

# Einführung

## MATLAB and SIMULINK

Christian Hespe (`christian.hespe @ tuhh.de`)  
Lennart Heeren (`lennart.heeren @ tuhh.de`)

Hamburg University of Technology  
Institute of Control Systems

13.11.2020

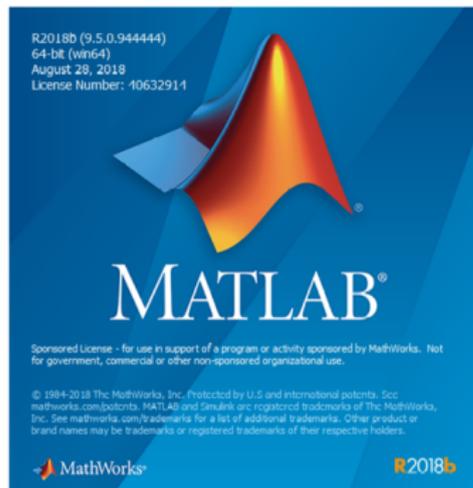
# Inhalt

- 1 MATLAB installieren
- 2 Help!
- 3 MATLAB Basics
- 4 MATLAB for Control
- 5 Der Gleichstrommotor

# Inhalt

- 1** MATLAB installieren
- 2 Help!
- 3 MATLAB Basics
- 4 MATLAB for Control
- 5 Der Gleichstrommotor

# MATLAB herunterladen und installieren



- Registrieren und herunterladen (min Ver. 2019a),
  - Dauer je nach Internetverbindung und Rechenpower bis zu 2 Stunden,
- `./setup.exe`
- Zum Installieren,
  - Das Motorexperiment funktioniert mit 32-Bit *und* 64-Bit.

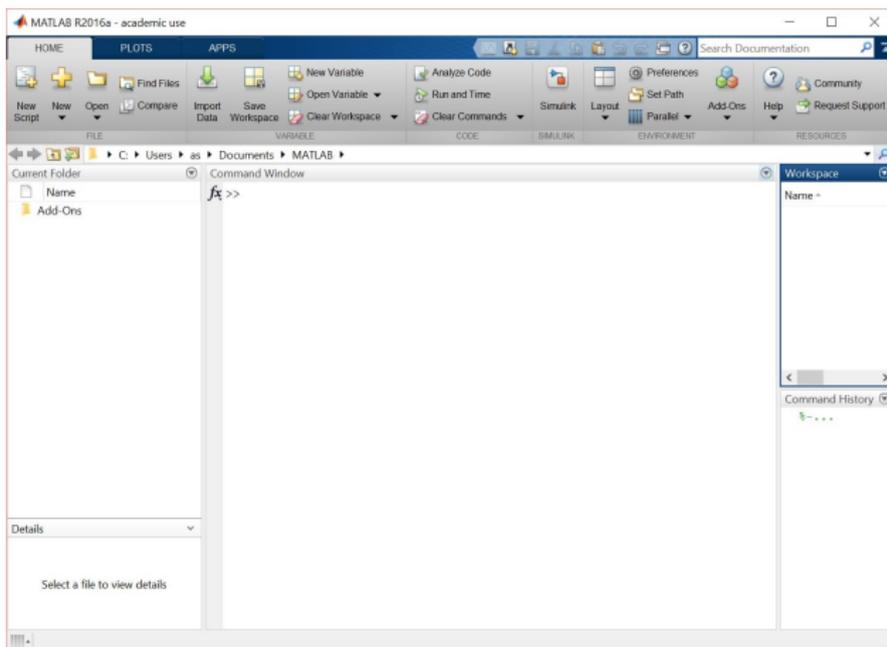
## Weitere Infos:

[www.tuhh.de/rzt/software/numerik/matlab](http://www.tuhh.de/rzt/software/numerik/matlab)

# Inhalt

- 1 MATLAB installieren
- 2 Help!**
- 3 MATLAB Basics
- 4 MATLAB for Control
- 5 Der Gleichstrommotor

# MATLAB Oberfläche



- MATLAB 2016a nach dem Öffnen,
- Aktuelle Version in den Pools: MATLAB 2019a

# Finding Help



## **help** FUNCTION

- Hilfe zum Befehl `FUNCTION` im Command Window.

## **doc** FUNCTION

- Hilfe zum Befehl `FUNCTION` in gesondertem Fenster.



- Hilfe zum Befehl auf dem sich der Cursor befindet im Pop-Up Fenster.

# Inhalt

- 1 MATLAB installieren
- 2 Help!
- 3 MATLAB Basics**
- 4 MATLAB for Control
- 5 Der Gleichstrommotor

# Matrizen

- MATLAB speichert (fast) alles als Matrizen

$a = [1, 2, 3];$       Ein Zeilenvektor  $a = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

---

$b = [1; 2; 3];$       Ein Spaltenvektor  $b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

---

$c = \begin{bmatrix} 1, & 2, & 3 & ; \dots \\ 4, & 5, & 6 & ; \dots \\ 7, & 8, & 9 & ];$       Eine Matrix  $C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

# Daten Generieren

- Funktionen zum Erzeugen von Vektoren und Matrizen

```
t = 0:0.01:10;
```

Bspw. interpretierbar als  
Zeitvektor in 10 ms Schritten:  
 $t = [0 \quad 0.01 \quad 0.02 \quad \dots \quad 10]$

```
u = ones(1, length(t));
```

Ein Zeilenvektor voller 1en,  
derselben Länge wie t

```
I = eye(3);
```

Matrix  $I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

**Tipps:** zeros, rand, ...

# Rechenoperationen

- „Intuitive“ Operatoren +, -, \*, /, ...
- Elementweise Operatoren .\*, ./, ...

**Gegeben**      $a = [1, 2, 3];$     $b = [1; 2; 3];$

---

$c = a*b;$      Inneres Produkt:  $c = 14$

---

$C = b*a;$      Dyadisches Produkt:  $C = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$

---

$c = a.*a;$      Elementweises Produkt:  
 $c_i = a_i \cdot a_i, \quad c = \begin{bmatrix} 1 & 4 & 9 \end{bmatrix}$

---

# Rechenoperationen und Indizierung

- Transponieren  $a'$
- Zugriff auf Elemente per Indizierung  $a(i, j)$

**Gegeben**

$$a = [1, 2, 3]; \quad b = [1; 2; 3];$$

---

$$c = a * a;$$

Dimensionen passen nicht!

---

$$c = a * a';$$

Inneres Produkt:  $c = a \cdot a^T = \|a\|^2 = 14$

---

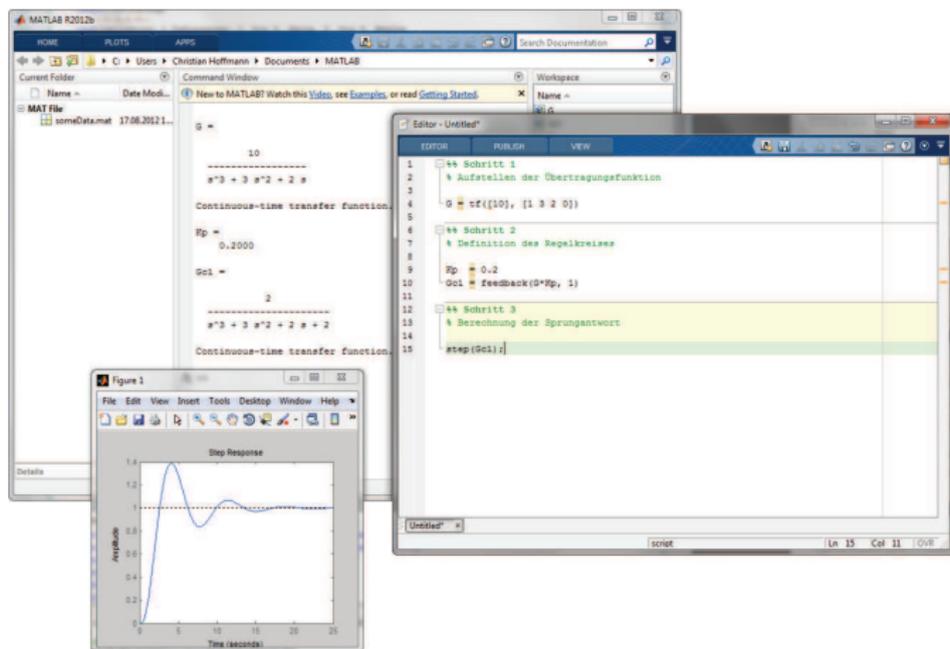
$$c = a(1:2);$$

Indizierung:  $c = \begin{bmatrix} 1 & 2 \end{bmatrix}$

---

$$C = C(1:3, 2:3);$$
 Indizierung: 1. bis 3. Zeile, 2. bis 3. Spalte

# MATLAB Skripts und Funktionen



- Ausführen eines Skripts

# MATLAB Skripts und Funktionen

- Komplizierte/Längere Berechnungen können auch in Skripts (m-files) für die spätere Wiederverwendung gespeichert werden,
- Skripts können in Abschnitte (Zellen) unterteilt werden.

`edit`      Ruft den Editor auf

## Beispiel

---

```
%% Zellenanfang  
% Kommentar: In dieser Zelle wird  
%                    der Sinus berechnet
```

```
t = 0:0.01:1;  
y = sin(t);
```

---

# Code Vektorisieren

- MATLAB unterstützt natürlich `for` Schleifen u.ä.,
- Aber: Eine mathematische Darstellung (Vektorisierung) ist meist schneller und kürzer.

```
i = 0;                                     Berechnet 1001 Werte für sin(t)
for t = 0:0.01:10
    i = i + 1;
    y(i) = sin(t);
end
```

---

```
t = 0:0.01:10;                             Identische Variable y
y = sin(t);
```

# Ergebnisse Darstellen

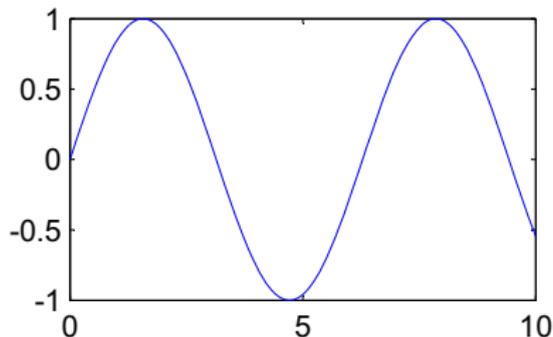
- Plots erzeugen...

```
t = 0:0.01:10;
```

```
y = sin(t);
```

```
plot(t, y);
```

Berechnet 1001 Werte für  $\sin(t)$   
und plottet  $y$  über  $t$



- ...und per graphischer Benutzeroberfläche editieren

## Ergebnisse Darstellen

- Mehrere Plots in einem Graphen erzeugen

```
t = 0:0.01:10;
```

```
y = sin(t);
```

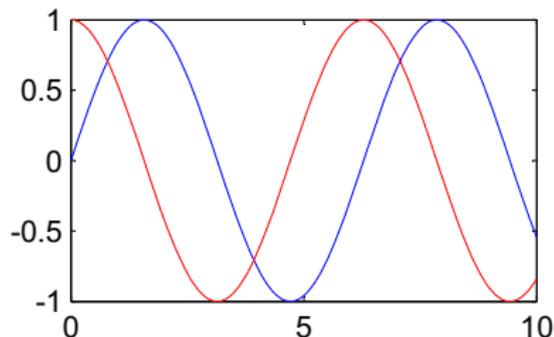
```
x = cos(t);
```

```
plot(t, y); hold on
```

```
plot(t, x);
```

Plottet  $y$  über  $t$  und  $x$  über  $t$  in ein Fenster

`hold on` verhindert, dass das Fenster überschrieben wird.



# MATLAB Workspace

- MATLAB Workspace enthält alle definierten Variablen
- Der Workspace (oder ein Teil) kann gespeichert werden

**save** myWS                      Speichert alle Variablen in  
myWS.mat

---

**save** myWS a b                Speichert die Variablen a und b in  
myWS.mat

- Das Speichern von bereits verarbeiteten Daten kann viel Zeit sparen,



# Inhalt

- 1 MATLAB installieren
- 2 Help!
- 3 MATLAB Basics
- 4 MATLAB for Control**
- 5 Der Gleichstrommotor

# Grundlegende Befehle für die Regelungstechnik

- Die Control Systems Toolbox bietet schon viele Befehle

$G = \mathbf{tf}(\text{num}, \text{den});$  Übertragungsfunktion definieren

---

$G = \mathbf{ss}(A, B, C, D);$  Zustandsraummodell definieren

---

$\mathbf{step}(G);$  Sprungantwort plotten

---

$\mathbf{lsim}(G, u, t);$  Systemantwort auf Eingang  $u$  plotten

---

$\mathbf{feedback}(G1, G2);$  System aus Feedbackverknüpfung berechnen

# Lösen einer Aufgabe mit Skripten

## Beispiel

- 1 Definition von  $G(s) = \frac{10}{s^3 + 3s^2 + 2s}$ ,
- 2 Sprungantwort plotten,
- 3 Schließen des Regelkreises mit einem P Regler.

# SIMULINK

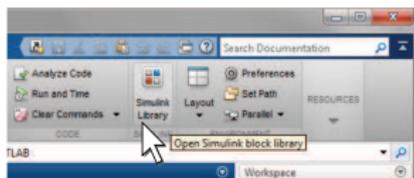
- SIMULINK ist eine graphische Benutzeroberfläche (Drag & Drop) zur Erstellung von Blockdiagrammen zur Simulation,
- Erstellen von Block Diagrammen ähnlich denen aus der Literatur.

# Starten von SIMULINK

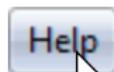
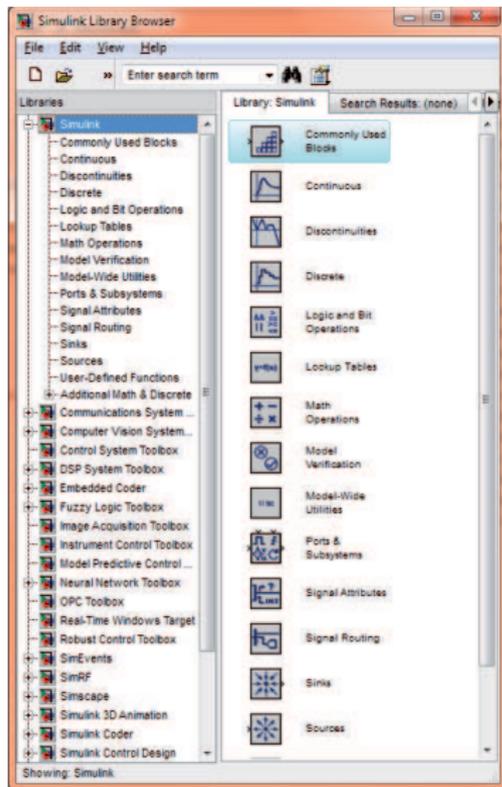
**simulink**

Startet SIMULINK bei Eingabe im Command Window

Alternativ



# SIMULINK Grundfunktionalitäten



Hilfe konsultieren!



Erzeugt neues Modell

## Wichtige Libraries

- *Commonly Used Blocks*
- *Continuous*
- *Sinks*
- *Sources*

# Lösen einer Aufgabe mit SIMULINK

## Beispiel

- 1 Definition von  $G(s) = \frac{10}{s^3 + 3s^2 + 2s}$ ,
- 2 Sprungantwort plotten,
- 3 Schließen des Regelkreises mit einem P Regler.

# Inhalt

- 1 MATLAB installieren
- 2 Help!
- 3 MATLAB Basics
- 4 MATLAB for Control
- 5 Der Gleichstrommotor**

# DC Motor

- Tragbares Koffer-Experiment
- Benötigt für Übungsaufgaben mit ***Motor Experiment***
- **Bestandteil von Übungen** in den PC-Pools
- Ausleihbar über die Bibliothek
  - **Stichwort:** DC Motor Control Experiment
  - **Wie lange:** max. 7 Tage
- Durchführung der Experimente
  - am eigenen Rechner
  - in den Computerpools
- Probleme und Fragen:  
online Diskussionsforum (Stud.IP)



# Installation Guide

## Installiert werden muss...

- der Arduino-Treiber
  - 1) neue Windows-Versionen: automatische Installation
  - 2) falls nicht:  
<https://www.arduino.cc/en/Guide/ArduinoDue#toc8>
  - 3) Überprüfen ob erfolgreich: Control Panel/Hardware and Sound/Devices and Printers

**Hilfe: *DC Motor User's Guide.pdf***

# User's Guide

- Motor-Simulink-Modelle etc. auf GitLab herunterladen
- Anschluss des Motors per USB
- Öffne Matlab
- Ordner mit den Motordateien als *'Current Folder'*
- Füge die Unterordner hinzu (Rechtsklick, "Add to Path" → "Selected Folders and Subfolders")
- Motor-Block in *'MotorLib.slx'* (Source block: MotorLib/motor)

Hilfe: *DC Motor User's Guide.pdf*

# User's Guide

- Motor-Simulink-Modelle etc. auf GitLab herunterladen
- Anschluss des Motors per USB
- Öffne Matlab
- Ordner mit den Motordateien als '*Current Folder*'
- Füge die Unterordner hinzu (Rechtsklick, "Add to Path" → "Selected Folders and Subfolders")
- Motor-Block in '*MotorLib.slx*' (Source block: MotorLib/motor)

Hilfe: *DC Motor User's Guide.pdf*

# User's Guide

- Motor-Simulink-Modelle etc. auf GitLab herunterladen
- Anschluss des Motors per USB
- Öffne Matlab
- Ordner mit den Motordateien als '*Current Folder*'
- Füge die Unterordner hinzu (Rechtsklick, "Add to Path" → "Selected Folders and Subfolders")
- Motor-Block in '*MotorLib.slx*' (Source block: MotorLib/motor)

Hilfe: *DC Motor User's Guide.pdf*

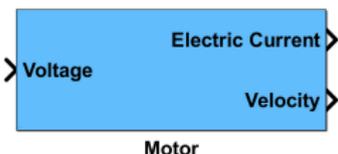
# User's Guide

- Motor-Simulink-Modelle etc. auf [GitLab](#) herunterladen
- Anschluss des Motors per USB
- Öffne Matlab
- Ordner mit den Motordateien als '*Current Folder*'
- Füge die Unterordner hinzu (Rechtsklick, "Add to Path" → "Selected Folders and Subfolders")
- Motor-Block in '*MotorLib.slx*' (Source block: MotorLib/motor)

**Hilfe: *DC Motor User's Guide.pdf***

# User's Guide

- Motor-Simulink-Modelle etc. auf [GitLab](#) herunterladen
- Anschluss des Motors per USB
- Öffne Matlab
- Ordner mit den Motordateien als '*Current Folder*'
- Füge die Unterordner hinzu (Rechtsklick, "Add to Path" → "Selected Folders and Subfolders")
- Motor-Block in '*MotorLib.slx*' (Source block: MotorLib/motor)

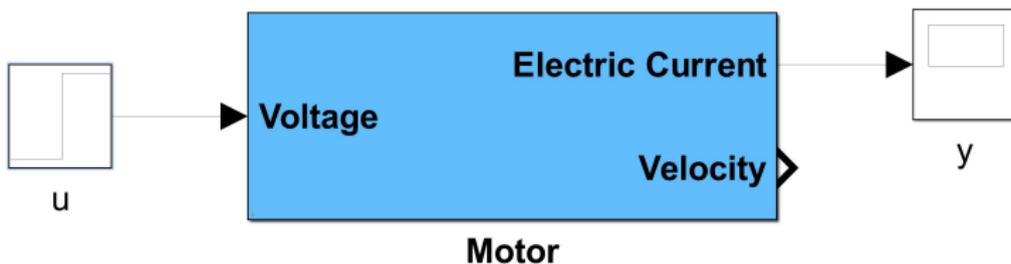


**Hilfe: *DC Motor User's Guide.pdf***

# Experimente Durchführen

**Test:** Simulink-Testdatei zur Überprüfung der Verbindung zum Motor

passendes Template: *test\_openloop.mdl*



# Motor-Block Einstellungen

Block Parameters: Motor

Motor Interface Block (mask) (link)

This block is sending the input as voltage to the motor and reads the electric current of the motor and the desired output and sends them as outputs.

To read the instruction, click on "help" button please.

Parameters

2nd Output Type Velocity

Sample Time

0.01

COM Port Number

4

OK Cancel Help Apply

- **2nd Output Type:**  
hier velocity
- **Sample Time [s]:**  
fast immer 10ms
- **COM Port Number:**  
Device Manager/ ports/  
Arduino Due  
Programming Port

Ende

**Vielen Dank für Ihre Aufmerksamkeit!**