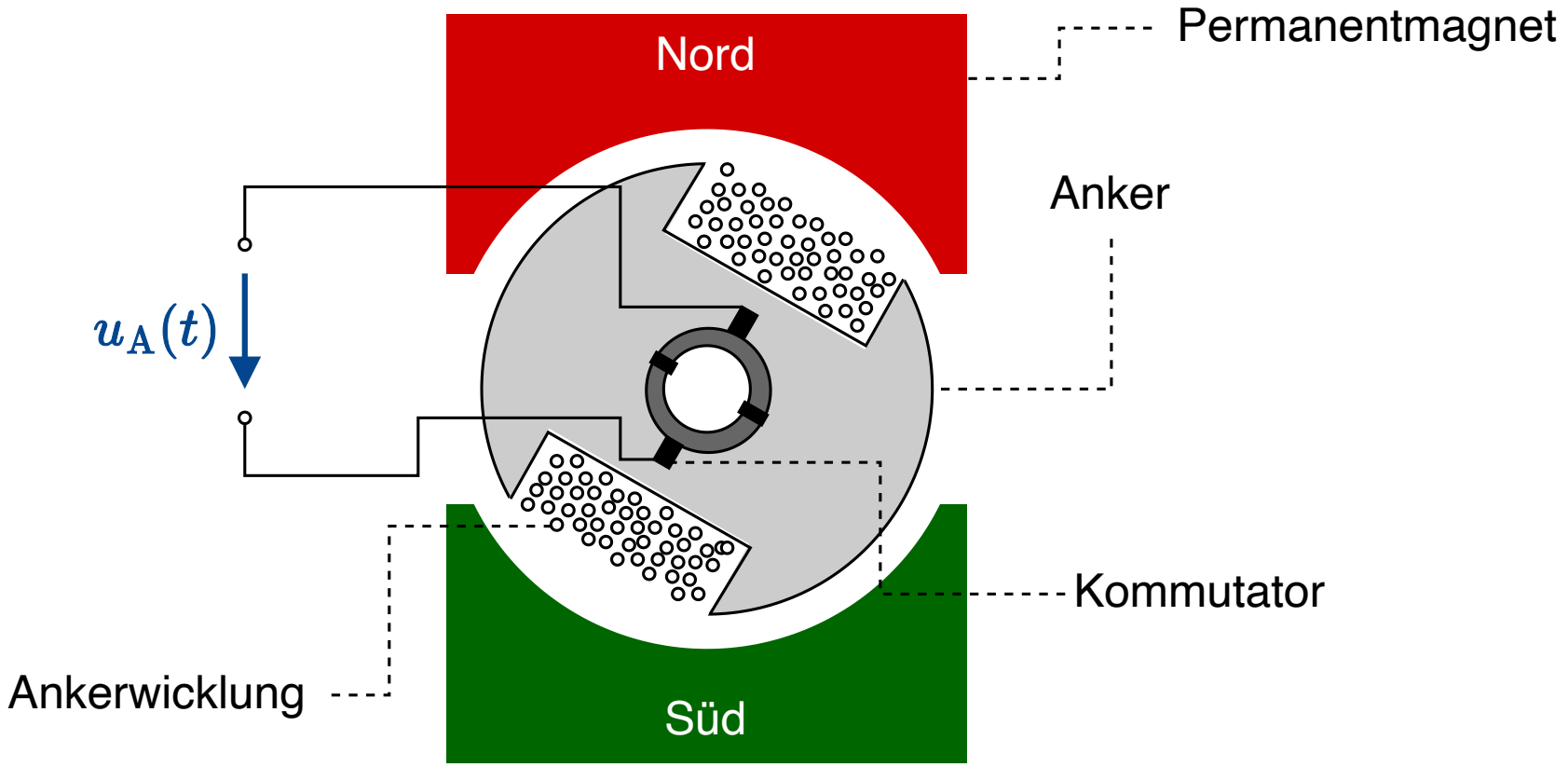
Modellierung verschiedener technischer Systeme

Modellierung eines Gleichstrommotors

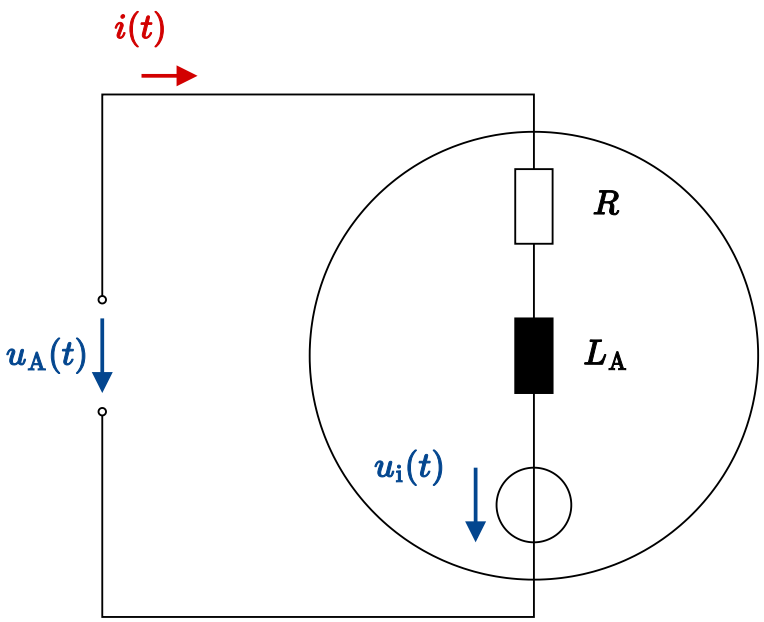
Modellierung eines permanenterregten Gleichstrommotors

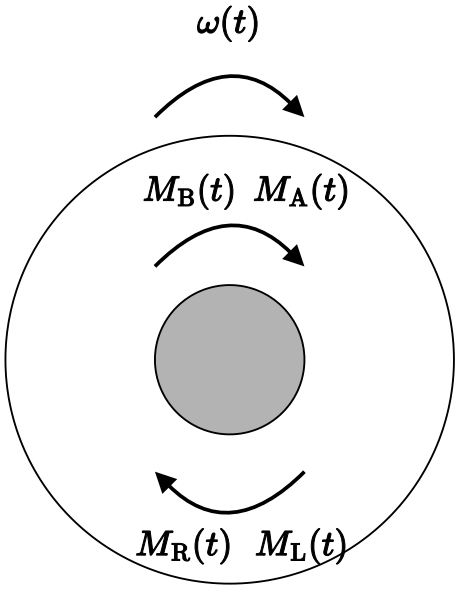
Betrachtung eines permanenterregten Gleichstrommotors mit Kommutator

Die Anker-Welle wird mit einem Drehmoment $M_{\mathrm{L}}(t)$ belastet



Ersatzschaltbild der Gleichstrommaschine





$R = 8\,\Omega$

$k_G = 0{,}015\,\mathrm{Vs}$

$J = 9 \cdot 10^{-7}\,\mathrm{kg \cdot m^2}$

$L_A = 100\,\mathrm{mH}$

$M_R = 0{,}0018\,\mathrm{Nm}$

$k_V = 9 \cdot 10^{-6}\,\mathrm{Nms}$

Modellierung

1. Zeichnen eines aussagekräftigen Schemas => *siehe Bild*

2. Ermittlung der Zustandsgrößen

$$i(t) = \frac{1}{L_\mathrm{A}} \int u_L(t)\,\mathrm{d}t \qquad \omega(t) = \frac{1}{J} \cdot \int M_\mathrm{B}(t)\,\mathrm{d}t \qquad \left(\varphi(t) = \int \omega(t)\,\mathrm{d}t\right)$$

3. Bilanzgleichungen

$$u_\mathrm{A}(t) = u_R(t) + u_L(t) + u_i(t) \qquad M_\mathrm{B}(t) = M_\mathrm{A}(t) - M_\mathrm{L}(t) - M_\mathrm{R}(t)$$

4. Statische Grundbeziehungen

$$u_\mathrm{i}(t) = k_\mathrm{G} \cdot \omega(t) \qquad u_R(t) = R \cdot i(t) \qquad M_\mathrm{A} = k_\mathrm{G} \cdot i(t)$$

5. Aufbau des Blockschaltbildes => *mittels Simulink*

Aufgaben I

1. Bauen Sie das Modell des Gleichstrommotors auf und stellen Sie folgende Größen dar
    1. Ankerspannung $u_\mathrm{A}(t)$
    2. Drehzahl $w(t)$ in der Einheit $\frac{\mathrm{Umdrehungen}}{\mathrm{min}}$
2. Simulieren Sie folgende Szenarien
    - keine Belastung des Motors und Eingangsspannung $u_\mathrm{A}(t) = 0\,\mathrm{V}$
    - Sprung der Eingangsspannung auf $u_\mathrm{A}(t) = 4\,\mathrm{V}$
    - mechanische Belastung des Motors mit $M_\mathrm{L} = 0{,}2\,\mathrm{Ncm}$
3. Ergänzen Sie folgende Anzeigen
    - elektrischen Leistungsaufnahme des Motors
    - Wirkungsgrad

Simulink Funktionen I

Probieren Sie beim Lösen der Aufgabe folgende Simulink Funktionen aus

- *Log-Signals* und *Data Inspector*
- *Subsysteme*
- *Simulation Manager*
- *Data Labels*
- nicht-lineare Kennlinien mit *Look-Up-Tables*

Überlegen Sie sich welchen Einfluss die *Modellarchitektur* besitzt.

Aufgaben II

1. Erweitern Sie das Modell um einen P-Regler und starten Sie mit einer Verstärkung von 1
2. Optimieren Sie händisch die Regelungsverstärkung bis ein akzeptables Verhalten einsetzt
3. Überlegen Sie sich folgende Aspekte Ihres Modells
   - Lassen sich beliebige Ankerspannungen $u_A(t)$ stellen?
   - Macht der Sprung tatsächlich Sinn?

Simulink Funktionen II

- *Compare* Funktion des *Data Inspector*
- *Begrenzer* und *Rate-Limiter*

Aufgaben III

1. Optimieren Sie den P-Regler mit Hilfe des Blocks *Varying PID Controller*.
2. Verbessern Sie den Regler durch einen integralen Anteil.
3. Überprüfen Sie auch die Störimpulsantwort
4. Überlegen Sie Vor- und Nachteile bei diesem Vorgehen im Entwicklungsprozess

Simulink Funktionen III

- *Varying PID Controller*

Aufgaben von Simulationsmodellen

Unterschiedliche Aufgaben benötigen andere Simulationsmodelle

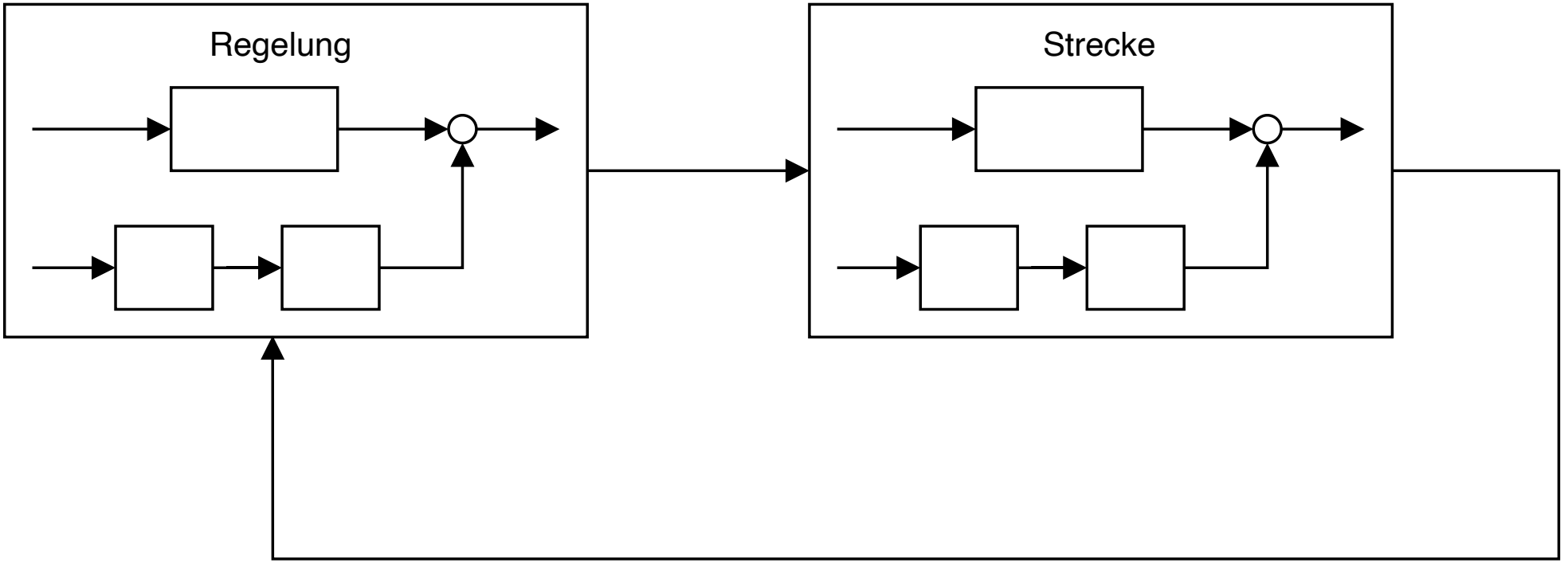
- Entwicklungsmodelle (z.B. für den Entwurf von Reglungen)
- Testmodel
- Integrationsmodell
- Systemmodell

Ziele von numerischen Simulationen

Simulationsstrategien

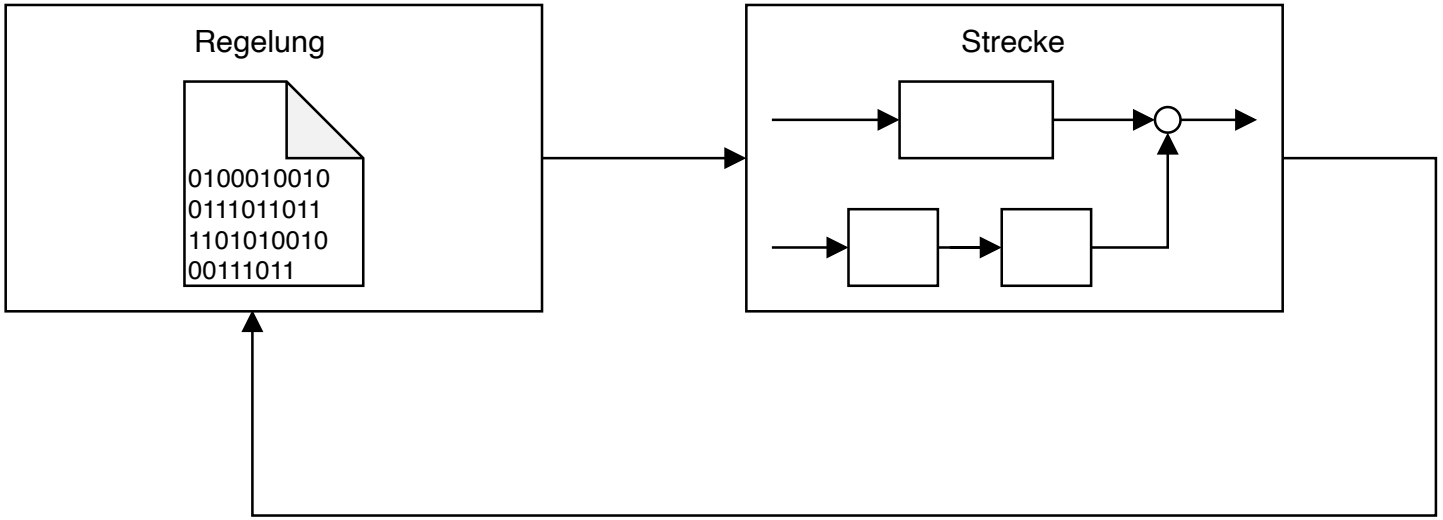
1. Model-in-the-Loop (MIL)
2. Software-in-the-Loop (SIL)
3. Processor-in-the-Loop (PIL)
4. Hardware-in-the-Loop (HIL)

Model-in-the-Loop (MIL)



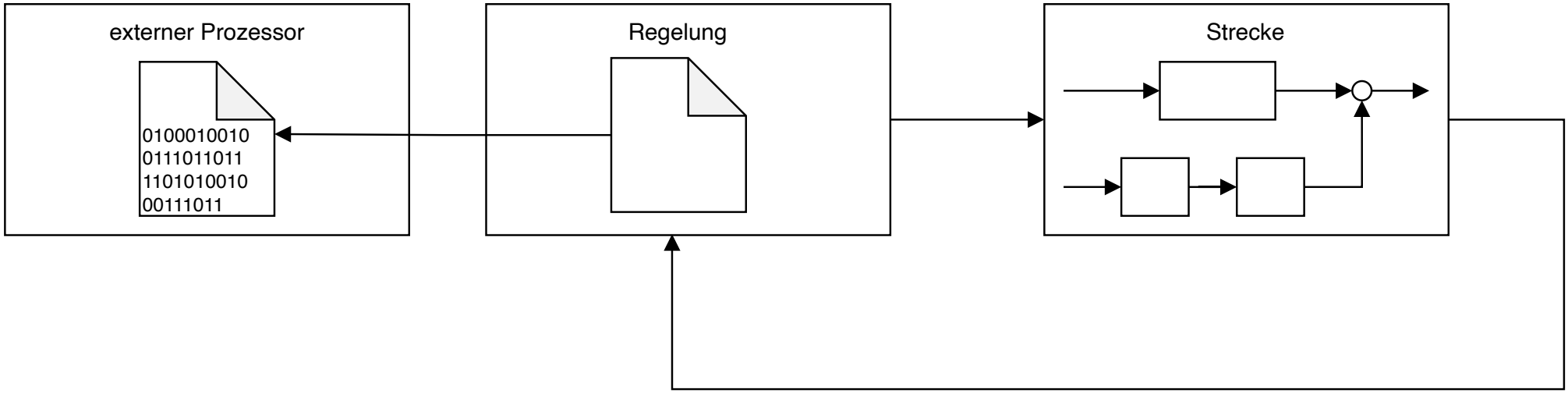
- Umsetzung von Regelung und Strecke in einem abstrakten Modell
- Einfache Möglichkeit zur Verifikation der Anforderungen
- Nächster Schritt: Umsetzung der Regelung auf *Zielsystem*

Software-in-the-Loop (SIL)



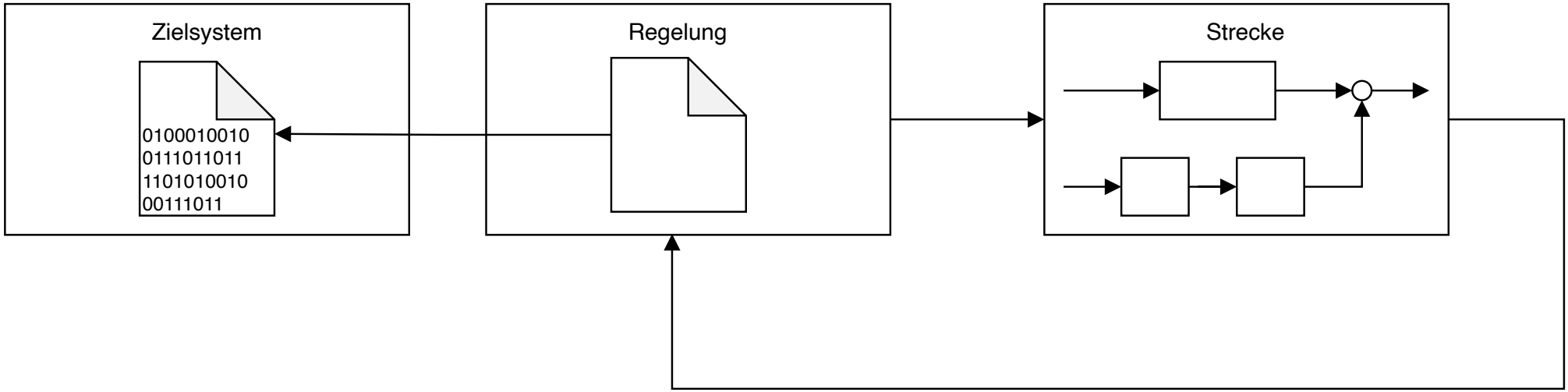
- Software des *Zielsystems* wird für Simulationsmodell aufbereitet
- Problematik:
  - Simulations- und Zielsystem verwenden unterschiedliche Hardware
  - Unterschiedliche Auswirkung auf Performance, Numerische Effekte, Bibliotheken von Drittherstellern
- Möglichkeit in Simulink: *level-2 s-functions*

Processor-in-the-Loop (PIL)



- Ausführung der Regelungssoftware auf gleichem Prozessor wie *Zielsystem*
- Üblicherweise Verwendung eines Entwicklungsboards des Prozessors
- Herausforderung: Integration des Entwicklungsboards in Simulationsumgebung
- Problematik: Unterschiedliche Auswirkung des zeitlichen Verhaltens

Hardware-in-the-Loop (HIL)



- Ausführung der Regelungssoftware auf *Zielsystem*
- Besondere Anforderungen an Simulationsumgebung der Strecke: übleicherweise *Echzeitanforderung*
- Meist nur vereinfachtes Streckenmodell umsetzbar
- Verifikation der Schnittstellund und des zeitlichen Verhaltens

Modellierung dynamischer Systeme in Matlab I

Matlab Transfer Function Objekte I

Gezielte Auslegung des Reglers in Simulink direkt nicht möglich: Analyse der Strecke und des Reglers in Matlab

Anlegen einer Übertragungsfunktion mittels Transfer Function Objects (`tf`-Objekten)

```
>> H = tf('s')

H =

  s

Continuous-time transfer function.
```

Matlab Transfer Function Objekte II

Anlegen eines `tf`-Objektes mit Hilfe von Zähler- und Nennerpolynom

```
>> H = tf([1, 2, 3], [4, 5, 6])

H =

   s^2 + 2 s + 3
  ---------------
  4 s^2 + 5 s + 6

Continuous-time transfer function.
```

Übertragungsfunktion in Pol-Nullstellendarstellung I

Erstellen einer Übertragungsfunktion mittels Pol- und Nullstellen

```
>> H = zpk([1, 2], [3, 4], 5)

H =

  5 (s-1) (s-2)
  -------------
    (s-3) (s-4)

Continuous-time zero/pole/gain model.
```

Übertragungsfunktion in Pol-Nullstellendarstellung II

Umwandlung eines `zpk`-Objektes in `tf`-Objekt (umgekehrte Umwandlung natürlich ebenfalls möglich)

```
>> H = zpk([1, 2], [3, 4], 5);
>> tf(H)

ans =

  5 s^2 - 15 s + 10
  -----------------
   s^2 - 7 s + 12

Continuous-time transfer function.
```

Darstellung von Systemen

```
>> stepplot(H)

>> bode(H)

>> pzmap(H)

>> nyquist(H)
```

Aufgaben IV

Gebeben sei die Übertragungsfunktion eines PT1-Gliedes

$$H(s) = \frac{7}{1 + \mathrm{s} \cdot 1\,\mathrm{ms}}$$

1. Verifizieren Sie die Zeitkonstante mit Hilfe der Sprungantwort
2. Ermitteln Sie die $3\,\mathrm{dB}$ Grenzfrequenz mit Hilfe des Bode-Diagramms
3. Fügen Sie zu $H(s)$ eine Eingangsverzögerung von $3\,\mathrm{ms}$ hinzu und unteruchen Sie erneut die Sprungantwort sowie das Bode-Diagramm

Gegeben sie die Übertragungsfunktion

$$H(s) = \frac{3\mathrm{s}^2 - 9\mathrm{s} + 6}{\mathrm{s}^3 + 7\mathrm{s}^2 + 16\mathrm{s} + 10} \cdot \mathrm{e}^{-0.3\mathrm{s}}$$

3. Verifizieren Sie die Pol und Nullstellen

Aufgaben V

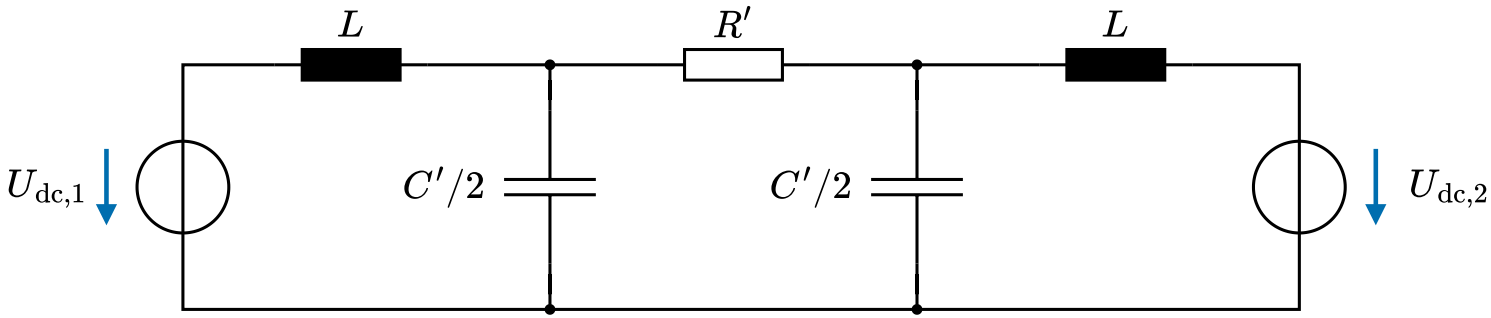
1. Erstellen Sie `tf`-Objekte für das Modell des Gleichstrommotors
2. Dimensionieren Sie einen PI-Regler mittels `pidTuner()`
3. Erstellen Sie ein Simulink Modell zum Gleichstrommotor mit `tf`-Objekten
4. Überlegen Sie Vor- und Nachteile dieses Simulationsmodells

Simulink Funktionen IV

- *tf* Block
- *Simulink Data-Dictionary*

Modellierung einer HGÜ-Leitung

Modellierung des DC-Kreises einer HGÜ-Anlage I



- DC-Nennspannung: $600\,\mathrm{kV}$
- Induktivität der DC-Quelle: $10\,\mathrm{mH}$
- Widerstandsbelag: $R' = 10\,\mathrm{m\Omega/km}$
- Kapzitätsbelag: $C' = 200\,\mathrm{nF/km}$
- Leitungslänge: $l = 200\,\mathrm{km}$

Aufgabe VI

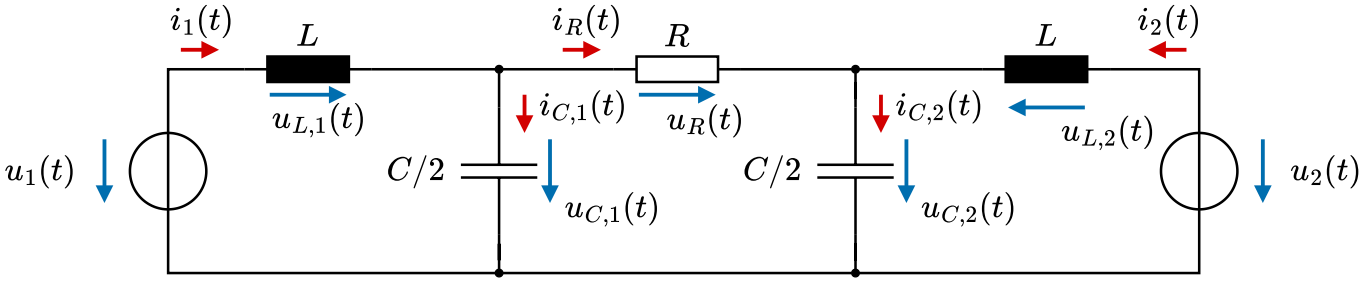
- Erstellen Sie ein Simulationsmodell der HGÜ-Anlage mittels (Euler-Verfahren mit einer Schrittweite $1\,\mu\mathrm{s}$)
  1. Vorgehen nach Nollau
  2. Simscape Elementen
- Stellen Sie die abgegebene Leistung der jeweiligen Stationen dar
- Simulieren Sie einen Spannungssprung von $600\,\mathrm{kV}$ auf $605\,\mathrm{kV}$
- Diskutieren Sie Vor- und Nachteile der verschiedenen Varianten

Simulink Funktionen IV

- *Goto*- und *From*-Blöcke
- Simscape *Solver Configuration* Block mit Option *Use local solver*

5-Punkte Lösung des HVDC-Modells

1.



2. $$u_{C,1}(t) = \frac{2}{C}\int i_{C,1}(t)\,\mathrm{d}t \quad u_{C,2}(t) = \frac{2}{C}\int i_{C,2}(t)\,\mathrm{d}t$$

$$i_1(t) = \frac{1}{L}\int u_{L,1}(t)\,dt \quad i_2(t) = \frac{1}{L}\int u_{L,2}(t)\,\mathrm{d}t$$

3. $$u_{L,1}(t) = u_1(t) - u_{C,1}(t) \quad u_{L,2}(t) = u_2(t) - u_{C,2}(t) \quad u_R(t) = u_{C,1}(t) - u_{C,2}(t)$$

$$i_{C,1}(t) = i_1(t) - i_R(t) \quad i_{C,2}(t) = i_2(t) + i_R(t)$$

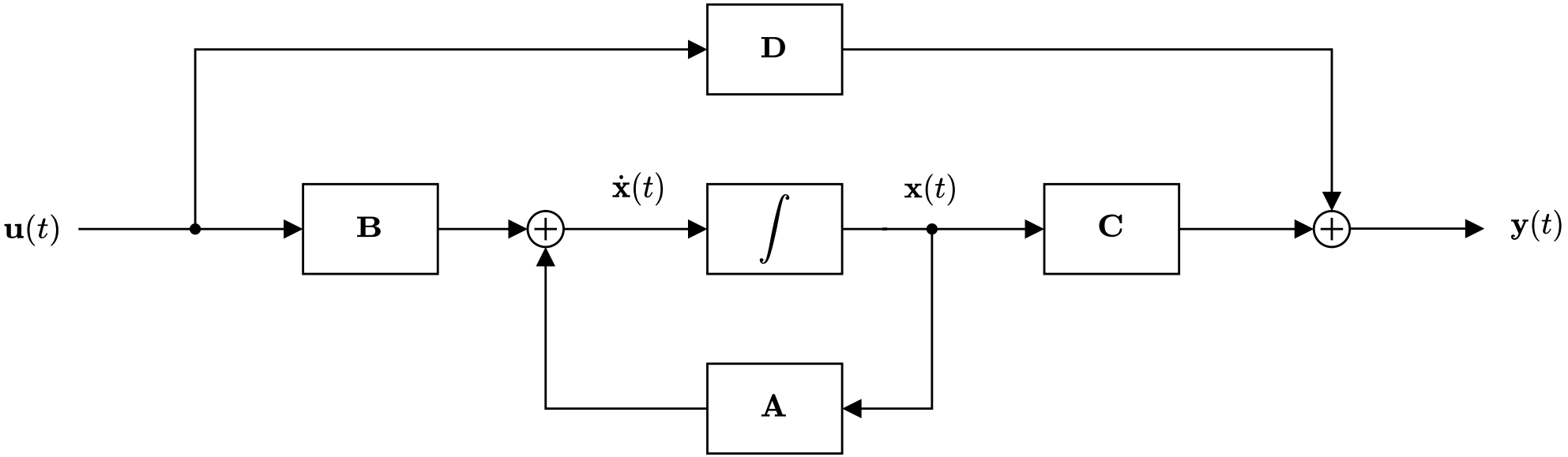
4. $$i_R(t) = \frac{1}{R}\cdot u_R(t)$$

Zustandsraumdarstellung

Darstellung des Systems im Zustandsraum

$$\dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{B} \cdot \mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C} \cdot \mathbf{x}(t) + \mathbf{D} \cdot \mathbf{u}(t)$$



Umformung der Zustandsgleichungen in Differentiation

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} i_{L,1}(t) \\ i_{L,2}(t) \\ u_{C,1}(t) \\ u_{C,2}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{L} \cdot u_{L,1}(t) \\ \frac{1}{L} \cdot u_{L,2}(t) \\ \frac{2}{C} \cdot i_{C,1}(t) \\ \frac{2}{C} \cdot i_{C,2}(t) \end{bmatrix}$$

Einsetzen der Bilanzgleichungen und der statischen Grundbeziehungen

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} i_{L,1}(t) \\ i_{L,2}(t) \\ u_{C,1}(t) \\ u_{C,2}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{L} \cdot (u_1(t) - u_{C,1}(t)) \\ \frac{1}{L} \cdot (u_2(t) - u_{C,2}(t)) \\ \frac{2}{C} \cdot (i_1(t) - \frac{1}{R} \cdot (u_{C,1}(t) - u_{C,2}(t))) \\ \frac{2}{C} \cdot (i_2(t) + \frac{1}{R} \cdot (u_{C,1}(t) - u_{C,2}(t))) \end{bmatrix}$$

Festlegen der Eingangs-, Ausgangs- und Zustandsgrößen

$$\mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \qquad \mathbf{y}(t) = \begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix} \qquad \mathbf{x}(t) = \begin{bmatrix} i_{L,1}(t) \\ i_{L,2}(t) \\ u_{C,1}(t) \\ u_{C,2}(t) \end{bmatrix}$$

Festlegen der Koeffizienten

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} 0 & 0 & -1/L & 0 \\ 0 & 0 & 0 & -1/L \\ \frac{2}{C} & 0 & -\frac{2}{RC} & \frac{2}{RC} \\ 0 & \frac{2}{C} & \frac{2}{RC} & -\frac{2}{RC} \end{bmatrix}}_{\mathbf{A}} \cdot \mathbf{x}(t) + \underbrace{\begin{bmatrix} 1/L & 0 \\ 0 & 1/L \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{B}} \cdot \mathbf{u}(t)$$

$$\mathbf{y}(t) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{C}} \cdot \mathbf{x}(t)$$

Aufgabe VII

- Bauen Sie das Modell in Simulink im Zustandsraum auf
- Vergleichen Sie die Ergebnisse mit der Simulation des 5-Punkte Schemas
- Ersetzen Sie den *Continuous*-Integrator durch einen *Discrete*-Integrator
- Vergleichen Sie die Optionen *Forward-Euler* und *Backward-Euler*

Simulink Funktionen V

- *State-Space*
- *Mux*- und *Demux*-Blöcke
- *Discrete-Time Integrator*

Darstellung der Vorwärts-Euler-Integration im z-Bereich

Berechnung des Update Schrittes

$$x(t + T_\mathrm{s}) = x(t) + \dot{x}(t) \cdot T_\mathrm{s} = x(t) + y(t) \cdot T_\mathrm{s}$$

Betrachtung diskreter Zeitschritte mit Schrittweite T_s

$$x[k] = x[k-1] + y[k-1] \cdot T_\mathrm{s}$$

Anwendung im z-Bereich

$$X(z) = X(z) \cdot z^{-1} + Y(z) \cdot z^{-1} \cdot T_\mathrm{s}$$

$$X(z) \cdot \left(1 - z^{-1}\right) = Y(z) \cdot z^{-1} \cdot T_\mathrm{s}$$

Berechnung der Übertragungsfunktion

$$\frac{X(z)}{Y(z)} = \frac{z^{-1} \cdot T_\mathrm{s}}{1 - z^{-1}} = \frac{T_\mathrm{s}}{z - 1}$$

Vorsicht: Polstelle auf dem Einheitskreis

Aufgabe VIII

- Bauen Sie die Vorwärts-Euler-Integration explizit auf

Simulink Funktionen VI

- *Unit Delay*

Problematik bei der Simulation: Verzögerung in der Feedback-Schleife verursacht Instablität

Backward-Euler Integration

Idee: Entfernung der zusätzlichen Verzögerung

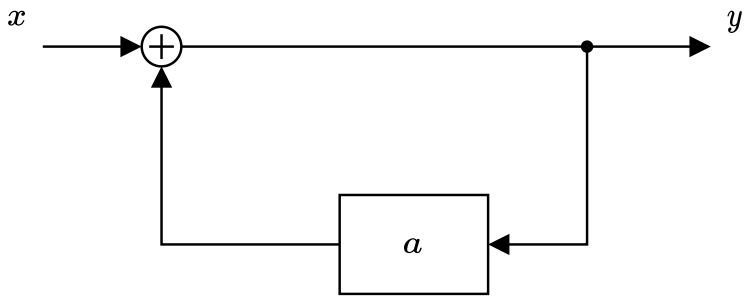
$$\frac{X(z)}{Y(z)} = \frac{T_\mathrm{s} \cdot z}{z-1}$$

Aufgabe IX

– Bauen Sie die Rückwärts-Euler-Integration explizit auf

Problematik bei der Simulation: Algebraische Schleife

Algebraische Schleifen in Simulink



Berechnung des Ausgangssignals y nicht möglich wegen impliziter Abhängigkeit

Versuch zur expliziten Auflösung der Abhängigkeit

$$y = x + y \cdot a$$

$$y \cdot (1 - a) = x$$

$$y = \frac{x}{1 - a}$$

Nur sinnvoll möglich für $a < 1$!

Simulink verwendet Algorithmus um algebraische Schleife aufzulösen.

Aufgabe X

- Lösen Sie die Algebraische Schleife bei der Rückwärts-Euler-Integration auf
- Vergleichen Sie das Ergebnis mit der Simulation in Simscape

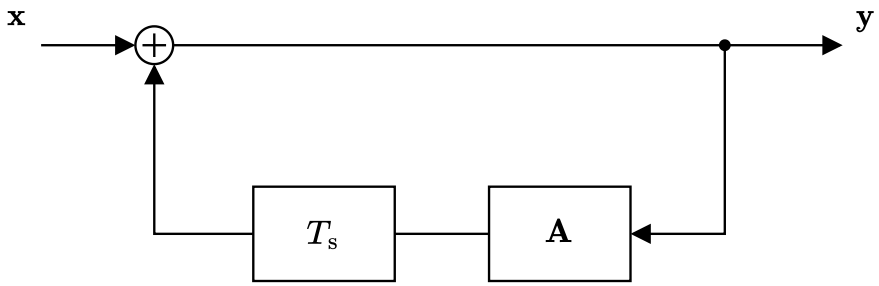
Hinweis: Umformung der algebraischen Schleife bei Matrix-Koeffizienten

$$\mathbf{y} = \mathbf{x} + T_\mathrm{s} \cdot \mathbf{A} \cdot \mathbf{y}$$

$$\mathbf{y} - T_\mathrm{s} \cdot \mathbf{A} \cdot \mathbf{y} = \mathbf{x}$$

$$\mathbf{y} \cdot (\mathbf{I} - T_\mathrm{s} \cdot \mathbf{A} \cdot \mathbf{y}) = \mathbf{x}$$

$$\mathbf{y} = \mathbf{x} \cdot (\mathbf{I} - T_\mathrm{s} \cdot \mathbf{A})^{-1}$$



Multiratensysteme

Aufgabe XI

Bauen Sie das DC Motor Modell mit Hilfe von Simscape Elementen auf

Simulink Funktionen VII

- DC Motor
- Rational Friction
- Ideal Rotational Motion Sensor
- Ideal Torque Source
- Inertia

Aufgabe XII

Erweitern Sie das Simscape DC Motor Modell um einen P-Regler.

Simulink Funktionen VIII

- Simulink Subsystem
- Subsystem Mask

Aufgabe XIII

Erweitern Sie den P-Regler um einen PI-Regler. Verwenden Sie hierbei als Integrator einen *Discrete-Time Integrator*

Multiratensysteme

Implementierung der Regelung als digitaler Regler (z.B. auf einem Mikrocontroller)

Regelung wird in einem festen Takt gerechnet

Simulation der Auswirkungen von Abtastung und reduzierter Taktrate

Anzeige der unterschiedlichen Takte der Simulation

- Schwarz: Kontinuierliche Zustände (Integratoren ode1, ode45, etc.)
- Gelb: Verschiedene Takte bei Übergängen
- Rot/Grün/Blau: Fixe Taktraten digitaler Systeme

Aufgabe XIV

Realisieren Sie die Regelung für eine Taktrate $100\ \mu\mathrm{s}$. Der Sollwert wird mit einem Takt von

1. $1\ \mathrm{ms}$
2. $0.33\ \mathrm{ms}$

vorgegeben.

Simulink Funktionen IX

- Rate Transition

Simulation verschiedener Datentypengenauigkeiten

Bei der Umsetzung des Regelungsalgorithmus auf einem einfachen Prozessor stehen nur vereinfachte Datentypen zur Verfügung.

Simulink verwendet (wie Matlab) standardmäßig zur Simulation den Datentyp `double`

Aufgabe XV

Erweitern Sie den Regelungsalgorithms, damit er nur mit einer Datentypengenauigkeit von `single` rechnet

Simulink Funktionen X

- Data Type Conversion

Symbolisches Rechnen

Symbolic Math Toolbox

Aufgabenstellung bei vielen Modellierungsbeispielen:

- Zusammenstellen verschiedener physikalischer Zusammenhänge
- Umformung von algebraischen Gleichungenn und/oder Differentialgleichungen

Matlab und Simulink sind numerische Tools und können *keine* Gleichungen umformen

Abhilfe: *Symbolic Math Toolbox* in Matlab

Erstellen symbolischer Objekte

Erzeugen einer symbolischen Variable `x`

```
>> x = sym('x')

x =

x

>> whos
  Name       Size                Bytes  Class                     Attributes

  x          1x1                     8  sym
```

Erstellen einer oder mehrerer symbolischer Variablen

```
>> syms x y z;
```

Arbeiten mit Funktionen

Definition einer Funktion

$$f(x) = \frac{\sin(x)}{x^2 + 2x + 3}$$

```
>> syms x
>> f(x) = sin(x)/(x^2 + 2*x + 3);
```

Schönere Darstellung des Ergebnisses mittels `pretty()`

```
>> pretty(f(x))
   sin(x)
------------
 2
x  + 2 x + 3
```

Ausrechnen von Funktionswerten

```
>> x = 1
>> f(x)

ans =

sin(1)/6

>> eval(f(x))

ans =

    0.1402
```

Substitution einzelner Variablen

```
>> subs(f(x), x, a^2)

ans =

sin(a^2)/(a^4 + 2*a^2 + 3)

>> subs(f(x), x, asin(x))

ans =

x/(2*asin(x) + asin(x)^2 + 3)
```

Symbolisches Rechnen mit konkreten Zahlen

```
>> ein_zehntel = sym(1/10)

ans =

1/10

>> 2*ein_zehntel

ans =

1/5
```

Rechnen mit beliebiger Genauigkeit: VPA Objekte (*variable precision arithmetic*)

```
>> vpa('123/456', 20)

ans =

0.26973684210526315789

>> vpa(pi, 50)

ans =

3.1415926535897932384626433832795028841971693993751

>> vpa(str2sym('exp(3)'))

ans =

20.085536923187667740928529654582
```

Analytische Berechnen eines Integrals I

Berechnung der Stammfunktion von

$$f(x) = x^2 + 2x + 3$$

```
>> sym x;
>> f(x) = x^2 + 2*x+3;
>> F = int(f(x));
>> pretty(F)
    2
x (x  + 3 x + 9)
----------------
        3
```

Berechnung von Grenzwerte

$$\lim_{x \to 0+} \frac{1}{x} \to +\infty \qquad \lim_{x \to 0-} \frac{1}{x} \to -\infty \qquad \lim_{x \to \infty} \frac{3x+2}{x-4} = 3$$

```
>> limit(1/x, x, 0, 'right')
ans =
Inf

>> limit(f, x, 0, 'left')
ans =
-Inf

>> limit((3*x+2)/(x-4), x, inf)
ans =
3
```

Analytische Berechnung der Ableitung

$f(x) = x^2 + x + 1 \qquad f'(x) = 2x + 1$

```
>> diff(x^2+x+1, x)
ans =
2*x + 1
```

$f(x) = \sin(a \cdot x) \qquad f'(x) = a \cdot \cos(a \cdot x)$

```
>> diff(sin(a*x), x)
ans =
a*cos(a*x)
```

Analytische Berechnen eines Integrals II

Berechnung des bestimmten Integrals

$$I = \int_a^b x^2 + 2x + 3 \, \mathrm{d}x$$

```
>> I(a, b) = int(f(x), a, b);
>> pretty(I)
   3               3
  a     2         b     2
- -- - a  - 3 a + -- + b  + 3 b
   3               3
```

Analytisches Lösen eines (linearen) Gleichungssystems I

Lösung eines linearen Gleichungssystems über die Definition von Gleichungen

$$3 \cdot x_1 + 4 \cdot x_2 = 7$$
$$4 \cdot x_1 - 12 \cdot x_2 = 8$$

```
>> gln = [3*x1 + 4*x2 == 7, 4*x1 - 12*x2 == 8];
>> solve(gln, x1, x2)
ans =
  struct with fields:
    x1: 29/13
    x2: 1/13
```

Analytisches Lösen eines (linearen) Gleichungssystems II

Lösung eines linearen Gleichungssystems über die Matrixdarstellung

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

```
>> [A, b] = equationsToMatrix(gln)
A =
[3,   4]
[4, -12]

b =
7
8

>> linsolve(A, b)
ans =
29/13
 1/13
```

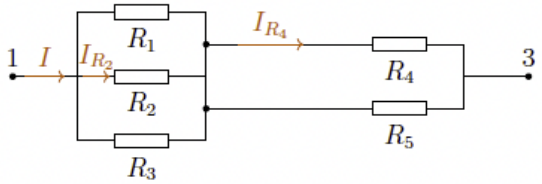
Aufgabe XVI

1. Eine Spule mit N Windungen und der Querschnittsfläche A rotiert in einem homogenen Magnetfeld der Flussdichte B. Berechnen Sie die induzierte Spannung $u(t)$.

$$\Phi(t) = B \cdot A \cdot \cos(\omega t) \qquad u(t) = -N \cdot \frac{\mathrm{d}}{\mathrm{d}t}\Phi(t)$$

2. Aufgabe aus ET1 - Übung (Blatt 3 - Aufgabe 1)

Gegeben: R_1, R_2, R_3, R_4. Bestimmen Sie den Widerstand R_5 so, dass durch die Widerstände R_2 und R_4 die gleichen Ströme fließen. $R_5 = f(R_1, \cdots, R_4)$



3. Vergleichen Sie folgende mathematischen Zufälle

$$\pi^4 + \pi^5 \approx \mathrm{e}^6$$

$$\mathrm{e}^{\pi\sqrt{163}} \approx 12^3(231^2 - 1)^3 + 744 \qquad \text{(Ramanujan's constant)}$$

Modellierung dynamischer Systeme in Matlab II

Lösen von Differentialgleichungen in Matlab I

Lösen von nicht-linearen Differentialgleichung ist ebenfalls in Matlab (ohne Simulink) möglich

Vorteile:

- *Modelling with Code* (Versionierung, Parametrierung)
- Flexible Anbindung an weitere IT-Systeme (Datenbanken, UI, etc.)
- Export als binary file für 3rd-Party Simulationsumgebungen (z.B. als DLL via Code Generierung in C/C++)

Formulierung des mathematischen Modells als

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = f(t, \mathbf{x}(t)) \qquad \text{mit Anfangswert} \qquad \mathbf{x}(t_0) = \mathbf{x}_0$$

Hinweis: Die Funktion $f(t, \mathbf{x}(t))$ muss nicht zwangsweise eine lineare Abbildung der Zustandsgrößen $\mathbf{x}(t)$ sein.

Function Handles in Matlab I

Funktionen sind in Matlab KEINE *first-class citizens*:

> *Funktionale Programmierung im klassischen Sinn nicht möglich*

Trotzdem lassen sich sogenannte *function handles* definieren

```
% my_function.m
function y = my_funktion(x)
   y = x.^2;
end
```

```
>> renamed_function = @my_function;
renamed_function =
  function_handle with value:
    @my_function

>> renamed_function(3)
ans =
     9
```

Function Handles in Matlab II

Funktion lassen sich in Matlab auch *anonym* (d.h. als sogenannte Lambda-Funktionen) definieren

```
>> square = @(x) x.^2

square =

  function_handle with value:

    @(x)x.^2
```

Im Gegensatz zu anderen Programmiersprachen ist es nicht möglich beliebige Ausdrücke zu realisieren wie z.B.

- if-else Verzweigungen
- komplexe Logik

Lösen von Differentialgleichungen in Matlab II

Definition einer Funktion zur Abbildung der Zustandsgrößen

```
function dxdt = xpunkt(t, x)
   dxdt = [
      ...
   ]
```

Lösen der Differentialgleichung mittels des Runge-Kutta Verfahrens `ode45()`

```
>> [t, x] = ode45(@xpunkt, tlim, x0)
```

Weitere Lösungsalgorithmen in Matlab

- `ode23`
- `ode113`
- `ode15s`
- `...`

Beispiel: Simulation eines Pendels mit Antrieb

$$F_\mathrm{B}(t) = m \cdot g \cdot \sin(\varphi(t))$$

$$F_\mathrm{R}(t) = -\frac{2}{\pi} \cdot F_{\mathrm{R},0} \cdot \arctan(10 \cdot \dot{\varphi}(t))$$

$$a(t) = -l \cdot \ddot{\varphi}(t)$$

Externer Antrieb

$$F_\mathrm{A}(t) = 20\,\mathrm{N} \cdot \sin(2\pi \cdot 0{,}2\,\mathrm{Hz} \cdot t)$$

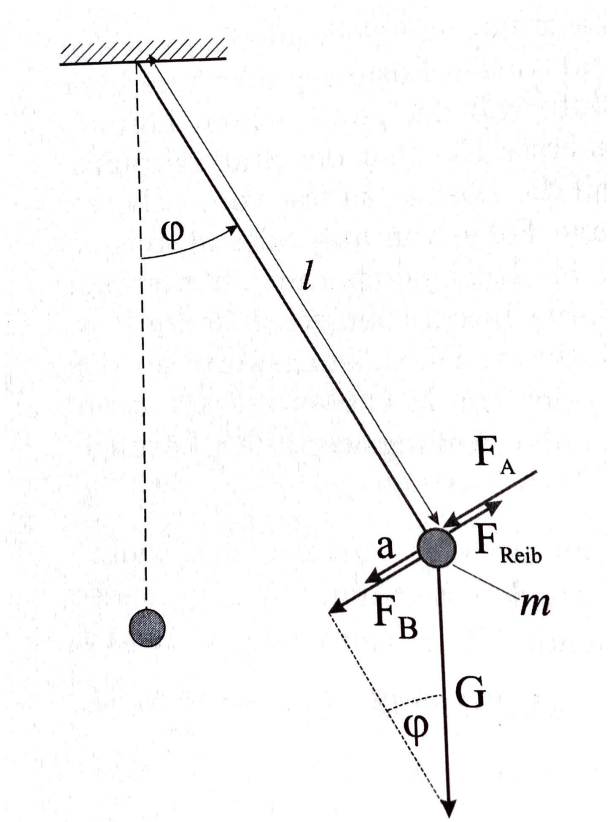
Kräftegleichgewicht

$$m \cdot a(t) = F_\mathrm{B}(t) - F_\mathrm{R}(t) + F_\mathrm{A}(t)$$

Aufstellen der Differentialgleichung

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \varphi(t) \\ \dot{\varphi}(t) \end{bmatrix}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} \dot{\varphi}(t) \\ \ddot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} \dot{\varphi}(t) \\ -\frac{F_\mathrm{B}(t) - F_\mathrm{R}(t) + F_\mathrm{A}(t)}{l \cdot m} \end{bmatrix}$$



Aufgabe XVII

1. Erstellen Sie ein Simulationsmodell in Matlab für den DC-Motor
2. Erstellen Sie ein Simulationsmodell in Matlab für das HGÜ-Modell
3. Erstellen Sie ein Simulationsmodell in Matlab zur Simulation eines epidemischen Verlaufes

$$\dot{S}(t) = -\alpha \cdot S(t) \cdot I(t)$$

$$\dot{I}(t) = \alpha \cdot S(t) \cdot I(t) - \beta \cdot I(t)$$

$$\dot{R}(t) = \beta \cdot I(t)$$

   - $S(t)$: Zahl der anfälligen Personen (*susceptible*)
   - $I(t)$: Zahl der infizierten Personen (*infectious*)
   - $R(t)$: Zahl der genesenen Personen (*recovered*)

YouTube: Mehr "Corona-Mathematik": Wie Epidemien modelliert werden

Referenzen

[1] A. Angermann, M. Beuschel, M. Rau, U. Wohlfarth, *Matlab - Simulink - Stateflow,* Oldenburg Verlag.

[2] H. Scherf, *Modellbildung und Simulation dynamischer Systeme,* Oldenburg Verlag.

[3] R. Nollau, *Modellierung und Simulation technischer Systeme,* Springer Verlag.