

# Mathematik am Computer

## 4. Übung: Matlab, Teil I

Marcus Grote und Helmut Harbrecht

Universität Basel

18. – 22. Oktober 2021

- 1 Grundlegendes
  - Matrizen
  - Bedienung von Matlab
- 2 Matlab als Taschenrechner
  - Operationen auf Matrizen
  - Operationen der Linearen Algebra
- 3 Matlab als Programmiersprache
  - Skripte und Funktionen
- 4 Graphische Ausgabe
  - Funktionen zeichnen

# Definition von Matrizen

Eine lineare Abbildung  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$  ist eindeutig bestimmt durch die Bilder

$$\mathbf{c}_1 = \Phi(\mathbf{e}_1), \dots, \mathbf{c}_n = \Phi(\mathbf{e}_n)$$

auf der Standardbasis  $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ .

Schreiben wir  $\mathbf{c}_1, \dots, \mathbf{c}_n$  in ein rechteckiges Schema

$$(\mathbf{c}_1 | \dots | \mathbf{c}_n),$$

so erhalten wir eine **Matrix**. Eine Matrix hat  $mn$  Elemente in  $m$  Zeilen und  $n$  Spalten.

# Definition von Matrizen

Sei  $\mathbb{R}^{m \times n}$  die Menge der **reellen Matrizen** mit  $m$  Zeilen und  $n$  Spalten.

Umgekehrt definiert jede Matrix  $\mathbf{A} = (\mathbf{a}_1 | \dots | \mathbf{a}_n) \in \mathbb{R}^{m \times n}$  eindeutig eine lineare Abbildung  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$  durch

$$\mathbf{e}_i \mapsto \mathbf{a}_i, \quad i = 1, \dots, n.$$

Beachte:  $\mathbf{e}_i$  sind die Einheitsvektoren in  $\mathbb{R}^n$  und  $\mathbf{a}_i \in \mathbb{R}^m$ .

# Grundrechenarten für Matrizen

Die **Grundrechenarten** für Matrizen sind folgendermassen definiert<sup>1</sup>:

- Addition, Subtraktion, Multiplikation mit Skalaren elementweise:  
 $\mathbf{A} = (a_{ij})_{i,j=1}^{m,n}$ ,  $\mathbf{B} = (b_{ij})_{i,j=1}^{m,n} \in \mathbb{R}^{m \times n}$ :

$$\mathbf{A} \pm \mathbf{B} = (a_{ij} \pm b_{ij}) \quad \alpha \mathbf{A} = (\alpha a_{ij})$$

- Definiere die Matrix-Vektor-Multiplikation so, dass für  $\mathbf{A} \in \mathbb{R}^{m \times n}$  und  $\mathbf{x} \in \mathbb{R}^n$  gilt:  $\mathbf{y} = \mathbf{Ax} \in \mathbb{R}^m$ :

$$y_i = \sum_{j=1}^n a_{ij} x_j,$$

wobei gilt  $\mathbf{y} = (y_i)_{i=1}^m$ ,  $\mathbf{x} = (x_i)_{i=1}^n$ .

---

<sup>1</sup>So entsprechen sie den Grundrechenarten für lineare Abbildungen.

# Grundrechenarten für Matrizen

- Matrixmultiplikation<sup>2</sup>: Für  $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{l \times m}$ ,  $\mathbf{B} = (b_{jk}) \in \mathbb{R}^{m \times n}$  ist

$$\mathbf{C} = \mathbf{AB} = (c_{ik}) \in \mathbb{R}^{l \times n} \text{ mit } c_{ik} = \sum_{j=1}^m a_{ij} b_{jk}.$$

Ist  $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$  und  $\mathbf{b} \in \mathbb{R}^m$  so ist

$$\mathbf{Ax} = \mathbf{b}$$

ein lineares Gleichungssystem für einen unbekanntem Vektor  $\mathbf{x} \in \mathbb{R}^n$  mit  $m$  Gleichungen und  $n$  Unbekannte.

---

<sup>2</sup>Sie entspricht der Komposition von linearen Abbildungen.

# Beispiele

## Grundrechenarten für Matrizen

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} - \begin{pmatrix} 3 & 2 & 1 \\ 0 & 10 & 0 \end{pmatrix} = \begin{pmatrix} -2 & 0 & 2 \\ 4 & -5 & 6 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 + 2 \cdot 0 + 3 \cdot 2 \\ 4 \cdot 1 + 5 \cdot 0 + 6 \cdot 2 \end{pmatrix} = \begin{pmatrix} 7 \\ 16 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 2 & 0 \end{pmatrix} = \begin{pmatrix} 7 & 3 \\ 16 & 9 \end{pmatrix}$$

# Beispiele

Lineares Gleichungssystem  $\mathbf{Av} = \mathbf{b}$

Das lineare Gleichungssystem

$$\begin{cases} x - y + z = 0 \\ y - z = -24 \\ x + 2z = 0 \end{cases}$$

ist äquivalent zu

$$\underbrace{\begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & 2 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} x \\ y \\ z \end{pmatrix}}_{\mathbf{v}} = \underbrace{\begin{pmatrix} 0 \\ -24 \\ 0 \end{pmatrix}}_{\mathbf{b}}$$



# Grundlegendes

## Matlab Features:

- 1 **Command Window** zum direkten Ausführen von Matlab-Befehlen
- 2 **Command History** Liste ausgeführter Befehle. Erneutes Ausführen durch Anklicken
- 3 **Current Folder** Aktuelles Verzeichnis — hier werden selbstprogrammierte Befehle gesucht.
- 4 **Workspace** Liste aller aktuell belegten Variablen
- 5 **Editor** zum Schreiben von eigenen Befehlsequenzen (Skripte und Funktionen)

**Abbruch:** `strg+C`.

# Beispiel

MATLAB R2019b - academic use

HOME PLOTS APPS EDITOR PUBLISH VIEW

FILE NAVIGATE EDIT BREAKPOINTS RUN

Current Folder: C:\Users\SRJM\Dropbox\Doktorat Simon Michel\Lehre\HS19 Praktikum\Serie 04\Eigene Loesung

Editor - Untitled

```

1

```

Workspace

Name	Value	Size
ans	122	1x1
v	[2;3]	2x1
x	11	1x1

Command Window

```

>> v = [2;3]

v =

     2
     3

>> x = v(1) + 3*v(2)

x =

    11

>> x^2 + 1

ans =

    122

```

Command History

```

$-- 14.10.2019 16:10 --$
clear
close all
clc
v = [2;3]
x = v(1) + 3*v(2)
x^2 + 1

```

script | Ln 1 | Col 1

# Matlab-Bedienung und Hilfe

Aufgrund der grossen Anzahl von Befehlen und weiterer Optionen gibt es Hilfestellungen um sich die volle Funktionalität und Syntax einzelner Befehle zu merken bzw. sich daran zu erinnern:

- 1 `help` + Funktionsname gibt die Hilfe dazu aus
- 2 Matlab-Hilfsmenü `doc`: Umfangreiche Suchmöglichkeiten
- 3 Buchstabenfolge + Tabulator: Automatische Befehlsergänzung
- 4 Pfeil-nach-oben: Befehlsverlauf
- 5 Online Ressourcen (`doc` ist auch über den Browser einsehbar, Foren, usw.)

Literatur:

- Gute deutschsprachige Einführung:  
<ftp://ftp.math.ethz.ch/users/wpp/NUMSTATMETH/Y2009/EinfMATLAB.pdf>

# Matlab als Taschenrechner

## Operationen auf Matrizen

Erzeugen von Matrizen:

`[1 2 3 ; 4 5 6]`      `zeros(2,3)`      `eye(5)`

Erzeugen von speziellen Zeilenvektoren:

`1:3` ergibt den Vektor

`( 1   2   3 )`.

`1:0.2:2` ergibt

`( 1.0   1.2   1.4   1.6   1.8   2.0 )`.

Zuweisung (um Werte in einer Variablen abzuspeichern):

`c1 = 6`      `A = [1 2 3 ; 4 5 6]`      `v = 1:10`

`x = [1;2;3]`      `y = [1 2 3]`

```
Command Window
>> c1 = 6
c1 =
    6
>> A = [1 2 3 ; 4 5 6]
A =
    1    2    3
    4    5    6
>> v = 1:10
v =
    1    2    3    4    5    6    7    8    9   10
>> x = [1;2;3]
x =
    1
    2
    3
>> y = [1 2 3]
y =
    1    2    3
fx >> |
```

# Matlab als Taschenrechner

## Operationen auf Matrizen

Verkleben von Matrizen:

`[A, zeros(2,2)]` ergibt

$$\begin{pmatrix} 1 & 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 \end{pmatrix}$$

`B = [A, zeros(2,2); eye(2), A]` speichert die Matrix

$$\begin{pmatrix} 1 & 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 \\ 1 & 0 & 1 & 2 & 3 \\ 0 & 1 & 4 & 5 & 6 \end{pmatrix}.$$

in der Variablen  $B$  ab.

# Matlab als Taschenrechner

## Operationen auf Matrizen

Elementauswahl für Zugriff und Zuweisung:

`A(1,2)` `A(1:2,2)` `A(1:2,1:3)` `A(:, [1 3])`

Elementweise *arithmetische* Operationen:

`+` `-` `.*` `./` `.^` `.'`

Elementweise Funktionen:

`abs` `sin` `cos` `exp` `sqrt` `min` `max` ...

Elementweise logische Operationen:

Das Ergebnis ist 0 (`false`) bzw. 1 (`true`).

`==` `~=` `<` `>` `<=` `>=` `&` `|` `~`

# Matlab als Taschenrechner

## Arithmetische Operationen der Linearen Algebra

### Grundrechenoperationen der Linearen Algebra

+   -   \*   ^

### Weitere Operationen:

$A \setminus B$

$\mathbf{A}^{-1} \cdot \mathbf{B}$

$A'$

$\mathbf{A}^\top$  (falls  $\mathbf{A}$  reell)

$\det(A)$

$\det(\mathbf{A})$

$\text{inv}(A)$

$\mathbf{A}^{-1}$

$\text{rank}(A)$

$\text{rang}(\mathbf{A})$



# Matlab als Taschenrechner

## Das Lösen von linearen Gleichungssystemen

Eingabe:

$$A = [ 1 \ 2 \ ; \ 3 \ 4 ]$$

$$b = [ 3 \ ; \ 5 ]$$

$$A \setminus b$$

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

$$\mathbf{b} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

$$\mathbf{A}^{-1}\mathbf{b}$$

Angabe ist die Lösung von  $\mathbf{Ax} = \mathbf{b}$ :

ans =

-1

2

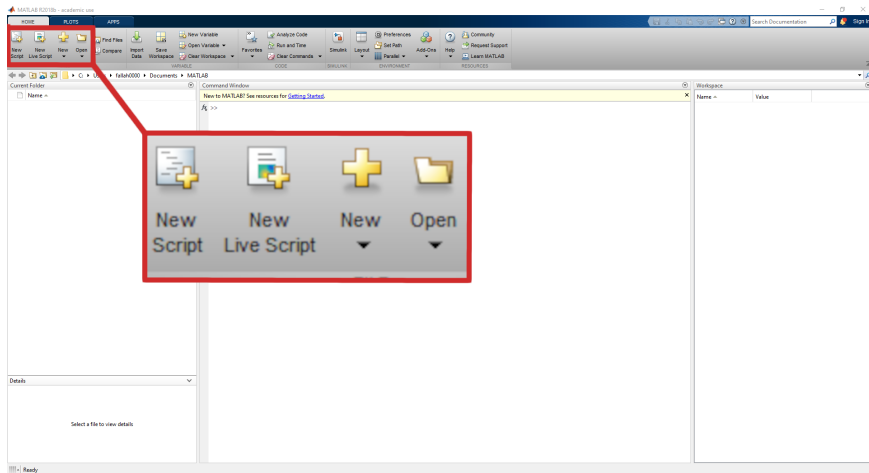
# Skripte

## Erläuterungen

- Eine Datei, welche eine Abfolge von Befehlen enthält, heisst **Skript** oder Programm.
- Name der Datei beliebig, Dateierweiterung muss `.m` sein.
- Aufruf des Skripts (ohne Dateierweiterung) führt Skript sequenziell aus.
- Semikolon `;` unterdrückt die Ausgabe.
- Kommentare beginnen mit einem Prozentzeichen `%`.

**Achtung:** Ein MATLAB-Skript und ein MATLAB-Live Skript ist nicht dasselbe!

# Skript erstellen



Für die Abgabe der Aufgaben sollen MATLAB-Skripte erstellt werden,  
z.B. S04A02.m, S04A03.m, S04A04.m usw.

# Skripte

## Einschränkungen

### Einschränkungen bei Namen von Variablen und Dateien

- alle Namen müssen sich unterscheiden
- Gross- und Kleinbuchstaben werden unterschieden
- Namen müssen mit einem Buchstaben beginnen
- Namen dürfen keine Sonderzeichen enthalten

# Funktionen einer Veränderlichen zeichnen

Matlab zeichnet keine Funktionen, sondern **Wertetabellen** als **Polygonzug!**

- 1 Definiere **Spaltenvektor**  $x$  von  $x$ -Werten, z.B.

```
x = (0 : pi/100 : pi)';
```

- 2 Definiere **Spaltenvektor**  $y$  von  $y$ -Werten einer Funktion, z.B.

```
y = sin(x);
```

- 3 Zeichne Wertetabelle

```
plot(x,y);
```

- 4 Alternativ:

```
plot(x, sin(x));
```

Beachte: Die Funktion, die gezeichnet werden soll, muss **elementweise** auf  $x$  anwendbar sein!

# Funktionen einer Veränderlichen zeichnen

Um **mehrere** Funktionen zu zeichnen, geht man wie folgt vor:

- 1 Definiere **Spaltenvektor**  $x$  von  $x$ -Werten, z.B.

```
x = (0 : pi/100 : pi)';
```

- 2 Definiere **Spaltenvektoren**  $y$  und  $z$  von  $y$ -Werten von Funktionen, z.B.

```
y = sin(x); z = cos(x);
```

- 3 Zeichne Wertetabellen

```
plot(x, y, x, z);
```

- 4 Alternativ:

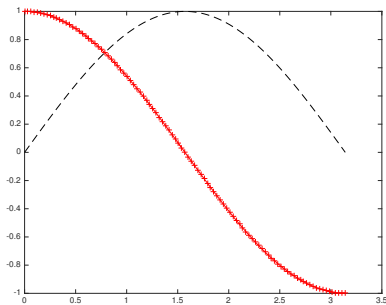
```
plot(x, [y, z]);
```

```
plot(x, sin(x), x, cos(x));
```

# Plotoptionen und Modifizierung von Graphen

- 1 Weitere Optionen beim Zeichnen von Kurven, z.B.

```
plot(x, sin(x), 'k--', x, cos(x), 'r-+')
```



# Plotoptionen und Modifizierung von Graphen

## 2 Modifizierung der Achsen:

```
axis([xmin xmax ymin ymax])  
axis equal  
xlim([xmin xmax])   usw.
```

## 3 Beschriftung:

```
title('Überschrift')  
xlabel('x-Achse')  
legend('Graph 1', 'Graph 2', 'Graph 3')
```

## 4 Weitere Modifizierungen direkt am Graphen möglich!



# Plotoptionen und Modifizierung von Graphen

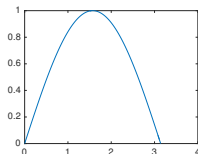
## Der subplot-Befehl

Um mehrere Plots **in einem Fenster** darstellen zu lassen, benutzen wir den Befehl

```
subplot(m, n, k)
```

Beispiel:

```
subplot(2, 2, 1)  
plot(x, sin(x))
```



# Plotoptionen und Modifizierung von Graphen

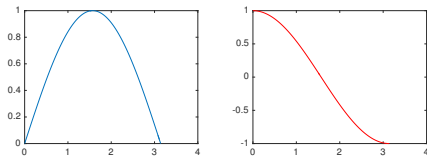
## Der subplot-Befehl

Um mehrere Plots **in einem Fenster** darstellen zu lassen, benutzen wir den Befehl

```
subplot(m, n, k)
```

Beispiel:

```
subplot(2, 2, 1)  
plot(x, sin(x))  
subplot(2, 2, 2)  
plot(x, cos(x), 'r')
```



# Plotoptionen und Modifizierung von Graphen

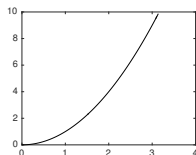
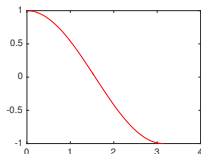
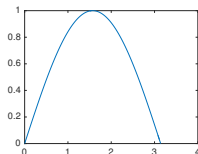
## Der subplot-Befehl

Um mehrere Plots **in einem Fenster** darstellen zu lassen, benutzen wir den Befehl

```
subplot(m, n, k)
```

Beispiel:

```
subplot(2, 2, 1)  
plot(x, sin(x))  
subplot(2, 2, 2)  
plot(x, cos(x), 'r')  
subplot(2, 2, 4)  
plot(x, x.^2, 'k')
```



# Plotoptionen und Modifizierung von Graphen

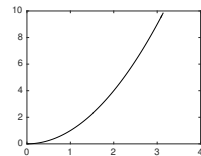
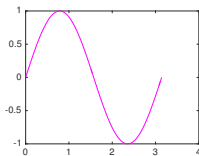
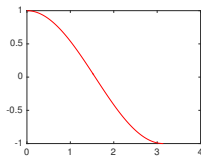
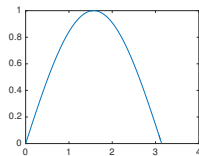
## Der subplot-Befehl

Um mehrere Plots **in einem Fenster** darstellen zu lassen, benutzen wir den Befehl

```
subplot(m, n, k)
```

Beispiel:

```
subplot(2, 2, 1)
plot(x, sin(x))
subplot(2, 2, 2)
plot(x, cos(x), 'r')
subplot(2, 2, 4)
plot(x, x.^2, 'k')
subplot(2, 2, 3)
plot(x, sin(2*x), 'm')
```



# Plotoptionen und Modifizierung von Graphen

## Der figure-Befehl

- Der Befehl `figure(n)` öffnet ein **Fenster** mit der Nummer  $n$ .
- Alle weiteren Plot-Befehle werden in diesem Fenster ausgeführt
- Mit den Befehlen `figure(1)`, `figure(2)`, usw. kann man zwischen den Fenstern 1, 2, usw. beliebig wechseln.
- Auch Plots in Subplot-Grids lassen sich mit `title('...')` einzeln betiteln; Dem gesamten Subplot-Grid kann mittels `sgtitle('...')` ein Titel hinzugefügt werden.
- Mit `close n` und `close all` lassen sich ein bestimmtes bzw. alle Fenster wieder **schliessen**.