmten Gleichungssystems $A * X = B$ im Sinne kleinster Quadrate.

□ „Rechtsdivision“ in MATLAB:

$$X = A / B \quad \text{ist definiert durch} \quad A/B = (B' \setminus A')'$$

Hier ist X die Lösung von $X * A = B$.

Operationen

Beispiel



Matrix-„Division“

```
>> b=A*x
b=
     5
     8
    -7

>> z = x
>> x=A\b
x=
    -1
     0
     2

>> all(z==x)
ans=
     1
```

□ Lösen des folgenden Gleichungssystems:

$$1 x_1 + 2 x_2 + 3 x_3 = 5$$

$$4 x_1 + 5 x_2 + 6 x_3 = 8$$

$$7 x_1 + 8 x_2 + 0 x_3 = -7$$

A=1	2	3	b= 5	x=-1
4	5	6	8	0
7	8	0	-7	2

Operationen

Logische Ausdrücke



□ Vergleichsoperatoren:

==	gleich
~=	nicht gleich
>	größer
>=	größer-gleich
usw.	

□ Logische Verknüpfungen:

&	logisches UND
	logisches ODER
~	logisches NICHT
xor	logisches EXKLUSIV-ODER

Operationen

Beispiel

Boolesche Variablen

```
>> L = (A >= 5)
```

```
L =
```

0	0	0
0	1	1
1	1	0

```
>> B = A(L)
```

```
B =
```

7
5
8
6

- Ergebnisse von Logik-Operationen werden in 0/1-Matrizen gespeichert.
- Boole-Matrix kann zur Indizierung verwendet werden.

Graphische Darstellung

Überlick



- MATLAB verfügt über umfangreiche Funktionen zum Erzeugen von graphischen Darstellung:
 - Einfache Darstellung von Messreihen in kartesischen Koordinaten.
 - Polarplots.
 - 3-dimensionale, farbcodierte Pixelbilder.
 - 3D-Flächen mit Höhenlinien.
 - Beleuchtete 3D-Flächen mit Schatten und Glanzlichtern.

Graphische Darstellung

Linien und Flächen

- Funktion `plot()` stellt Werte in der xy-Ebene dar.

Format: `plot(x, y, 'symbols')`

- Funktion `plot3()` stellt Werte im xyz-Raum dar.

Format: `plot3(x, y, z, 'symbols')`

- String `'symbols'` gibt Farbe, Punkt-/Linienform und -typ an.

- Zahlreiche weitere Plot-Befehle:

`line, axis, view, mesh, surf, surfc`

Programmieren in MATLAB

Schleifen & Bedingungen

□ If/else-Anweisung:

```
if condition
    statements
else
    statements
end
```

□ While-Schleife:

```
while condition
    statements
end
```

□ „condition“ ist ein Boolscher Ausdruck.

□ For-Schleife:

```
for counter=first:last
    statements
end
```

□ „break“ zum vorzeitigen verlassen einer For- bzw. While-Schleife.

□ „continue“ zum Sprung in die nächste Iteration.

Programmieren in MATLAB

Funktionen

- **Dateiname** `testfunktion.m`
- **Dateikopf** `function [A, B] = testfunktion(C,D,E)`
`A = ...`
`B = ...`
- **Aufruf der Funktion im eigentlichen Programm durch**
`[M, N] = testfunktion (A,B,C);`
- **Aufgaben können in Unterprogramme aufgeteilt werden.**
- **Jede globale Funktion in einer eigenen Datei.**
- **Definition von lokalen Funktionen innerhalb einer globalen Funktion möglich.**

Zusammenfassung

- MATLAB = Numerikprogramm für effizientes Lösen mathematischer Berechnungen.
- Interpretersprache, an mathematische Formulierungen angelehnt.
- Geeignet für Prototyping mathematischer Verfahren.
- Zahlreiche Visualisierungsmöglichkeiten.