Einführung in moderne Simulationstechniken
Kursteil: MATLAB / Simulink

Carsten Herzog

Institut für Steuer- und Regelungstechnik
Fakultät für Luft- und Raumfahrttechnik
Universität der Bundeswehr München

Teil 1: Einführung in MATLAB

- Was ist MATLAB?
- MATLAB Standardansicht
- MATLAB als Taschenrechner
- MATLAB Programmierung
- Anwendung in der Regelungstechnik

Teil 2: Einführung in Simulink

- Was ist Simulink?
- Vorgehen und Bibliotheken
- Beispiele und Aufgaben
- Subsysteme
- MATLAB Funktionen in Simulink

Übungsaufgaben

Übertragungsfunktion eines SISO-Systems:

$$G(s) = \frac{Z(s)}{N(s)} = \frac{z_m s^m + z_{m-1} s^{m-1} + \cdots + z_1 s + z_0}{n_n s^n + n_{n-1} s^{n-1} + \cdots + n_1 s + n_0} = \frac{\text{Zählerpolynom}}{\text{Nennerpolynom}}$$

$$G_1(s) = \frac{s+2}{s^2+5s+4}$$

```
>> G1 = tf([1 2],[1 5 4])
```

Pol-Nullstellen-Form eines SISO-Systems:

$$G(s) = K \cdot \frac{\prod_{j=1}^{m}(s - n_j)}{\prod_{i=1}^{n}(s - p_i)}$$

$$G_2(s) = \frac{2}{(s-1)(s+2)}$$

Pole, Nullstellen, Systemverstärkung

```
>> G2 = zpk([],[1 -2],[2])
```

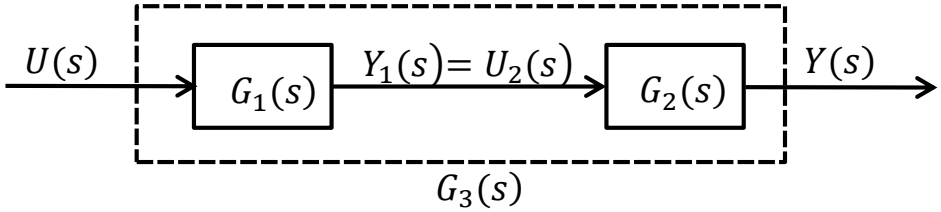
Zustandsraumdarstellung und Frequenzgangdaten: `ss`, `frd`

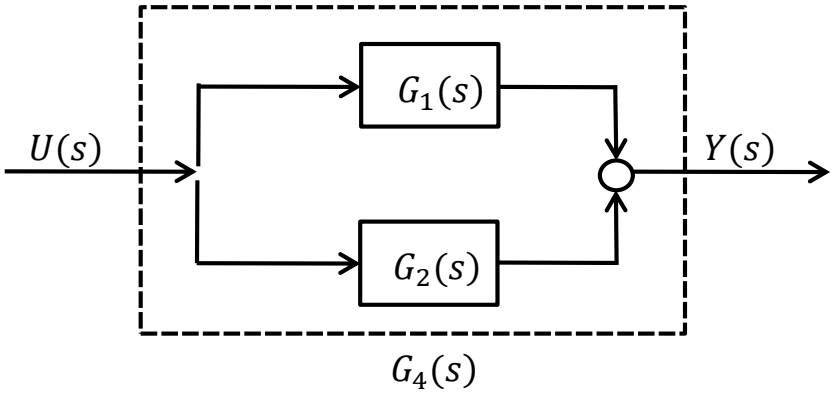
PN-Plan:

```
>> pzmap(G1);                    % PN-Grafik
>> [p,z] = pzmap(G1);            % Vektoren der Pole und Nullstellen
```

Reihen- oder Parallelschaltung:

```
>> G3 = series(G1,G2)

          2 (s+2)
----------------------
(s-1) (s+1) (s+2) (s+4)

>> G4 = parallel(G1,G2)

(s+0.522) (s^2 + 4.478s + 7.662)
--------------------------------
    (s+4) (s+2) (s+1) (s-1)
```



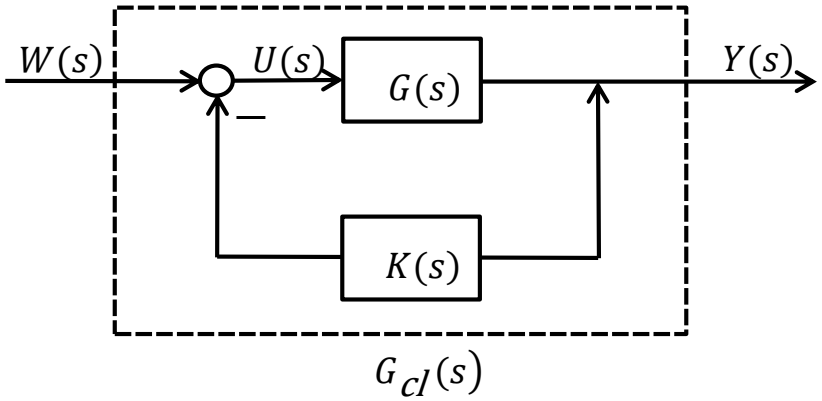


Kürzen von Pol- und Nullstellen

```
>> G5 = minreal(G3)          % Kürzen von Null- und Polstellen
         2
-----------------
(s-1) (s+1) (s+4)
```

geschlossener Kreis:

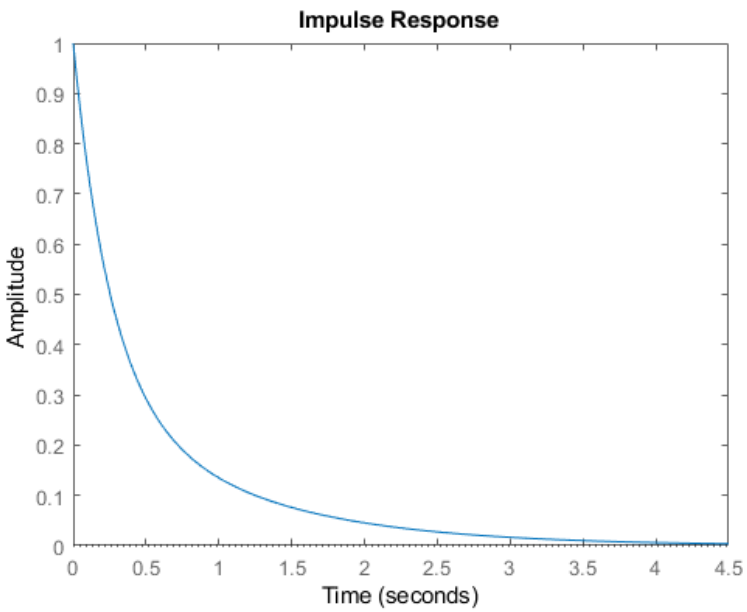
```
>> K=tf([1],[1 1])
  1
-----
s + 1

>> G_cl = feedback(G1,K)
   s^2 + 3 s + 2
----------------------
s^3 + 6 s^2 + 10 s + 6
```



Anwendung in der Regelungstechnik

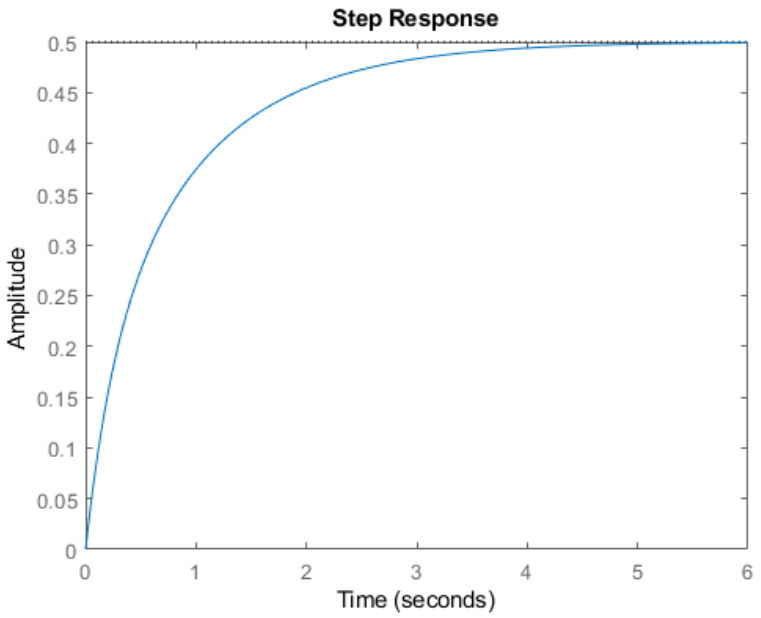
Impulsantwort (Gewichtsfunktion):

```
>> impulse(G1);
```



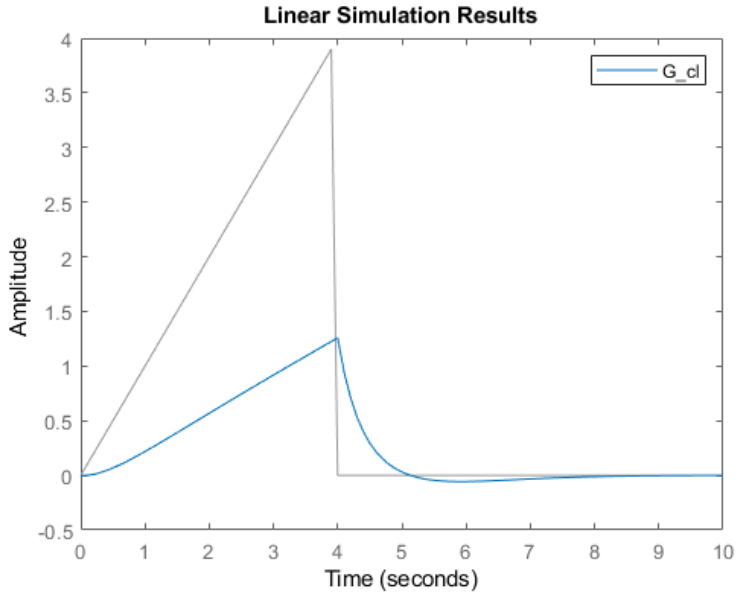
Sprungantwort (Übergangsfunktion):

```
>> step(G1);
```



Antwort auf beliebige Eingangssignale:

```
>> t = [0:0.1:10];
>> u = t;
>> u(41:101) = 0;
>> lsim(G_cl,u,t)      % linear simulation
```



Anwendung in der Regelungstechnik

Aufgaben:

Gegeben sind die beiden Übertragungsfunktionen

$$G_1(s) = \frac{s+2}{s^2+5s+4} \qquad G_2(s) = \frac{s+2}{s^2+0.5s+1}$$

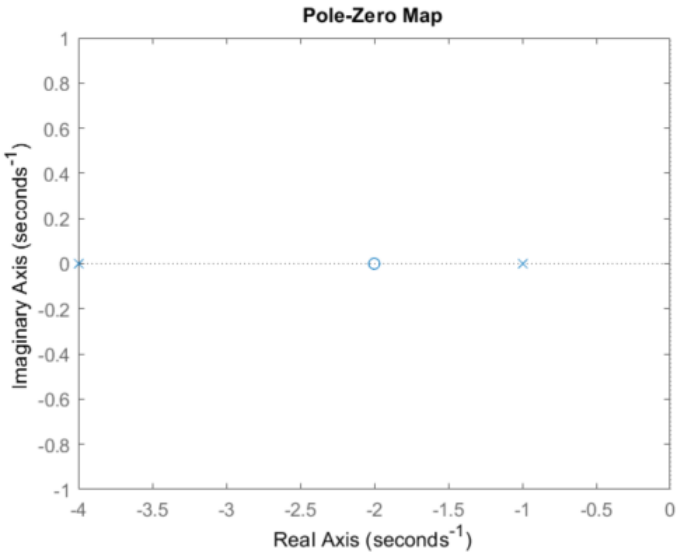
- Erstelle einen Pol-Nullstellen-Plot beider Übertragungsfunktionen
- Erzeuge die Impuls- und Sprungantworten

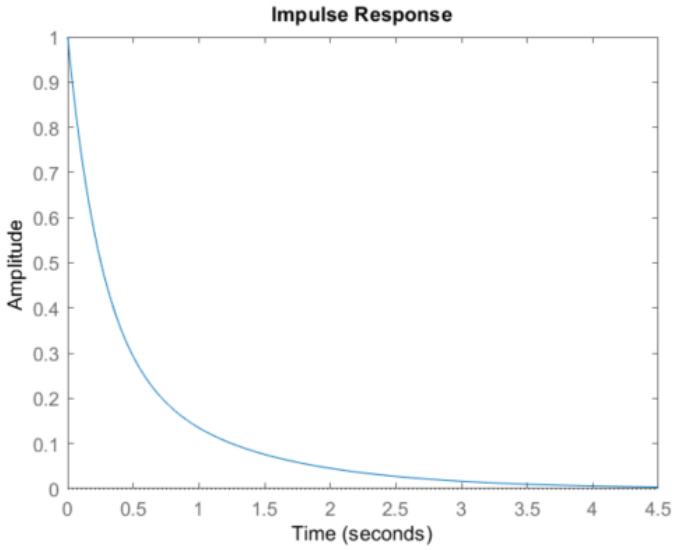
- Erzeuge jeweils eine Reihen- und Parallelschaltung aus beiden Funktionen sowie eine Rückkopplung (geschlossener Kreis) mit $G_2(s)$ in der Rückführung

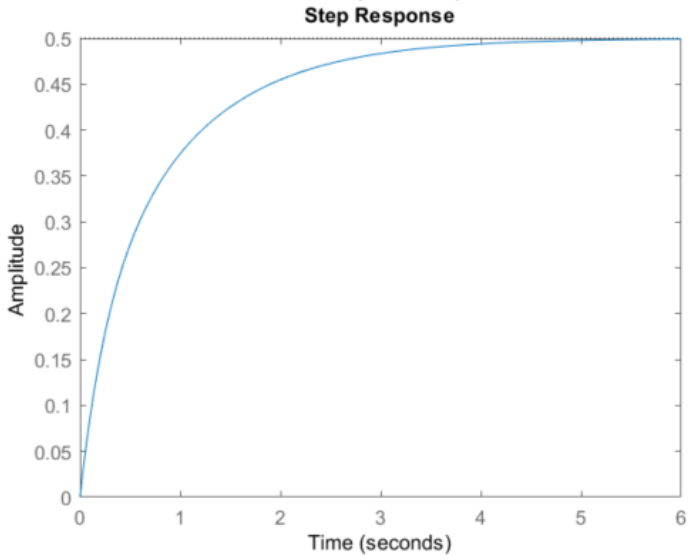
- Erstelle ebenfalls einen Pol-Nullstellen-Plot der drei resultierenden Systeme sowie die Impuls- und Sprungantworten
- Simuliere die beiden Systeme sowie ihre Reihenschaltung mit folgendem Signal $t = [0\colon 0.1\colon 10]$, $u(0s \leq t \leq 5s) = t$ und $u(t > 5s) = 0$

Lösung:

```
>> G1 = tf([1 2],[1 5 4])

>> pzmap(G1)
>> impulse(G1)
>> step(G1)
```

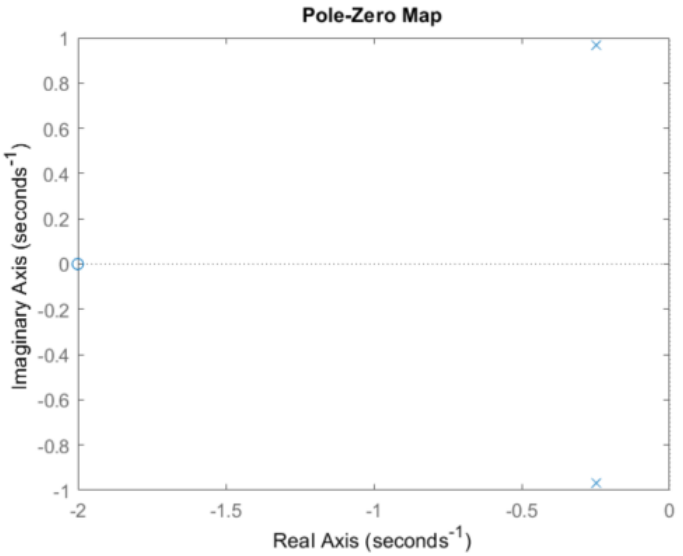


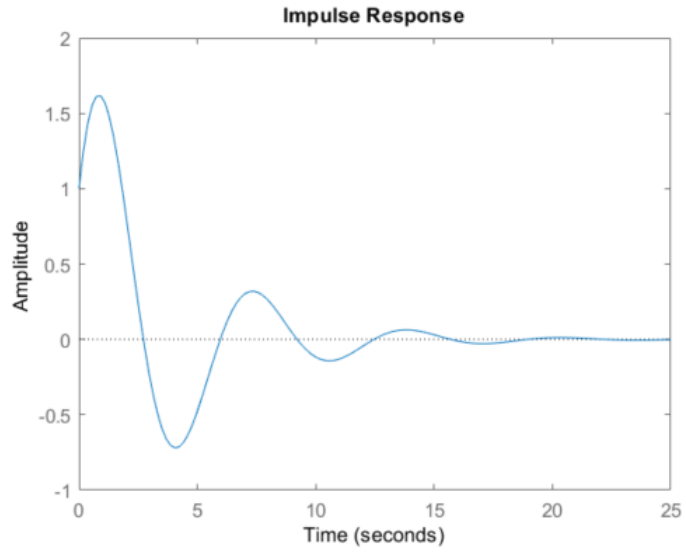


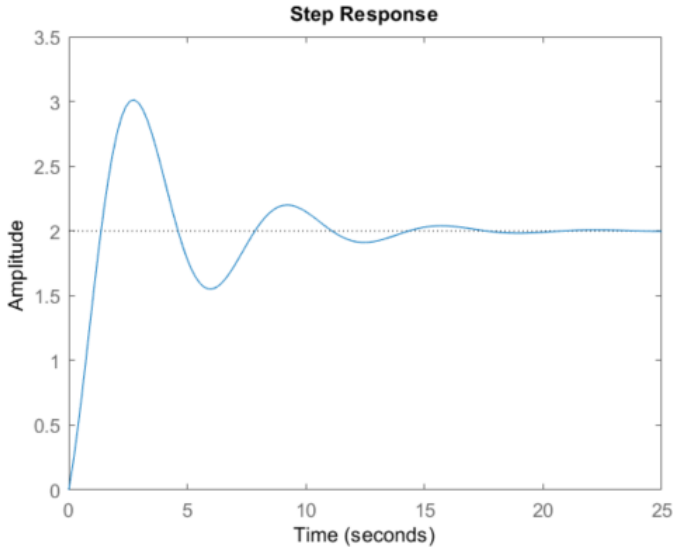


Lösung:

```
>> G2 = tf([1 2],[1 0.5 1])

>> pzmap(G2)
>> impulse(G2)
>> step(G2)
```





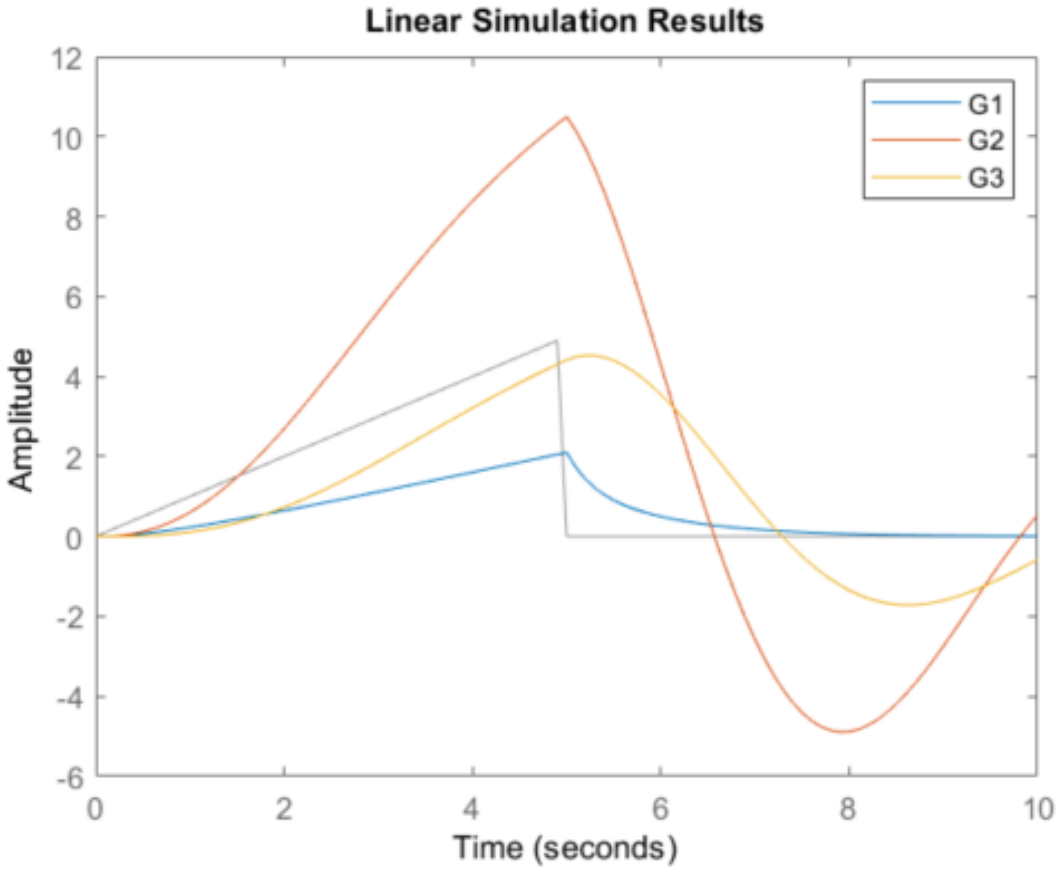


Lösung:

```
>> G3 = series(G1,G2)                    % Reihenschaltung von G1,G2

                s^2 + 4 s + 4
G3 = ---------------------------------
     s^4 + 5.5 s^3 + 7.5 s^2 + 7 s + 4

>> G4 = parallel(G1,G2)                  % Parallelschaltung von G1,G2

          2 s^3 + 9.5 s^2 + 16 s + 10
G4 =  ---------------------------------
      s^4 + 5.5 s^3 + 7.5 s^2 + 7 s + 4

>> G5 = feedback(G1,G2)                  % Rückkopplung von G1 mit G2

            s^3 + 2.5 s^2 + 2 s + 2
G5 = ---------------------------------
     s^4 + 5.5 s^3 + 8.5 s^2 + 11 s + 8
```

Lösung:

```
t = [0:0.1:10];
u = t;
u(51:101) = 0;
lsim(G1,u,t);
hold on
lsim(G2,u,t);
lsim(G3,u,t);
```



Teil 1: Einführung in MATLAB

- Was ist MATLAB?
- MATLAB Standardansicht
- MATLAB als Taschenrechner
- MATLAB Programmierung
- Anwendung in der Regelungstechnik

Teil 2: Einführung in Simulink

- Was ist Simulink?
- Vorgehen und Bibliotheken
- Beispiele und Aufgaben
- Subsysteme
- MATLAB Funktionen in Simulink

Übungsaufgaben

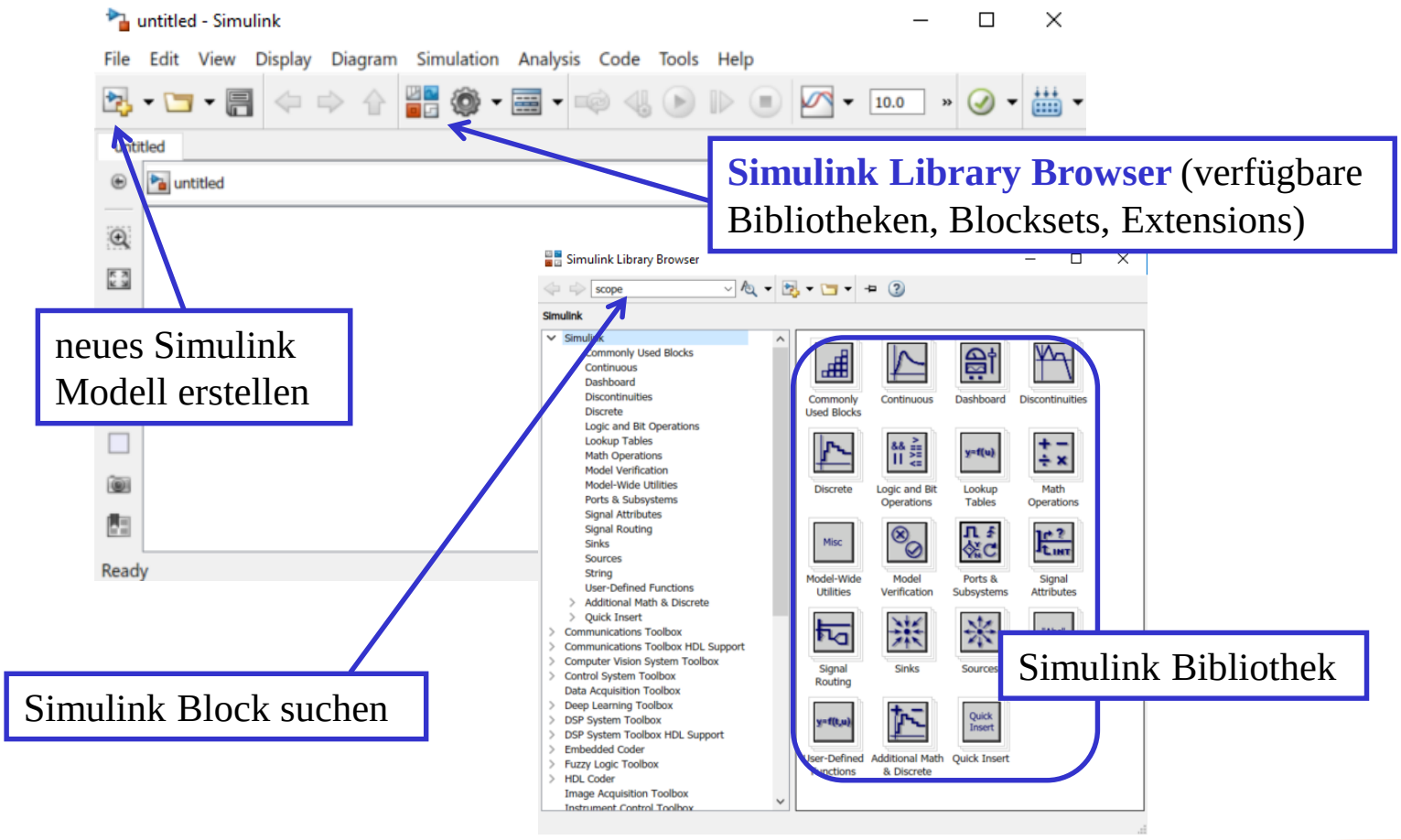
Was ist Simulink?

- Simulink ist eine grafische Benutzeroberfläche zur Modellierung, Simulation und Analyse von Systemen
- Simulink ist eine MATLAB-Toolbox
- Simulink nutzt im Basismodul bereitgestellte numerische Algorithmen

- Simulink Erweiterungen:
  - Blocksets (z. B. Aerospace Blockset, Simscape mit SimDriveline)
  - Extensions (z. B. Real-Time Workshop, Simulink Control Design, Stateflow)

- Typische Anwendungen:
  - Codegenerierung und Rapid Control Prototyping
  - Hardware in the Loop Simulationen

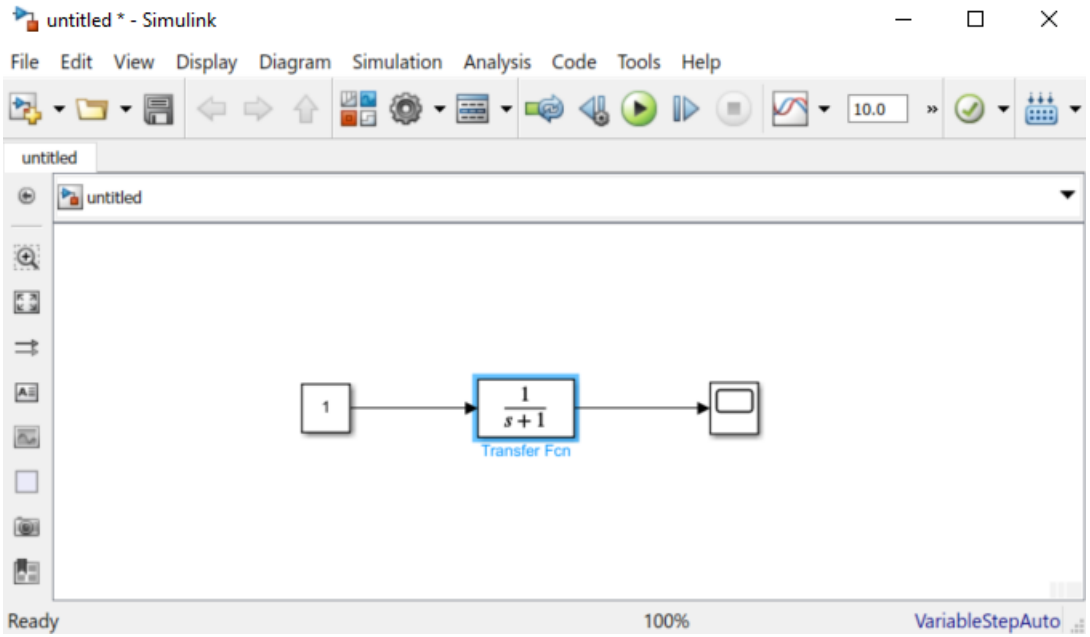
Aufruf im Command Window: `>> simulink`

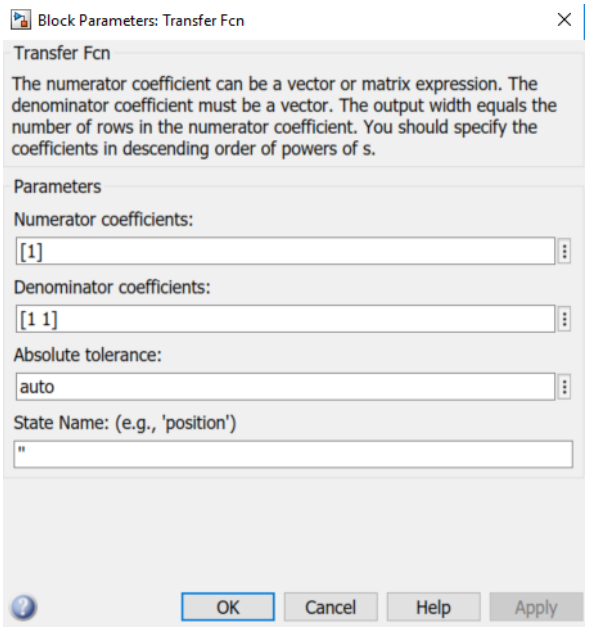


Vorgehen: 1. Simulink-Modell erstellen

Platzieren und Verbinden der Simulink-Blöcke:

- Simulink-Block per Drag-and-Drop aus Bibliothek in Modellfenster ziehen
- Blöcke verbinden: Ausgang des ersten Blocks mit linker Maustaste anklicken und zum Eingang des zweiten Blocks ziehen
- Einstellen der Parameter: Doppelklick auf Simulink-Block.

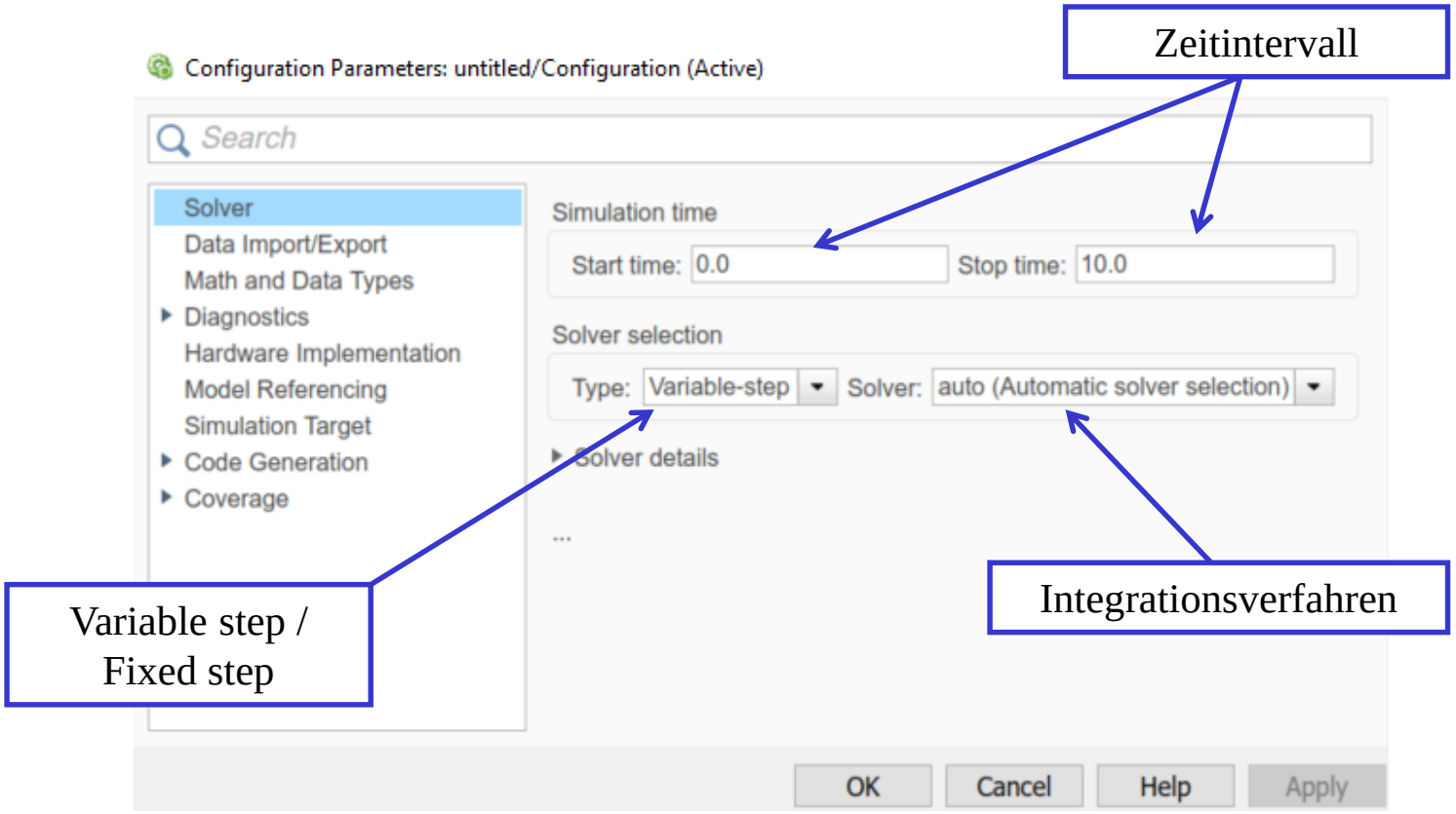




Vorgehen: 2. Simulationsparameter einstellen

grundlegende Einstellungen zur Simulation:

→ Simulation → Model Configuration Parameters



Vorgehen: 2. Simulationsparameter einstellen

Grundlegende Einstellungen zur Simulation:

- Integrationsverfahren mit fester Schrittweite:
  → definierte Rechenzeit, notwendig für Echtzeitanwendungen

- Integrationsverfahren mit variabler Schrittweite:
  → Genauigkeit, Geschwindigkeit, Fehlerüberwachung und Erkennung von Nulldurchgängen.

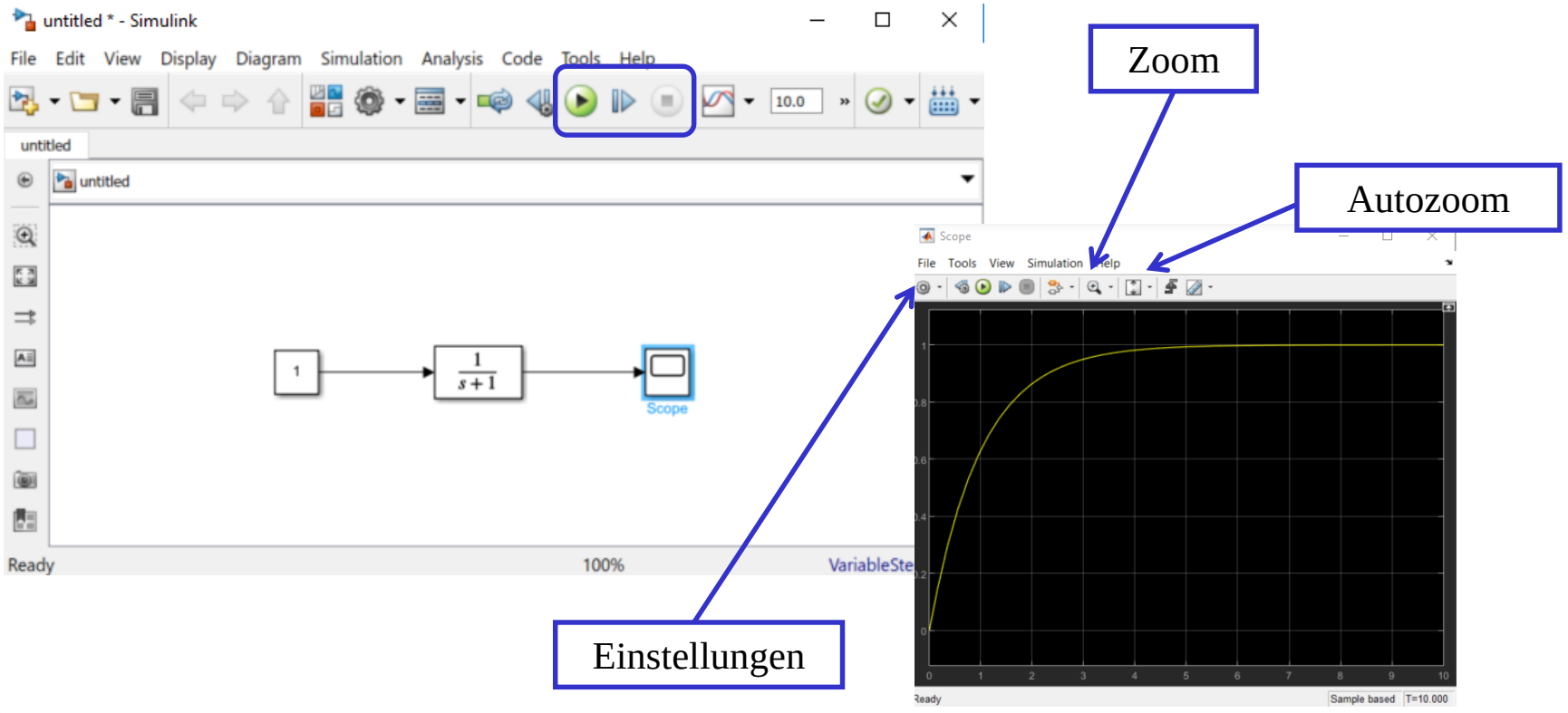
Tipp:

– zunächst Simulation mit Standard-Einstellungen
– bei Bedarf an Systemdynamik anpassen

Vorgehen: 3. Simulieren und Visualisieren

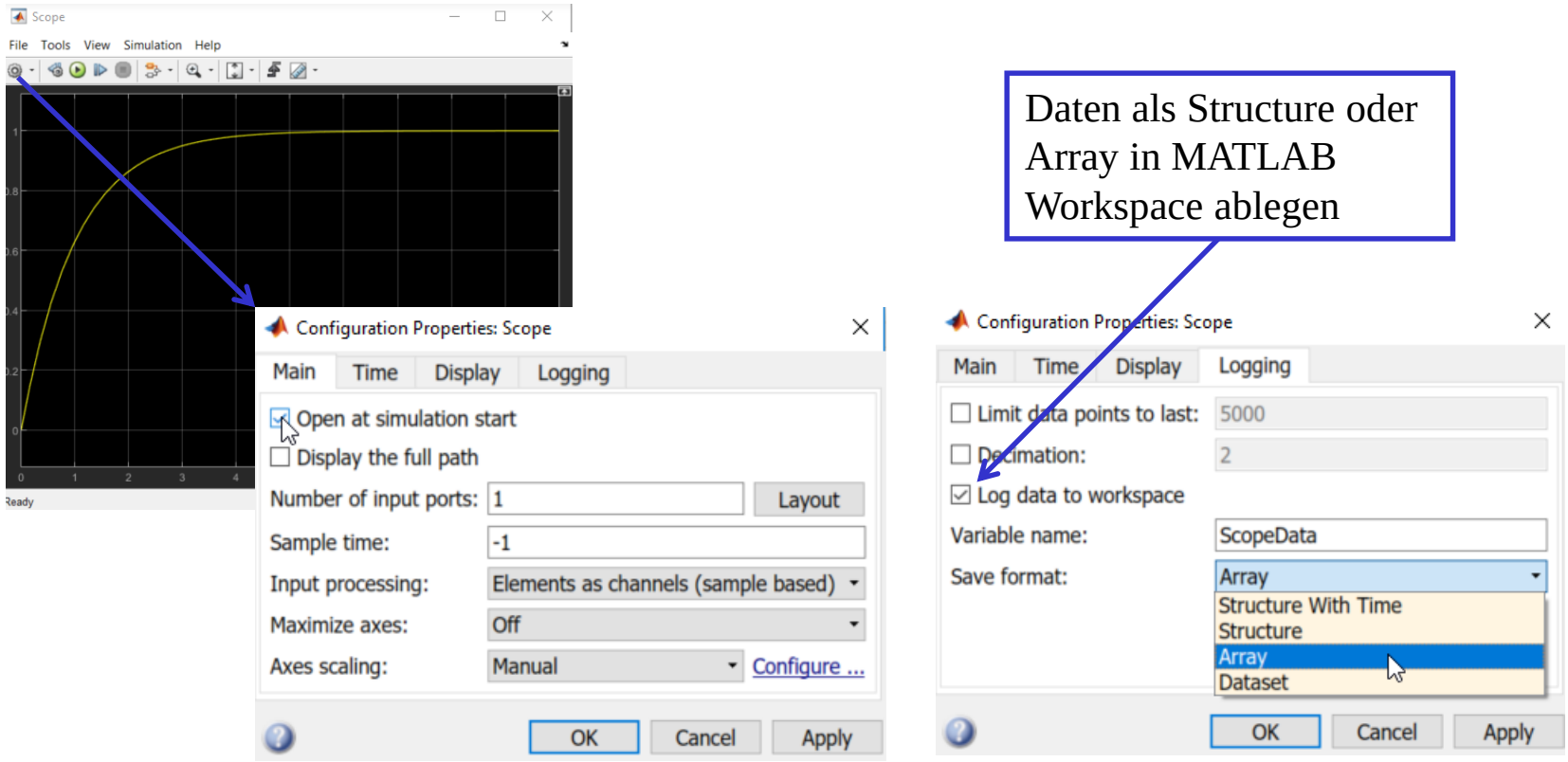
Start der Simulation: ▶ : Start | ⏸ : Pause | ■ : Ende

Visualisierung mit Scope (Doppelklick)



Vorgehen: 4. Auswerten und Weiterverarbeiten

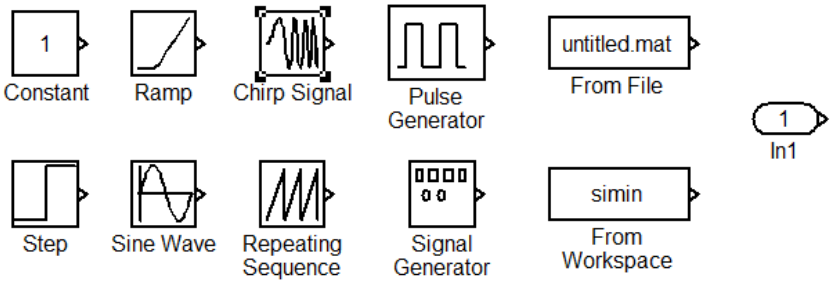
Ergebnisse der Simulation auswerten und weiterverarbeiten:



Wichtige Simulink Bibliotheken

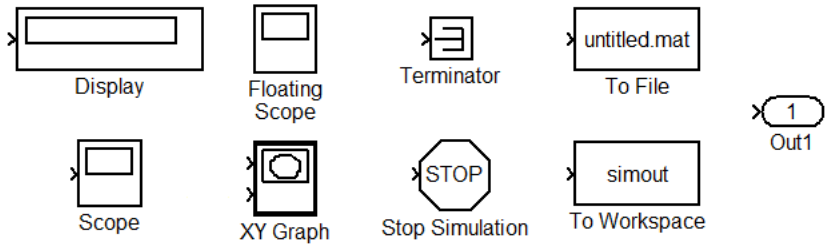
Sources:

- Eingangssignale erzeugen
- Einlesen von Daten aus dem MATLAB Workspace
- Einlesen von Daten aus Dateien



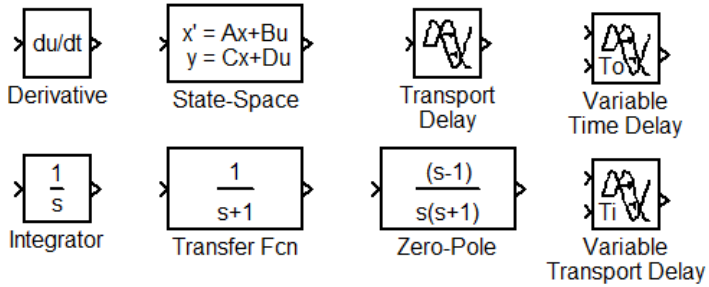
Sinks:

- Visualisieren von Signalen
- Schreiben von Daten auf den MATLAB Workspace
- Schreiben von Daten in Dateien



Continuous:

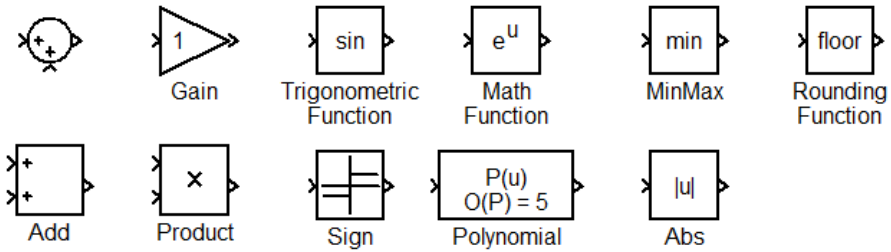
- zeitkontinuierliche Systeme
- Totzeiten



Weitere Simulink Bibliotheken, Blocksets

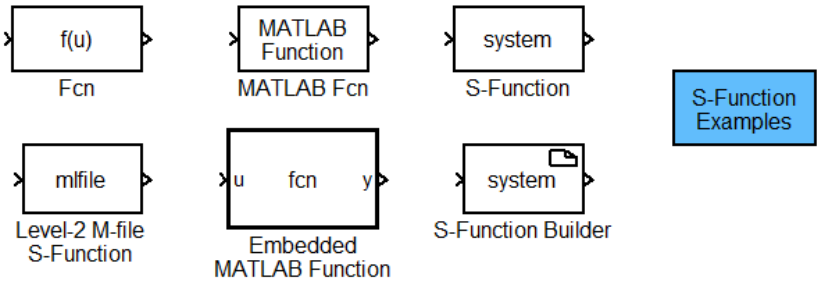
Bibliothek Math Operations:

- Arithmetische Operationen
- Mathematische und trigonometrische Funktionen



Bibliothek User Defined Functions:

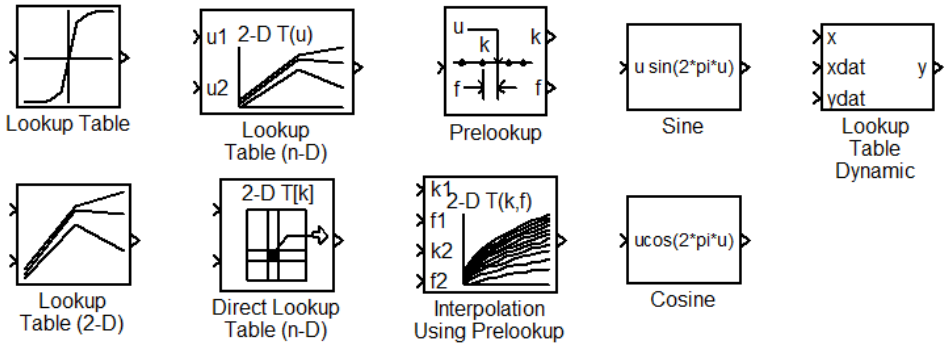
- Frei programmierbare Funktionen
- S-Functions



Weitere Simulink Bibliotheken, Blocksets

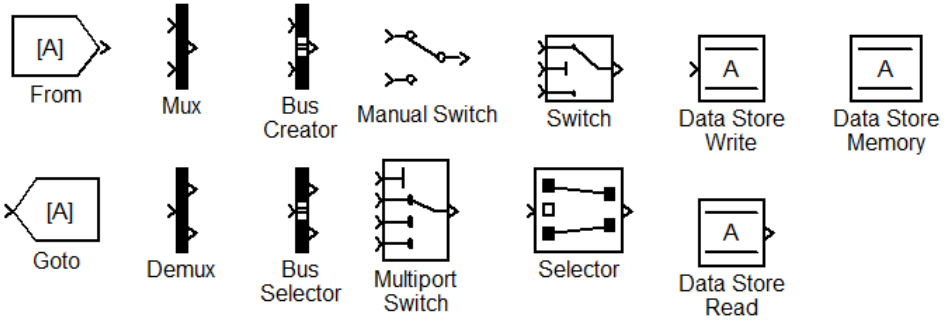
Bibliothek Lookup Tables:

Approximation von Kennlinien und Kennfeldern mit diskreten Werten und unterschiedlichen Interpolationsverfahren



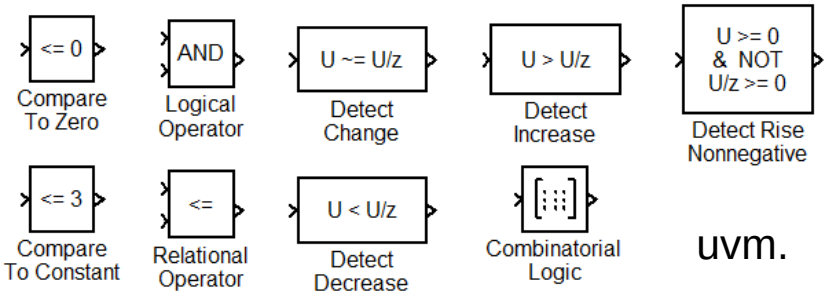
Bibliothek Signal Routing:

- Verknüpfung und Auswahl von Signalen
- Datenspeicher-Management



Bibliothek Logic and Bit Operations:

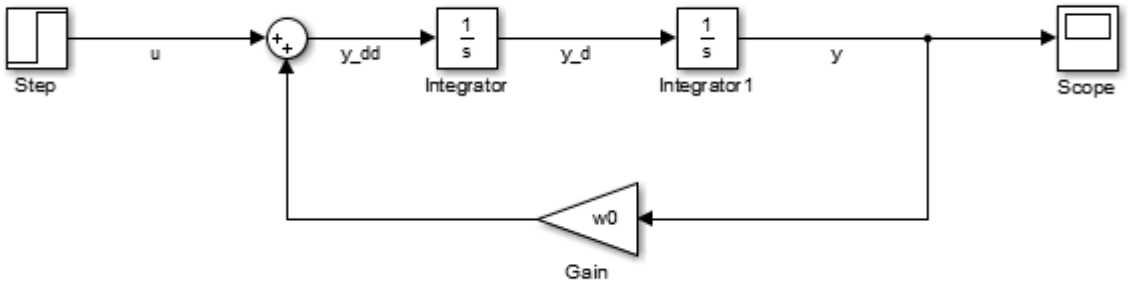
- Logische Operationen
- Operationen auf Bitebene,
- Signalüberwachung

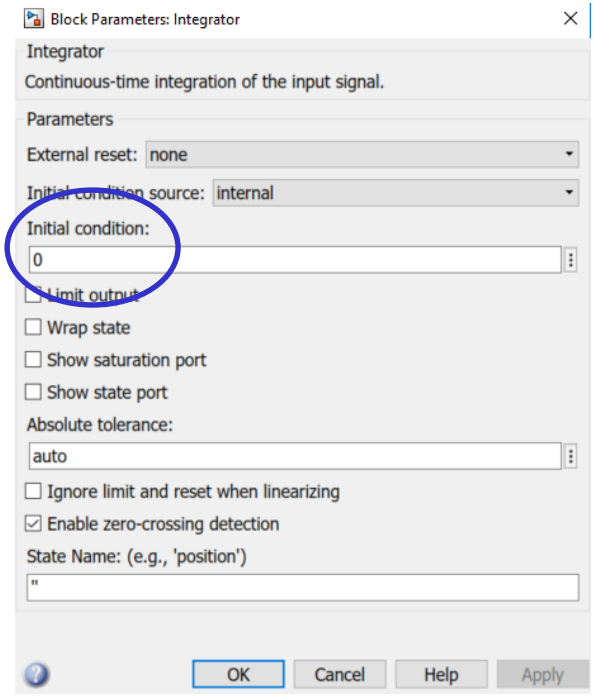


Beispiel: Simulation einer DGL

Es soll die folgende DGL in Simulink programmiert werden:

$$\ddot{y}(t) - \omega_0 \cdot y(t) = u(t)$$

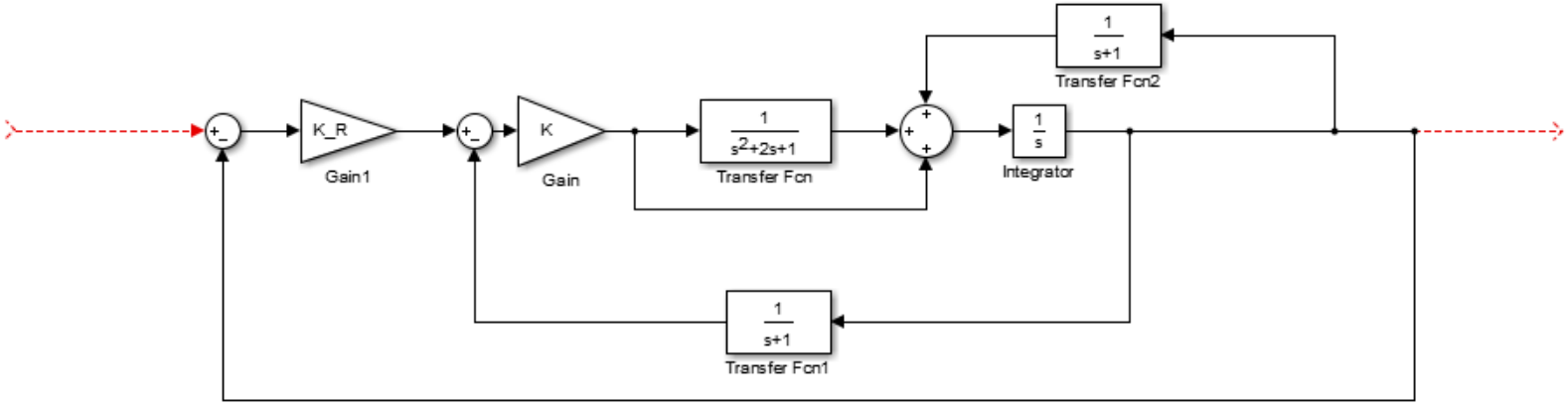




- Anfangswerte für die zeitabhängigen Größen werden am jeweiligen Integrator vorgegeben
- Eingegebener Wert entspricht dem Anfangswert des Ausganges

Beispiel: Simulation eines Regelkreises

Beispiel eines Regelkreises:



- einfaches Aufbauen von Blockschaltbildern
- kein vorheriges Zusammenfassen notwendig
- direkte Simulation und damit Testen von Reglern möglich
- schnelles anpassen bzw. ändern

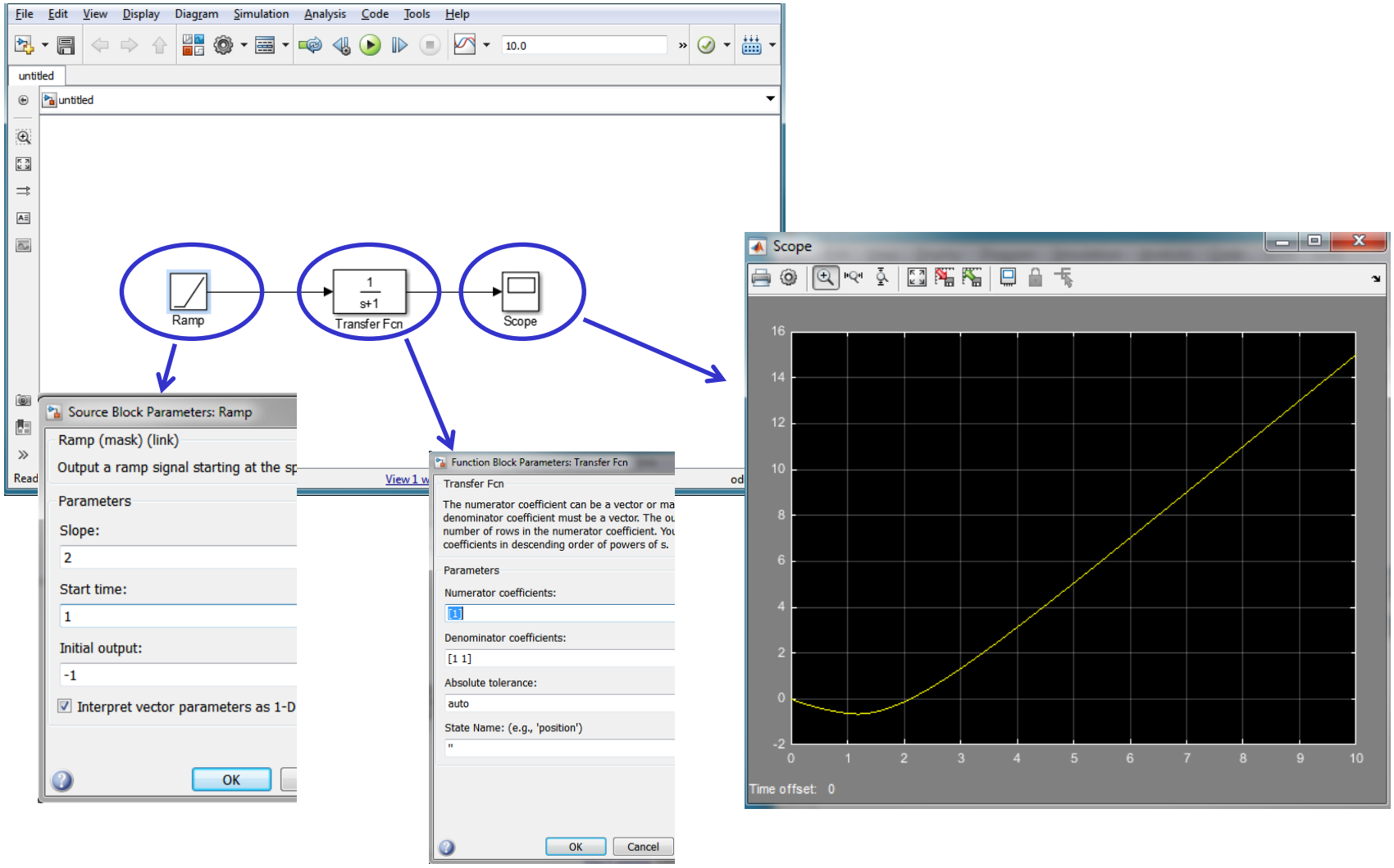
Aufgaben

Aufgaben:

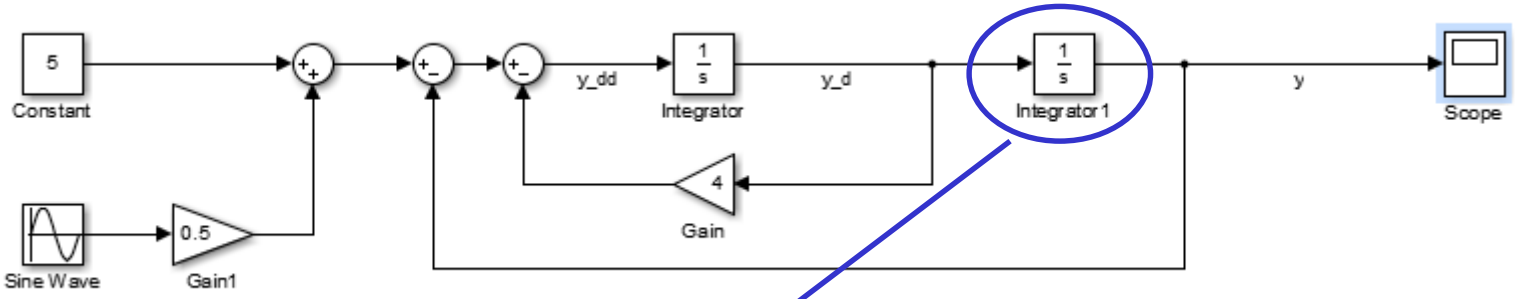
- Erstelle zwei Simulink-Modelle:
  1. Strecke: G = 1/(1+s)
     - Rampe: $u(0) = -1$, Rampe beginnt z. Zt. 1 und hat Anstieg 2
     - Visualisierung des Ausgangs
  2. Differentialgleichung $\ddot{y}(t) + 4 \cdot \dot{y}(t) + y(t) = u_1(t) + 0.5 \cdot u_2(t)$

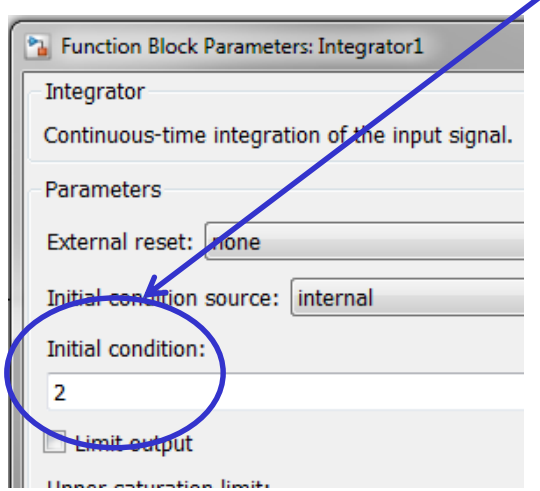
     Eingangssignale:
     - $u_1 = 5$
     - $u_2 = \sin(t)$
     - Startwerte: $y(0) = 2, \ \dot{y}(0) = \ddot{y}(0) = 0$
     - Visualisierung des Ausgangs
- Simulationsparameter:
  - Zeitintervall [0,10]
  - restliche Einstellungen auf den Standardeinstellungen belassen
- Simulieren, Visualisieren

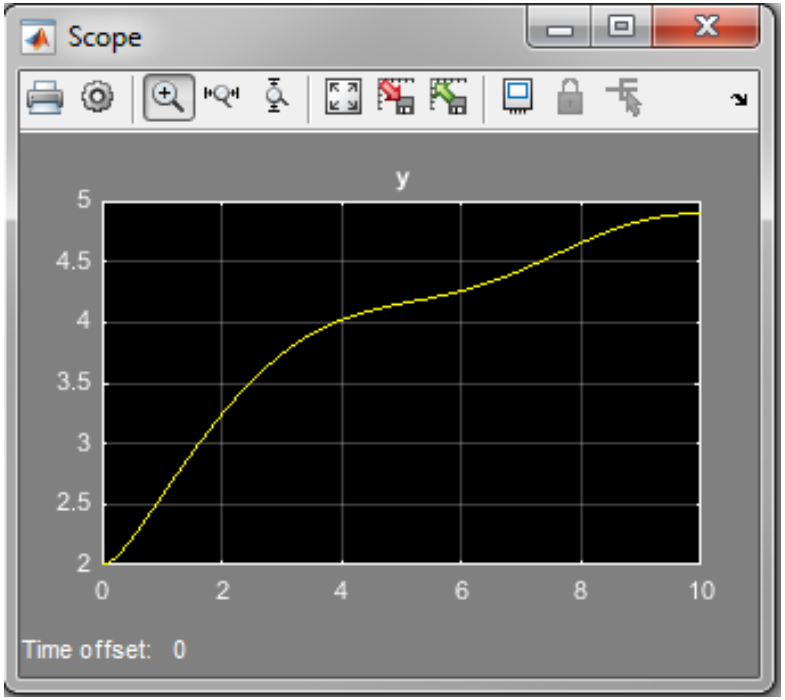
Lösung


File Edit View Display Diagram Simulation Analysis Code Tools Help
10.0
untitled
Ramp
1
s+1
Transfer Fcn
Scope
Source Block Parameters: Ramp
Ramp (mask) (link)
Output a ramp signal starting at the sp
Parameters
Slope:
2
Start time:
1
Initial output:
-1
Interpret vector parameters as 1-D
OK
Function Block Parameters: Transfer Fcn
Transfer Fcn
The numerator coefficient can be a vector or ma
denominator coefficient must be a vector. The ou
number of rows in the numerator coefficient. You
coefficients in descending order of powers of s.
Parameters
Numerator coefficients:
[1]
Denominator coefficients:
[1 1]
Absolute tolerance:
auto
State Name: (e.g., 'position')
''
OK
Cancel
Scope
16
14
12
10
8
6
4
2
0
-2
0 1 2 3 4 5 6 7 8 9 10
Time offset: 0

$$\ddot{y}(t) + 4 \cdot \dot{y}(t) + y(t) = u_1(t) + 0.5 \cdot u_2(t)$$

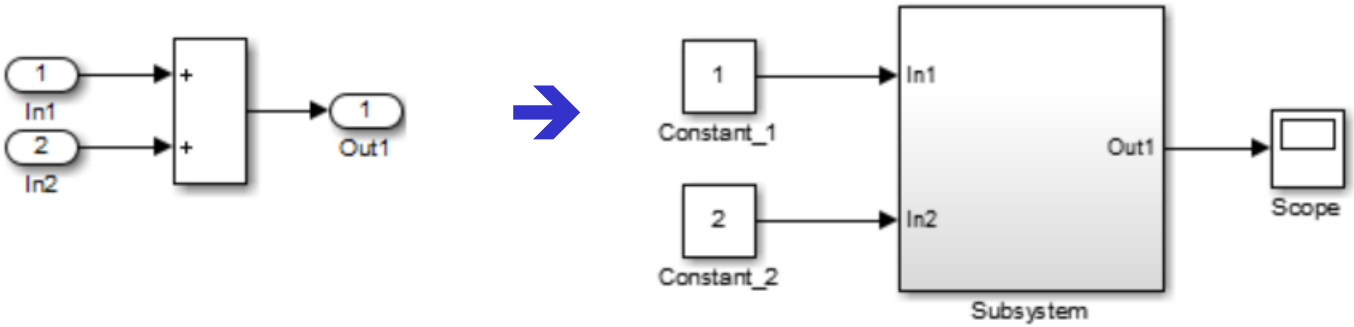






Verwendung von Subsystemen:

- mehrere Blöcke können zu einem neuen Simulink-Block zusammengefasst werden
- Ports & Subsystems library → Subsystem-Block



- Übersichtlichkeit
- Effizienz (Wiederverwendung getesteter Strukturen...)
- Parametrisierung von komplexen Teilsystemen mit rechter Maustaste
- eigene Bibliothek mit Subsystemen, die oft benötigt werden

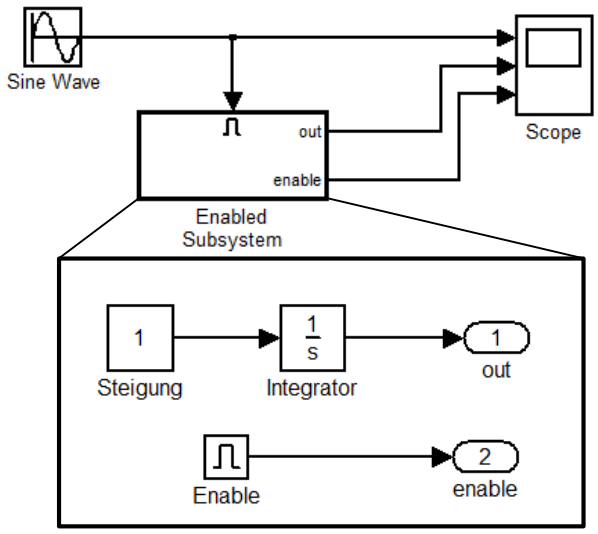
Verwendung von Subsystemen:

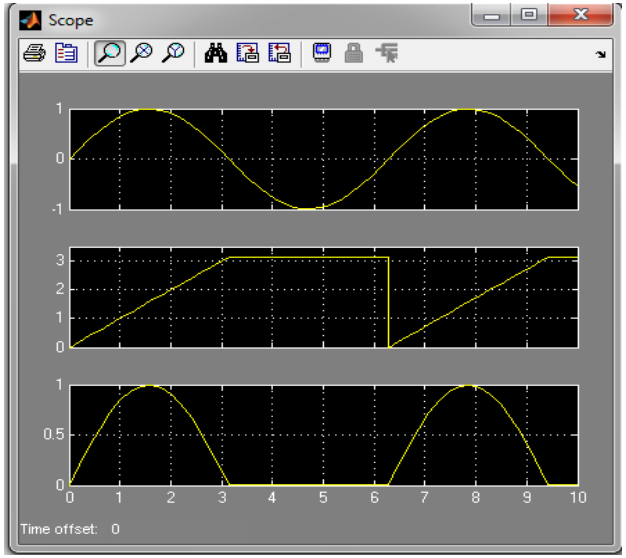
Ausführung des Subsystems kann durch Steuersignal bestimmt werden, abhängig von Bedingungen:

→ *Enabled Subsystems, Triggered Subsystems*

→ *Control Flow Subsystems, für Schleifen und Verzweigungen*

Beispiel: $y_1(t) = sin(t), y_2(t) = t \cdot 1(t)$ für $sin(t) > 0$





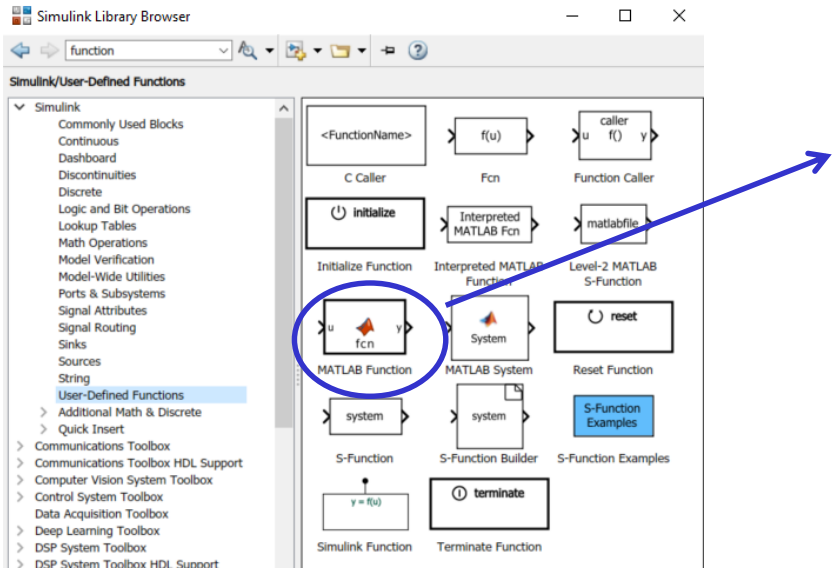
MATLAB-Functions

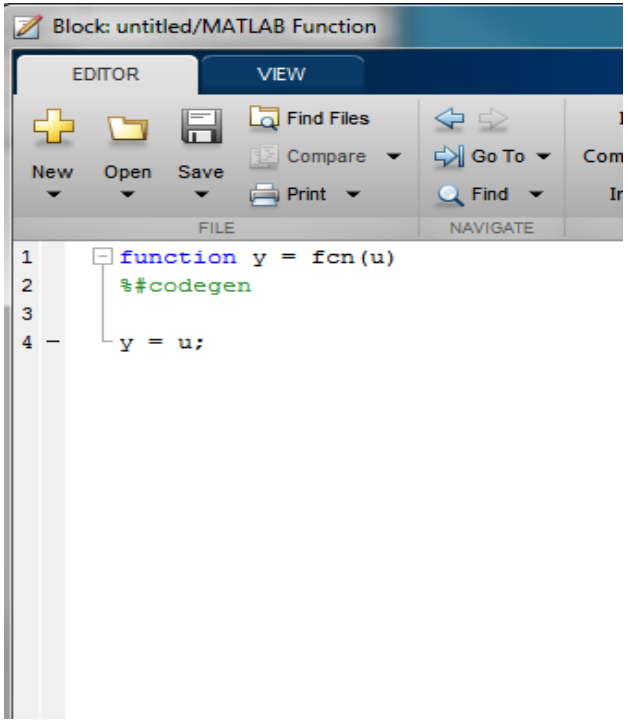
MATLAB Function:

- frei programmierbarer Simulink-Block
- Programmierung analog zur `function` in MATLAB
- ermöglicht das Aufrufen von bereits bestehenden MATLAB-Funktionen
- einfachere Programmierung als mit Simulink-Blöcken.
- geringere Rechenzeit als mit Simulink-Blöcken.
- bei komplizierten Rechnungen häufig einfacher umsetzbar
- weitere Anwendung: Systembeschreibung liegt als Code vor (z. B. von einem Industriepartner), aber nicht als Modell aus elementaren Blöcken

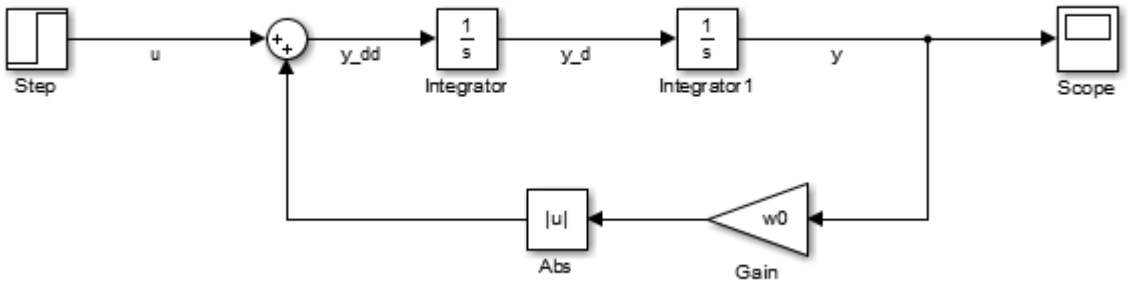
Wie erstellet man einen MATLAB-Function-Block?

- Block „MATLAB-Function“
- User Defined Functions
- Erstellung analog zur Funktion in MATLAB
- Definition von Ein- und Ausgängen
- Programmieren von beliebigen Rechenoperationen

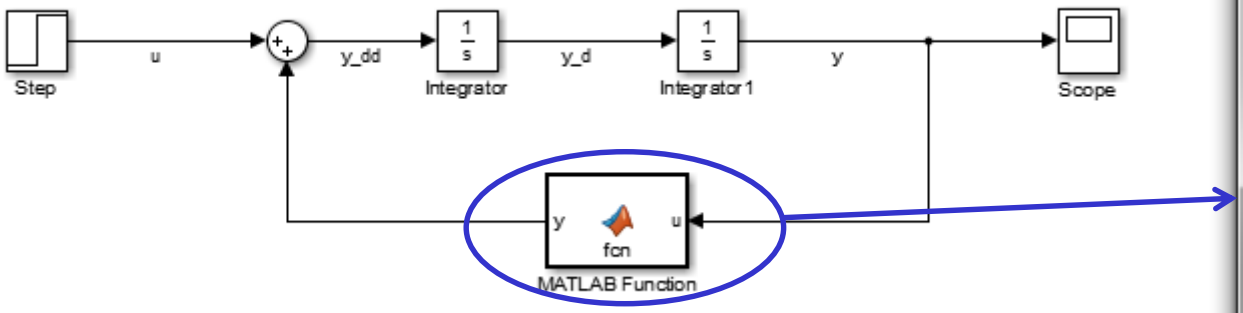


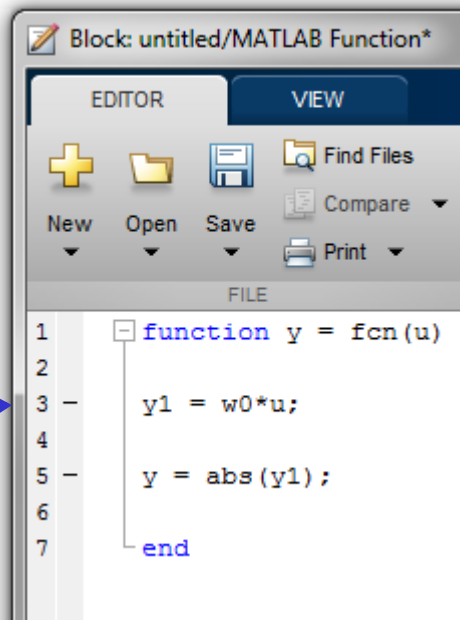


MATLAB-Functions: Beispiel



Ersetzen der Rückführung
durch MATLAB-Function





<u>Aufgabe 1:</u> „Schiefer Wurf“

Simulieren Sie die Flugbahn einer punktförmigen Masse M unter Einfluss der Schwerkraft, die bei $t = 0$ im Punkt $(0, h)$ mit der Geschwindigkeit $(v_{x0}, v_{z0})^T$ startet.

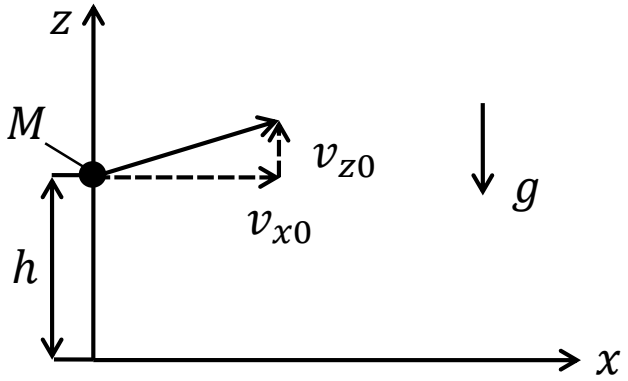


Abbildung 1: Schiefer Wurf

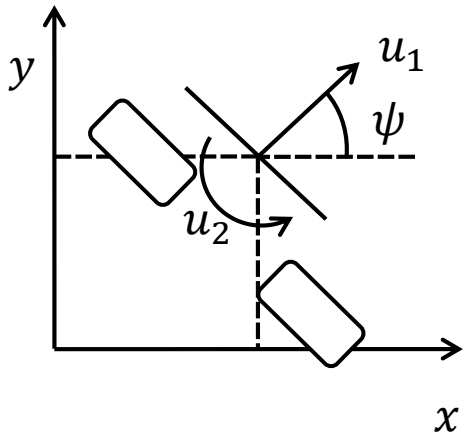
Tipp: $\dot{x} = v_{x0}$,
$\ddot{z} = -g.$

Vernachlässigen Sie alle Reibungs- und Kontakteffekte.

$M = 5kg,\ h = 5m,\ g = 9.81\ \frac{m}{s^2},\ (v_{x0}, v_{z0}) = (15{,}10)\ \frac{m}{s}$

<u>Aufgabe 2:</u> Fahrdynamik eines Roboters

Es wird das folgende Modell eines Roboters betrachtet:



$$\dot{x} = u_1 \cdot cos(\psi),$$

$$\dot{y} = u_1 \cdot sin(\psi),$$

$$\dot{\psi} = u_2.$$

$$(x_0, y_0, \psi_0)^T = (0,0,0)^T.$$

Man wähle $u_1 = 5, \quad u_2 = 0.1 \cdot \sin(t)$