

Einführung in Matlab - Einheit 1

Streifzug durch Matlab, Vektoren und Matrizen

Jochen Schulz

Georg-August Universität Göttingen 

- Anmeldung über StudIP
<https://www.studip.uni-goettingen.de/>
Einführung in Matlab (Mathematische Anwendersysteme)
- Alle Unterlagen (Aufgabenblätter, Vorlesungsfolien, Beispiele, Musterlösungen) → StudIP

- Anmeldung über StudIP
<https://www.studip.uni-goettingen.de/>
Einführung in Matlab (Mathematische Anwendersysteme)
- Alle Unterlagen (Aufgabenblätter, Vorlesungsfolien, Beispiele, Musterlösungen) → StudIP

Dozent

Jochen Schulz

NAM, Zimmer 04 (Erdgeschoß)

Telefon: 39-4525

Email: schulz@math.uni-goettingen.de

XMPP: [schulz@jabber.num.math.uni-goettingen.de](xmpp:schulz@jabber.num.math.uni-goettingen.de)

Ablauf der Veranstaltung

- Blockveranstaltung vom 12.9 - 23.9.2011
- **Vorlesung:** 9.15 - 11.30 (MN)
- **Übungsbetrieb/Praktikum:** 13:00 - 18:00 (MM-Raum oder NAM-CIP)
 - 1 Übungszettel/Tag.
 - Besprechung Aufgaben vom Vortag
 - Hilfestellung beim Bearbeiten der aktuellen Aufgaben
- **Klausur:** 30.9.2011 (?); 10:00 - 11:30; Anmeldung über FlexNow.

Inhalt der Vorlesung

1. **Tag** Organisatorisches, Streifzug durch Matlab
2. **Tag** Programmieren, Datenstrukturen
3. **Tag** Rekursionen, Grafik
4. **Tag** Polynome, Interpolation, Visualisieren, Debugging
5. **Tag** Mehrdimensionale Arrays, Funktionen, Dünnbesetzte Matrizen, Numerische Mathematik
6. **Tag** ?
7. **Tag** Grafik-Handle, GUIs
8. **Tag** Schnittstelle zu C
9. **Tag** Wunschvorlesung
10. **Tag** Fragestunde

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien



2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

- MATLAB steht für **Mat**rix **lab**oratory; ursprünglich speziell Matrizenrechnung.
- Entwickelt von Cleve Moler Ende der 70'er in FORTRAN.
- Heutige Version ist in C/C++ programmiert.
- Interaktives System für numerische Berechnungen und Visualisierungen.
- Kein Computer-Algebra-System (Aber erweiterbar durch `symbolic math toolbox`)

Vorteile von MATLAB

- *High-Level* Sprache:
 - Programmieren ist leicht (aber auch beschränkt(er))
 - Schnelle Erfolge
 - Sehr geeignet für Prototyping und Debugging
- Vielfältige Visualisierungsmöglichkeiten.
- MATLAB-Programme sind vollständig portierbar zwischen Architekturen (cross-plattform).
- zusätzliche Toolboxes (Symb. Math T., PDE T., Wavelet T.)
- Ausgereifte Oberfläche.

-  Matlab online-help :-).
-  **Matlab Guide**, D.J. Higham, N.J. Higham, SIAM, 2000,
-  **Introduction to Scientific Computing**, C.F. van Loan, Prentice Hall, New Jersey, 1997,
-  **Scientific Computing with MATLAB**, A. Quarteroni, F. Saleri, Springer, 2003,
-  **Graphics and GUIs with MATLAB**, P. Marchand, O.Th. Holland, Chapman & Hall, 2003, 3. Aufl.
-  **MATLAB 7**, C. Überhuber, St. Katzenbeisser, D. Praetorius, Springer 2005.
-  **Using MATLAB**, offizielle Handbücher.

1 Streifzug durch MATLAB

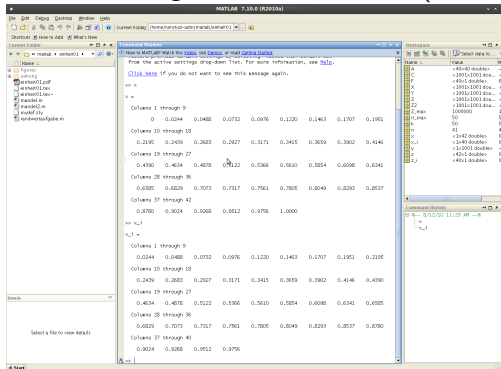
- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

MATLAB Fenster-Aufbau

Starten von MATLAB: Eingabe von `matlab` & (in einem Terminal).



- **Launch Pad:** Startmenü

- **Command Window:** Befehlseingabe und Standardausgabe

- **Workspace:** Ansicht von Variablen und deren Art und Grösse; Ändern der Einträge
- **Grafik:** normalerweise in separaten Fenstern

Command Window - Befehle

- Erster Befehl

```
>> 2+2  
ans = 4
```

- Editieren alter Eingaben: ↑, ↓ (wie in Unix)
- Mit ; am Ende jeder Befehlszeile wird Standardausgabe unterdrückt.

```
>> 2+2;
```

- Hilfe zu Befehlen: `help <command>` oder `doc <command>`
- Zuweisung

```
>> a = 2+2;
```

- Funktionsaufruf

```
>> sin(2)  
ans = 0.9093
```

- Verlassen von MATLAB: `quit` oder `exit`

Workspace - globale Variablen

- Alle definierten (globalen) Variablen werden im Workspace gespeichert.
- Zugriff während einer MATLAB-Sitzung.
- Inhalt des Arbeitsspeichers: `whos` oder `who`

```
>> whos
```

Name	Size	Bytes	Class
ans	1x1	8	double

- Löschen von Variablen : `clear <var>;`
`clear` löscht den gesamten Arbeitsspeicher (Workspace).

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Erste Schritte

- MATLAB als Taschenrechner
(Ergebnis wird in `ans` gespeichert.)

```
>> 1+(sin(pi/2)+ exp(2))*0.5  
ans = 5.1945
```

- Eingabe von (Zeilen-)Vektoren

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

- Transponieren und speichern in Variable `b`

```
>> b = transpose(x)
```

Erste Schritte II

- Erzeugen einer Matrix

```
A = [0 2 3 ; 4 5 6; 7 8 9];
```

- Lösen des Gleichungssystems $A \cdot z = b$

```
z = A \ b
```

- Probe

```
A*z
```

Erste Schritte III

- Berechnen der Determinante von A

```
det(A)
```

- Hilfe zu `det`

```
help det
```

```
DET      Determinant.
```

```
DET(X) is the determinant of the square matrix X.
```

```
Use COND instead of DET to test for matrix  
singularity.
```

- Erzeugen einer Einheitsmatrix

```
B = eye(3,3)
```

Erste Schritte IV

- Matrizenoperationen

$A+B$, $A-B$, $A*B$, $\text{inv}(A)$

- Anwendung von Vektoren

$y = \text{sqrt}(x)$

$y =$

1.0000

1.4142

1.7321

Erste Schritte V

- Komponentenweise Multiplikation

```
y = x.*x
```

y =
1 4 9

- Zeilenvektor mit Werten von 1 bis 100

```
a = [1:100];
```

- Berechne $\sum_{j=1}^{100} \frac{1}{j^2}$

```
(1./a)*transpose(1./a)
```

ans = 1.6350

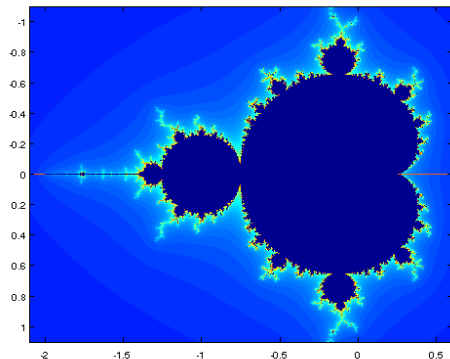
1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Die Mandelbrot-Menge



Die Mandelbrot-Menge ist die Menge von Punkten $c \in \mathbb{C}$ bei denen die Folge $(z_n)_n$, die durch

$$\begin{aligned} z_0 &:= c \\ z_{n+1} &= z_n^2 + c, \quad n \in \mathbb{N} \end{aligned}$$

definiert ist, beschränkt ist.

Die Mandelbrot-Menge

```
x = linspace(-2.1,0.6,601);  
y = linspace(-1.1,1.1,401);  
  
[X,Y] = meshgrid(x,y);  
C = complex(X,Y);  
  
it_max = 50;  
  
Z = C; Anz = zeros(size(Z));  
  
for k = 1:it_max  
    Z = Z.^2+C;  
    Anz = Anz + isfinite(Z);  
    waitbar(k/it_max);  
end  
image(x,y,Anz);  
title('Mandelbrot Set', 'FontSize',16);
```


Verwendete Befehle

- `linspace(a,b,n)`
ist ein Vektor mit n Einträgen der Form $a, a + (b - a)/(n - 1), \dots, b$
- `[X,Y] = meshgrid(x,y)`
erzeugt Matrizen

$$X = \begin{pmatrix} x_1 & \dots & x_n \\ & \vdots & \\ x_1 & \dots & x_n \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 & \dots & y_1 \\ & \vdots & \\ y_n & \dots & y_n \end{pmatrix}$$

- `C = complex(X,Y)`
erzeugt $C = (C(j, k))_{jk}$ mit $C(j, k) = X(j, k) + i Y(j, k)$

Verwendete Befehle

- `B = isfinite(A)`
Matrix B hat gleiche Größe wie A . Die Einträge sind 1, wenn der entsprechende Eintrag von B finit ist und 0 sonst.
- `image(x,y,A)`
erzeugt eine Grafik auf der Basis des Gitters (x,y) mit Werten A .
Durch den entsprechenden Eintrag von A wird die Farbe bestimmt.
- `title`
Überschrift der Grafik.
- `for, end`
Schleife (Details später).

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Probleme beim Mandelbrot:

- Bei jeder Änderung von z.B. `it_max` muss alles erneut im interaktiven Modus eingegeben werden.
- Abrufen der Befehle bei späteren Sitzungen ist kaum möglich.
- Bei komplexen Algorithmen wird es unübersichtlich.

Ausweg: Die Befehlsfolge wird in einer Datei abgelegt. MATLAB arbeitet dann sukzessive die einzelnen Kommandos ab.

Erzeugen eines Programms

- Starten des Editors:

```
edit datei_name
```

öffnet die Datei `datei_name`.

- Speichern der Datei mit Hilfe des Menüs: `File->Save` bzw. `File->Save As` oder per Shortcut.
- Kommentarzeilen beginnen mit `%`.

Achtung: Alle MATLAB-Dateien haben die Endung `'.m'`. Man spricht deswegen auch von *m*-Files.

Struktur von Skript-Files

- Skript-Files bestehen aus einer Sequenz von Befehlen, die nacheinander abgearbeitet werden.
- Am Anfang des Files als Kommentar:
 - Name des Programms
 - (kurze) Beschreibung
 - Autor-Informationen und Datum
- operiert auf Daten im *Workspace*.
- Gestartet wird das Programm `name.m` durch Eingabe von `name`.
- Beschreibung des Skript-Files:

```
help plot_poly
```

```
-----  
      plot_poly.m  
zeichnet den Graphen eines Polynoms  
Gerd Rabin           1.11.2003  
-----
```

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- **Function-Files**
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Functions - Graph eines Polynoms

Aufgabe:

Zeichnen Sie den Graphen eines Polynoms

$$p(x) = \sum_{i=0}^N a_i x^i, \quad a_i \in \mathbb{R}$$

Problem:

Zu Werten $(x_i)_{i=1}^n$ muß man $(p(x_i))_{i=1}^n$ berechnen, d.h. Funktionswerte müssen sehr oft berechnet werden.

Lösung:

Funktionen


```
function y=ausw_poly1(a,x)
%-----
% ausw_poly berechnet den Funktionswert von
%       $p(x)=a_1 + a_2 x + a_3 x^2 + \dots + a_n x^{(n-1)}$ 
%      INPUT:  a Vektor der Koeffizienten
%              x auszuwertender Punkt
%      OUTPUT: y Funktionswert (y=p(x))
%      Gerd Rapin      1.11.2003
%-----

n = length(a);
aux_vector = x.^(0:n-1);
y = aux_vector*transpose(a);
```

Vektorielle Version

```
function y = ausw_poly2(a,x)
%-----
% ausw_poly berechnet den Funktionswert von
%       $p(x)=a_1 + a_2 x + a_3 x^2 + \dots + a_n x^{(n-1)}$ 
%      INPUT:  a Vektor der Koeffizienten
%              x Vektor der auszuwertenden Punkte
%      OUTPUT: y Vektor der Funktionswerte
%      Gerd Rapin      1.11.2003
%-----

n = length(a);
k = length(x);
A = repmat(transpose(x),1,n);
B = repmat(0:(n-1),k,1);

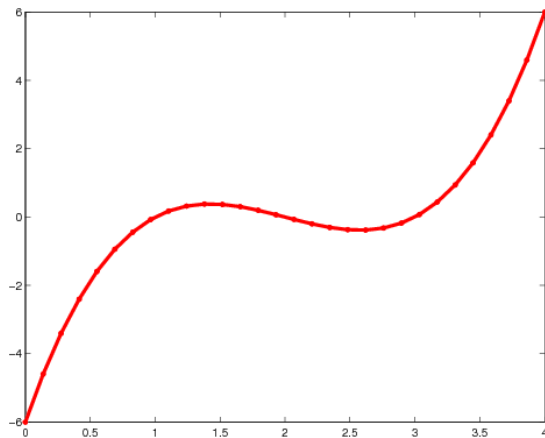
y = (A.^B)*transpose(a);
```

Plotten des Polynoms

```
%-----  
%      plot_poly.m  
%  
%  zeichnet den Graphen eines Polynoms  
%  
%  
%  Gerd Rapin              1.11.2003  
%-----  
  
% Koeffizienten  
a = [9 0 -10 0 1];  
  
x = linspace(0,4,30); % Betrachte [0,4]  
y = ausw_poly2(a,x);  
  
% Plotten  
plot(x,y,'r*-','LineWidth',3,'MarkerSize',4)
```

Plotten des Polynoms

$$p(x) = (x-1)(x-2)(x-3)$$



Struktur von Function-Files

Beispiel: 'myfunction.m'

```
function [Out_1,...,Out_k] = myfunction(In_1,...,In_l)
% Beschreibung der Funktion
..
Out_1=..
..
Out_k=..
```

Soll keine Variable zurückgegeben werden, so besteht die erste Zeile aus

```
function myfunction(In_1,...,In_k)
```

Wichtig: Funktionsname = Dateiname.

- Funktionen sind mit Kommentaren zu versehen:
 - (kurze) Beschreibung
 - Input-Argumente
 - Output-Argumente
 - Autor-Informationen und Datum
- Variablen lokal, d.h.
 - Variablen des Workspace sind nicht verfügbar.
 - definierte Variablen werden nicht im Workspace gespeichert.

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Priorität beim Programmaufruf

Beispiel-Programmaufruf

name

Testet ob, ..

- 1 Variable
- 2 **Unterfunktion**. Eine Unterfunktion ist ein Programm/Funktion, die in derselben Datei wie der Aufruf steht.
- 3 Programm im **aktuellen Verzeichnis**.
- 4 *private function*.
- 5 Programm im **Suchpfad**.

Bei gefundenem Namen wird die Suche beendet.

Der Suchpfad (Variable `path`) enthält Verzeichnisse in einer geordneten Liste.

- Abarbeitung erfolgt der Ordnung gemäss.
- Suchpfade hinzufügen:

```
addpath <pfadname>
```

- Suchpfade entfernen:

```
rmpath <pfadname>
```

Verwalten von m-Files

- `doc <name>`
öffnet grafisches Hilfefenster zum jeweiligen Programm.
- `lookfor <name>`
Suche nach `name` in den Kommentaren zu den Funktionen (auch: grafisches Hilfefenster).
- `what`
m-Files im aktuellen Verzeichnis.
- `type <name>`
Inhalt von `name.m` (Command Window).
- `which <name>`
absoluter Pfad der Datei, in dem die Funktion `name` gespeichert ist.
- `edit <name>`
Ruft den Editor mit `name.m` auf.

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Vektoren I

- Erzeugen 'per Hand'

```
b = [1 2 4]
```

```
b =  
    1    2    4
```

- Abfragen der Einträge von b

```
b(2)
```

```
ans =    2
```

Index \equiv Position im Vektor

Achtung: Indizes beginnen immer mit 1!

Doppelpunkt - Notation

`x:s:z` erzeugt einen Vektor der Form

$$(x, x + s, x + 2s, x + 3s, \dots, z).$$

```
>> a = 2:11
a =
    2    3    4    5    6    7    8    9   10   11

>> c = -2:0.75:1
c =
-2.0000 -1.2500 -0.5000  0.2500  1.0000
```

Vektoren II

- `length(a)`
Länge des Vektors a an.

- `linspace(x1,x2,N)`
Vektor

$$x_1, x_1 + \frac{x_2 - x_1}{N - 1}, x_1 + 2\frac{x_2 - x_1}{N - 1}, \dots, x_2$$

der Länge N .

```
linspace(1,2,4)
```

```
ans =  
    1.0000    1.3333    1.6667    2.0000
```

- `logspace(x1,x2,N)`
wie `linspace`, nur logarith. Skalierung

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Erzeugen von Matrizen

- Erzeugen 'per Hand'

```
B = [1 3 4; 5 6 7]
```

B =

1	3	4
5	6	7

- `eye(n,m)`
($n \times m$)-EinheitsMatrix)

```
eye(2,3)
```

ans =

1	0	0
0	1	0

(1 auf der Hauptdiagonalen und 0 sonst).

Erzeugen von Matrizen II

- `zeros(n,m)`
($n \times m$)- Matrix mit 0 als Einträge.
- `ones(n,m)`
($n \times m$)- Matrix mit 1 als Einträge.
- Blockmatrizen

```
C = [B zeros(2,2); eye(2,3) eye(2,2)]
```

C =

1	3	4	0	0
5	6	7	0	0
1	0	0	1	0
0	1	0	0	1

Achtung: Matrizen in einer Zeile müssen dieselbe Zeilenanzahl haben und Matrizen in einer Spalte dieselbe Spaltenanzahl.

Erzeugen von Matrizen III

- `repmat(A,n,m)`

Blockmatrix mit $(n \times m)$ aus A bestehenden Blöcken

```
D = repmat(B,1,2)
```

D =

1	3	4	1	3	4
5	6	7	5	6	7

- `blkdiag(A,B)`

Blockdiagonalmatrix.

- `diag(v,k)`

Matrix der Größe $(n + |k|) \times (n + |k|)$ mit den Einträgen des Vektors v auf der k -ten Nebendiagonalen.

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Beispiel-Matrizen

- Überblick: `help gallery`
- Beispiel

```
E = gallery('moler',4)
```

E =

1	-1	-1	-1
-1	2	0	0
-1	0	3	1
-1	0	1	4

- Hilfe zur Matrix 'moler' erhält man durch `help private/moler`
- weitere Matrizen: `magic`, `hilb`, `vander`

Zugriff auf Matrizen

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

A =

1	2	3
4	5	6
7	8	9

Abfragen eines Eintrags

```
>> A(2,1)
ans =
     4
```

Abfrage von Blöcken

```
>> A(2:3,1:2)
ans =
     4     5
     7     8
```

Abfrage einer Zeile

```
>> A(2,:)
ans =
     4     5     6
```

Abfrage mehrerer Zeilen

```
>> A([1 3], :)
ans =
     1     2     3
     7     8     9
```

Löschen

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

A =

1	2	3
4	5	6
7	8	9

Löschen einer Zeile

```
>> A(2,:) = []
```

A =

1	2	3
7	8	9

Löschen von Spalten

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

A =

2
5
8

1 Streifzug durch MATLAB

- Einleitung
- Grundlegende Bedienung
- Erste Schritte
- Etwas komplexeres Beispiel
- Skript-Files und der Editor
- Function-Files
- Verwaltung von Dateien

2 Vektoren und Matrizen

- Erzeugen von Vektoren
- Erzeugen von Matrizen
- Manipulation von Matrizen
- Matrix- und Vektoroperationen

Matrizenoperationen

Standard-Matrix Operationen $+$, $-$, $*$

```
A = [1 2; 3 4]; B = 2*ones(2,2);
```

Multiplikation

```
>> A*B
```

```
ans =
```

```
     6     6
    14    14
```

Addition

```
>> A+B
```

```
ans =
```

```
     3     4
     5     6
```

Subtraktion

```
>> A-B
```

```
ans =
```

```
    -1     0
     1     2
```

Andere Operatoren

- $A \setminus B$
Lösung X von $A * X = B$.
- A / B
Lösung X von $X * A = B$.
- $\text{inv}(A)$
Inverse von A .
- A'
oder $\text{ctranspose}(A)$: komplex Transponierte von A .
- $A.'$
oder $\text{transpose}(A)$: Transponierte von A .
- A^z
(quadratische Matrizen) $\underbrace{A * A * \dots * A}_{z\text{-mal}}$
- $\text{size}(A)$
Größe einer Matrix A .

Punktnotation

```
A = [1 2; 3 4]; B = 2*ones(2,2);
```

- $C(i,j) = A(i,j) * B(i,j)$.

```
>> C = A.*B
```

```
C =
```

```
     2     4
     6     8
```

- $C(i,j) = A(i,j)/B(i,j)$.

```
>> C = A./B
```

```
C =
```

```
    0.5000    1.0000
    1.5000    2.0000
```

Punktnotation

- $C(i,j) = B(i,j)/A(i,j)$.

```
>> C = A.\B
C =
    2.0000    1.0000
    0.6667    0.5000
```

- $C(i,j) = A(i,j)^{B(i,j)}$

```
>> C = A.^B
C =
     1     4
     9    16
```

Achtung: Dimension von A und B gleich.

Matrizen können durch Skalare ersetzt werden, z.B. $A.^2$.

Skalarprodukt

- Vektoren $a = (a_1, \dots, a_n)$, $b = (b_1, \dots, b_n)$
- Skalarprodukt: ab^t
- Summe der Einträge von a : $(1, \dots, 1)a^t$

Beispiel:

```
>> a=1:100; b=linspace(0,1,100);  
>> a*transpose(b)  
ans =  
    3.3667e+03  
>> ones(1,100)*transpose(a)  
ans =  
    5050
```