

Modellbildung und Simulation

Einführung in MATLAB

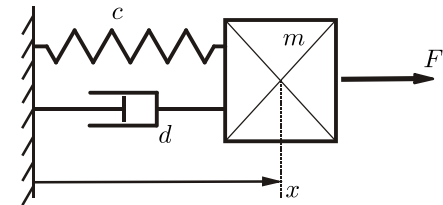
Institut für Technische Mechanik – Teilinstitut Dynamik/Mechatronik

Allgemeines und Lernziele

- *MATLAB® ist eine höhere Programmiersprache und interaktive Umgebung für numerische Berechnungen, Visualisierung und Programmierung. MATLAB dient zur Datenanalyse, Algorithmen-Entwicklung und zur Erstellung von Modellen und Anwendungen. Mit der Programmiersprache, den Tools und den integrierten mathematischen Funktionen können Sie verschiedene Ansätze ausprobieren und schneller zu einer Lösung gelangen als mit Tabellenkalkulationen oder herkömmlichen Programmiersprachen wie C/C++ oder Java™ (www.mathworks.de)*

Lernziele

- Kennenlernen der Benutzeroberfläche
- Verstehen der Programmsyntax
- Anwenden einfacher Befehle
- Numerische Simulation eines Ein-Massen-Schwingers

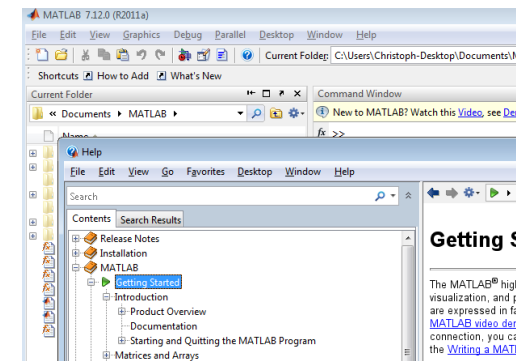


Literatur

- MATLAB® Hilfe
- Bücher: z.B. über KIT-Bibliothek

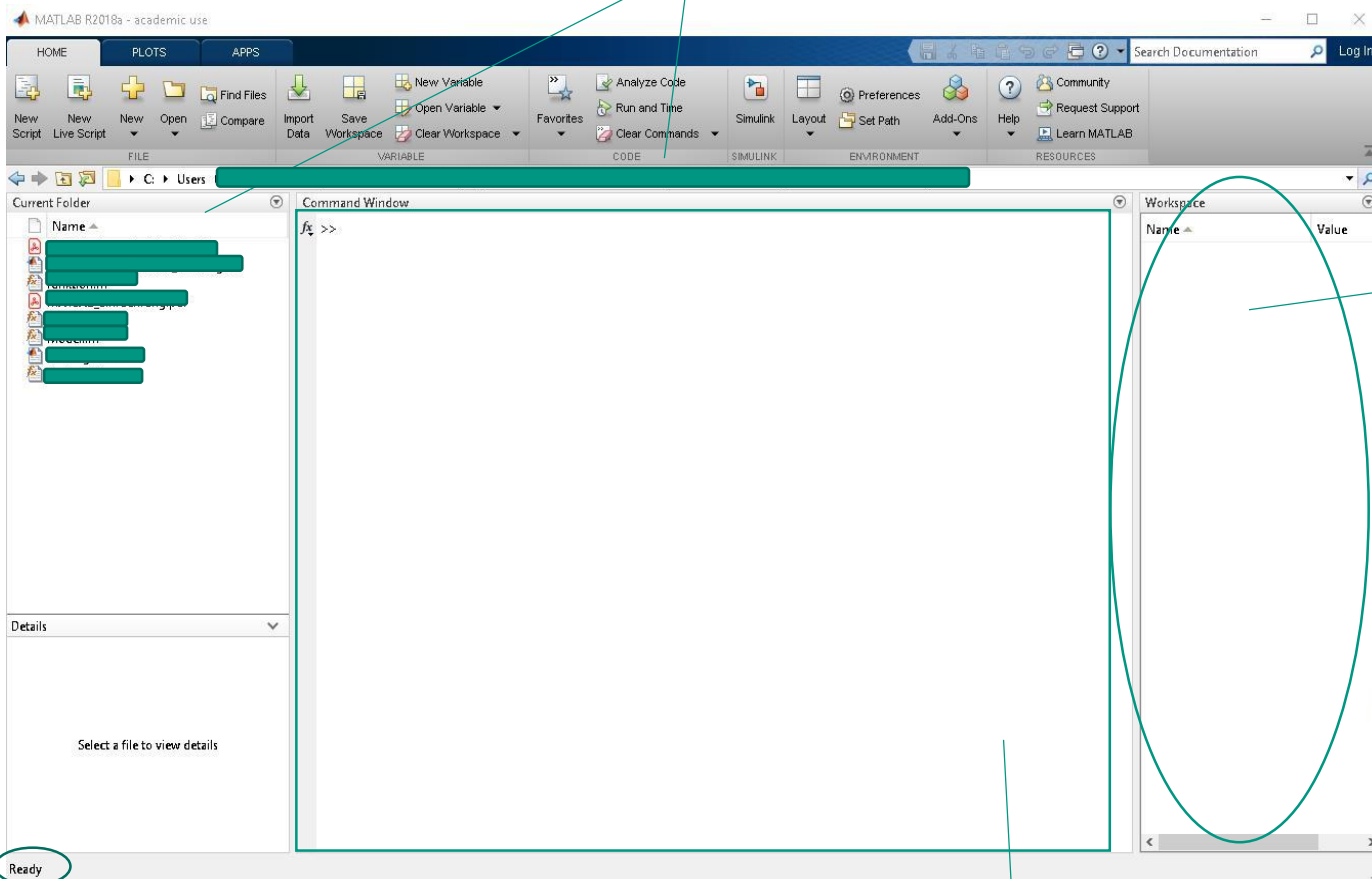
Software

- MATLAB® kann über die SCC-Webseite bezogen werden.
<http://www.scc.kit.edu/produkte/3841.php>




Oberfläche

Arbeitsverzeichnis



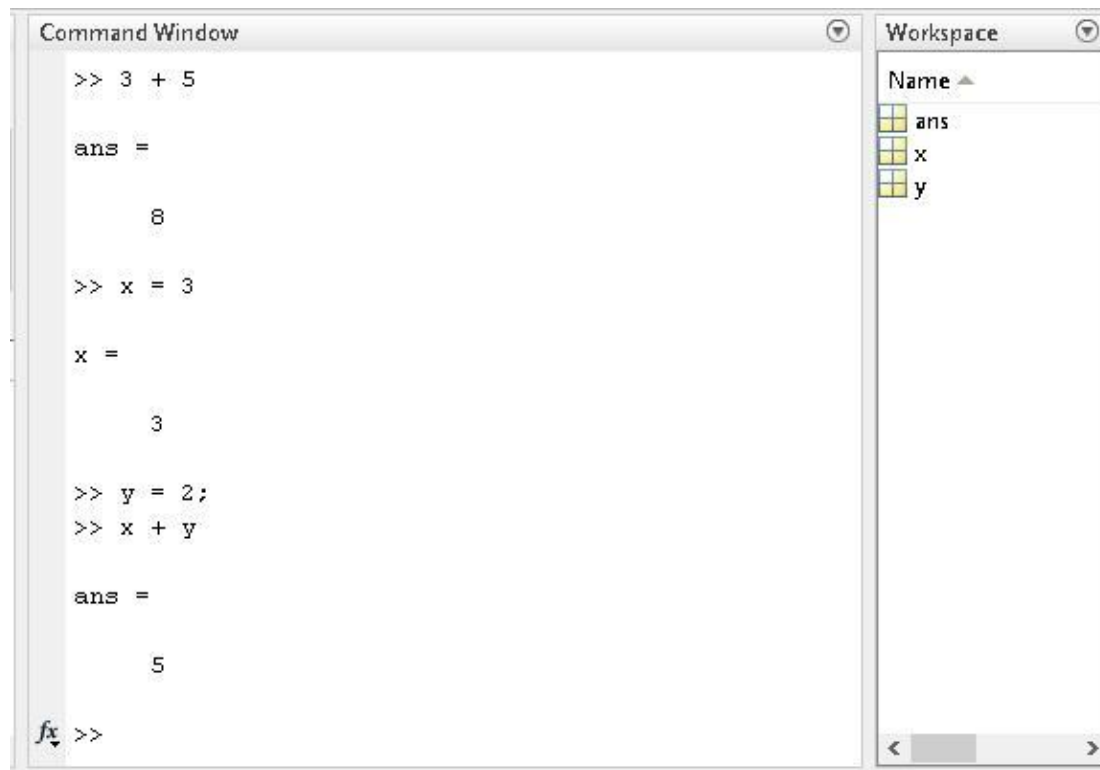
,workspace'
alle Variablen
samt Größe,
Zahlenformat, usw.

,command window'
Befehlsoberfläche;
letzte Befehle mit 

Status

Command Window

- Jeder Befehl kann direkt über das **,command window‘** eingegeben und ausgeführt werden. Insbesondere können auf die Variablen im **,workspace‘** zugegriffen werden.
- z.B.



Erzeugung eines m-file (Skript)

- Mehrere Befehle können in sog. *m-files* gespeichert werden, um dann gemeinsam ausgeführt zu werden.

1. Neuer m-file: Icon



2. Code in m-file schreiben:

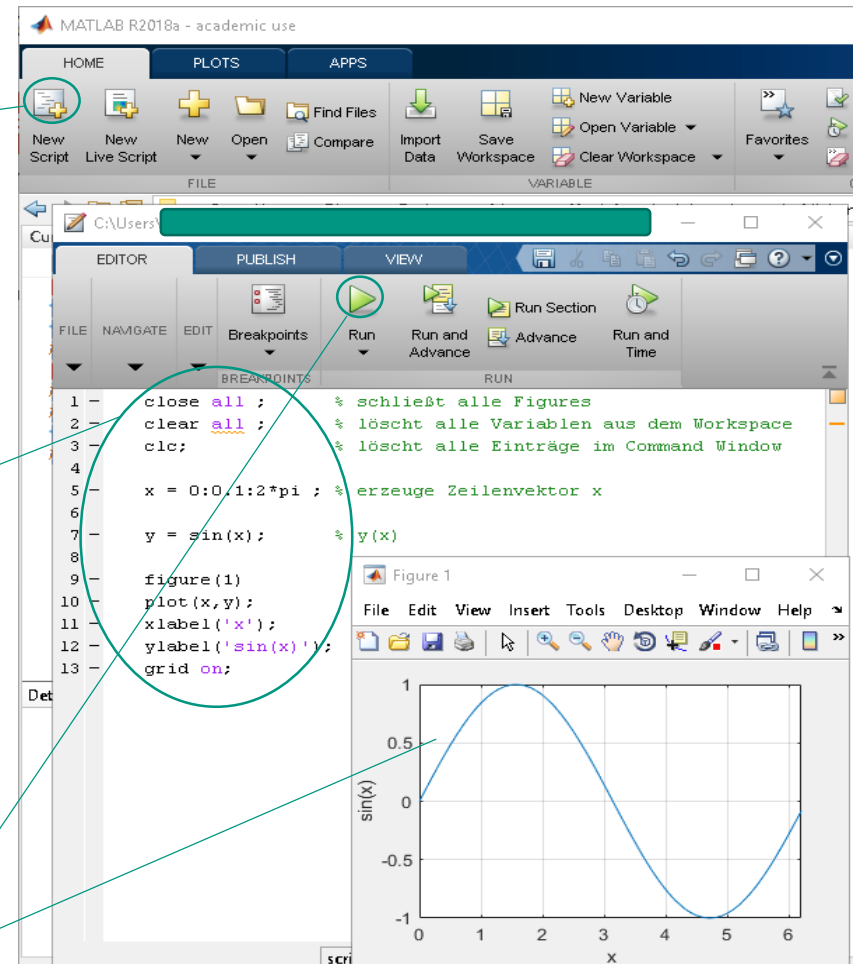
- Endet Befehl mit ;
→ keine Ausgabe
- Endet Befehl ohne ;
→ Ausgabe im **,command window‘**
- Jedes Skript sollte folgendermaßen anfangen:

```
close all; Schließt alle figures
clear all; Leert den ,workspace‘
clc; Leert das ,command window‘
```

3) m-file ausführen: Icon



4) Ergebnis



Matrizen/Array

- Eingabe in eckigen Klammern []
- Einträge einer Zeile durch , oder Leerzeichen getrennt
- Zeilen durch ; oder Zeilenumbruch getrennt
- Vektoren sind [1xn] - bzw. [nx1] -Matrizen
- Skalare sind [1x1] -Matrizen (Klammern können entfallen)
- Beispiel:

■ >> A=[1, 2; 3, 4];

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

■ >> B=[1 2
3 4];

→ $\mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

Matrizendefinition und -operationen

- Leere Matrix: $A = [\]$
- $[n \times m]$ -Matrix mit allen Einträgen =1: `ones (n , m)`
- $[n \times m]$ -Matrix mit allen Einträgen =0: `zeros (n , m)`
- $[n \times m]$ -Einheitsmatrix: `eye (n , m)`
- Zuweisung: `a = 1 ;`
- Addition, Subtraktion: `C=A+B ; C=A-B ;`
- Matrixmultiplikation: `C=A*B ;`
- Elementweise Multiplikation / Division: `C=A.*B ; =A./B ;`
- Inversion einer Matrix: `inv (A)` oder A^{-1}
- Transponieren einer Matrix: `A.'` oder A' , wenn A reell
- Lösung von $Ax=B$: `x=inv (A) *B`

Matrizenoperationen

$$\mathbf{a} = [a_1, a_2, \dots, a_n], \mathbf{b} = [b_1, b_2, \dots, b_n], c = \langle \text{Skalar} \rangle$$

Addition Matrix-Skalar	$\mathbf{a} + c = [a_1 + c, a_2 + c, \dots, a_n + c]$
Multiplikation Matrix-Skalar	$\mathbf{a} * c = [a_1 * c, a_2 * c, \dots, a_n * c]$
Addition Matrix-Matrix	$\mathbf{a} + \mathbf{b} = [a_1 + b_1, a_2 + b_2, \dots, a_n + b_n]$
Element. Multiplikation Matrix-Matrix	$\mathbf{a} . * \mathbf{b} = [a_1 * b_1, a_2 * b_2, \dots, a_n * b_n]$
Elementweise Division Matrix-Matrix	$\mathbf{a} . / \mathbf{b} = [a_1 / b_1, a_2 / b_2, \dots, a_n / b_n]$
Elementweise Potenzieren	$\mathbf{a} . ^ \mathbf{b} = [a_1 ^ b_1, a_2 ^ b_2, \dots, a_n ^ b_n]$
Matrix Multiplikation	$\mathbf{C} [n, n] = \mathbf{a} [n, m] * \mathbf{b}' [m, n] \quad \text{mit}$
	$c_{ij} = \sum_{k=1}^m a_{ik} b_{kj}, \quad i, j = 1..n$

Matrizenmanipulation

Anfangswert : Schrittweite : Endwert

>> **B**=[0:0.2:1]; \Rightarrow **B** = $\begin{bmatrix} 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 \end{bmatrix}$

Matrizen können aus Submatrizen aufgebaut werden!

>> **A**=[1,2] ; **B**=[3,4] ;

>> **C**=[**A**,**B**] ;

>> **D**=[**A**;**B**] ;

>> **E**=[zeros(2,2) , ones(2,2) ; **A**,**B**]

C = $\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$

D = $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

E = $\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$

Zeilen- und Spaltenanzahl müssen zueinander passen!

Indizierung

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

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

→ Matricelement(e) ansprechen: **Matrix(Zeilenindex, Spaltenindex)**

```
>> B = A(2, 1)
```

2. Zeile, 1. Spalte
→
 $\mathbf{B} = [4]$

```
>> C = A(:, :)
```

ALLE Zeilen, ALLE Spalten
→
 $\mathbf{C} = \mathbf{A}$

```
>> D = A(1, :)
```

1. Zeile, ALLE Spalten
→
 $\mathbf{D} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

```
>> D = A(2, 2:3)
```

2. Zeile, 2. und 3. Spalte
→
 $\mathbf{D} = \begin{bmatrix} 5 & 6 \end{bmatrix}$

→ Vektorelement(e) ansprechen: **Vektor(Elementindex)**

```
>> A = [1, 2, 3, 4];
```

```
>> B=A(1:2);
```

Element 1 bis 2
→
 $\mathbf{B} = \begin{bmatrix} 1 & 2 \end{bmatrix}$

```
>> C=A(3:end);
```

Element 3 bis Ende
→
 $\mathbf{B} = \begin{bmatrix} 3 & 4 \end{bmatrix}$

Graphische Darstellung

- Zeichnen von $(x_i; y_i)$ -Wertepaaren
- Linienstil und Farbe:
- Zeichnen von mehreren Kurven:
- Achsenbeschriftung:
- Überschrift / Titel:
- Wertebereich:
- gleiche Skalierung der Achsen:
- Beispiel:

```

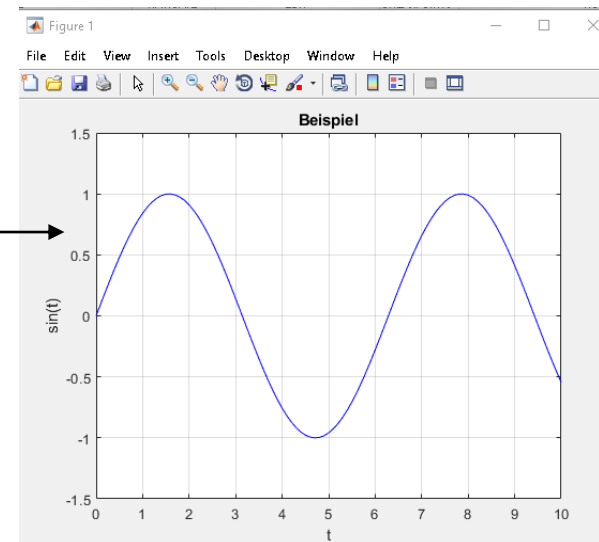
plot(x,y)
plot(x,y,'option')
plot(x1,y1,'opt1',x2,y2,'opt2')
xlabel('...'); ylabel('...')
title('...')
axis([xmin; xmax; ymin; ymax])
axis equal

```

```

>> t=[0:0.1:10]; x=sin(t);
>> plot(t,x,'b')
>> xlabel('t')
>> ylabel('sin(t)')
>> title('Beispiel')
>> axis([0; 10; -1.5; 1.5])
>> grid on; (zeichnet Gitter)

```



Kontrollstrukturen

- if-Abfrage:

```
if logischer Ausdruck  
    Befehle  
elseif logischer Ausdruck  
    Befehle  
else ← optional  
    Befehle  
end
```

- for-Schleife

```
for zaehler= start:schrittweite:ende  
    Befehle  
end
```

- while-Schleife

```
while logischer Ausdruck  
    Befehle  
end
```

Funktionen

- Eigenständige Dateien mit der **Dateinamen-Erweiterung .m**

```
function [Ausgabeliste] = Name(Eingabeliste)
```

```
% Kommentare
```

Matlab-Code

Matlab
m-File

- **Name** sollte dem Dateinamen entsprechen
- Variablen sind lokal (nur in der Funktion) definiert!
- Funktionen werden in **MATLAB®** mit ihrem Dateinamen aufgerufen (ohne .m)

```
function [a,b] = FunktionName(x,y)
```

```
a=x+y;
```

```
b=x*y;
```

Funktion (m-File)

- Aufruf der Funkt. „FunktionName“ z.B. im **„command window“** mit $x=1, y=2$
→ Rückgabe A & B

```
>> [A,B] = FunktionName(1,2)
```

→ A = 3 und B = 2

Numerische Integration gewöhnlicher Differentialgleichungen (DGL/engl.: ode)

- in MATLAB® muss dem solver („Löser“) neben Parametern auch die DGL an sich übergeben werden
- Ein *Differentialgleichungssystem* (DGLS) in Zustandsform $\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}, t)$ ist vollständig durch seine rechte Seite bestimmt
- DGLS in Zustandsform wie eine Funktion in ein .m-File schreiben (z.B. modell.m)
- Mit dem solver ode45 (es gibt noch weitere solver) ist das DGLS lösbar,
 - Beispiel
 - `[T,X] = ode45(@modell,[0 10], [0; 1],a,b,c);`
 - Löst das in modell.m gespeicherte DGLS für die Zeit t=0 bis t=10 mit den Anfangsbedingungen [0; 1] (Vektor!) und den Parametern a,b,c
 - Ergebnis: ein Spaltenvektor T, der die Zeitpunkte der Integration enthält und die Lösungsvektoren (Matrix X)
 - Ergebnis: Zeichnen: plot(T,X);

Numerische Integration, Beispiel

- DGL: $m \ddot{x} + d \dot{x} + c x = f \rightarrow$ Zustandsform: $\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{1}{m} (f - d \dot{x} - c x) \end{bmatrix}$

- **Modellfile**

```
function xp=Modell(t,x,m,d,c,f)
xp=[x(2); 1/m*(f-d*x(2)-c*x(1))];
```

- **Hauptfile**

- Parameter definieren

```
m=1; d=0.5; c=10; f=3;
```

- Zeitvektor/AB's

```
Zeit=[0, 10]; AB=[2;0];
```

- Toleranzoptionen für numerisch Integration:

```
SolverOptionen=odeset('RelTol',1e-3,'AbsTol',1e-6);
```

- Ode45-Solver

```
[T,Y]=ode45(@Modell,Zeit,AB,SolverOptionen,m,d,c,f);
```

- Plotten

```
plot(T,Y(:,1),'b',T,Y(:,2),'r');
```