
Praktikum MATLAB®/Simulink® II

Prof. Dr.-Ing. U. Konigorski
Versuchsunterlagen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

REGELUNGSTECHNIK *rtm*
UND MECHATRONIK

Praktikum MATLAB®/Simulink® II

Prof. Dr.-Ing. U. Konigorski

Versuchsunterlagen



Technische Universität Darmstadt
Institut für Automatisierungstechnik und Mechatronik
Fachgebiet Regelungstechnik und Mechatronik
Prof. Dr.-Ing. U. Konigorski

Landgraf-Georg-Straße 4
64283 Darmstadt
Telefon 06151/16-25200
www.rtm.tu-darmstadt.de

Das Gesamtdokument ist unter CC BY-ND veröffentlicht:



<https://creativecommons.org/licenses/by-nd/4.0/>

Der Inhalt dieses Dokuments ausschließlich der Logos, des Layouts und der Schriftarten ist unter CC BY-SA veröffentlicht:



<https://creativecommons.org/licenses/by-sa/4.0/>



Inhaltsverzeichnis

1	Modellierung und Simulation eines Schlitten-Pendel-Systems	5
2	Steuerbarkeit und Beobachtbarkeit	21
3	LQ-Regelung und Animation	27
4	Beobachterentwurf – Benutzeroberflächen	35
5	Aufschwungsteuerung für das Pendel	41
6	Trajektorienfolgeregelung	45



Versuch 1

Modellierung und Simulation eines Schlitten-Pendel-Systems

0 Vorbereitungsaufgaben

Hinweis: Zu dem ersten Versuch sind im Skript Vorbereitungsaufgaben zu finden. Diese sind vor der Versuchsdurchführung zu lösen! Da die Ergebnisse wesentlich für die Versuchsdurchführung sind, ist die Musterlösung zu den Vorbereitungsaufgaben hier angegeben. Die wesentlichen Ergebnisse sind

- die nichtlinearen Bewegungsgleichungen (0.3) und (0.4),
- die linearisierten Bewegungsgleichungen für den unteren und oberen Arbeitspunkt (Gl. (0.5) und (0.6) bzw. Gl. (0.7) und (0.8)) sowie
- die Zustandsraumdarstellungen für den unteren und oberen Arbeitspunkt ((0.9) bzw. (0.10)).

0.1 Aufgabe 1.1 (Vorbereitung): Physikalische Modellbildung

Die physikalische Modellbildung erfolgt anhand der in Abbildung 0.1 dargestellten Anordnung. Das System hat zwei Freiheitsgrade. Der Übersicht halber wird im Folgenden die Abhängigkeit von der Zeit t weggelassen.

Die im Weiteren vorgestellte Herleitung der Bewegungsgleichungen anhand der LAGRANGEschen Gleichungen zweiter Art ist weitgehend aus [2] übernommen, wobei diese um die Beschreibung des Schlittens und der Reibungen erweitert wurde.

Mit den generalisierten Koordinaten

$$q_1 = x_S \quad \text{und} \quad q_2 = \varphi$$

lässt sich jede Position sowohl des Schlittens, als auch des Pendels beschreiben. Mit Angabe des Pendelschwerpunktes nach

$$x_P = x_S + \frac{l}{2} \sin \varphi \quad \text{und} \quad y_P = -\frac{l}{2} \cos \varphi$$

lassen sich die entsprechenden Geschwindigkeiten ableiten:

$$\dot{x}_P = \dot{x}_S + \dot{\varphi} \frac{l}{2} \cos \varphi \quad \text{und} \quad \dot{y}_P = \dot{\varphi} \frac{l}{2} \sin \varphi .$$

Nun erfolgt die Bestimmung der kinetischen Energie T :

$$\begin{aligned} T &= \underbrace{\frac{1}{2} m_P v_P^2 + \frac{1}{2} J \omega^2}_{\text{Pendel}} + \frac{1}{2} m_S v_S^2 \\ &= \frac{1}{2} m_P \cdot \left(\left(\dot{x}_S + \dot{\varphi} \frac{l}{2} \cos \varphi \right)^2 + \left(\dot{\varphi} \frac{l}{2} \sin \varphi \right)^2 \right) + \frac{1}{2} \cdot \frac{1}{12} m_P l^2 \dot{\varphi}^2 + \frac{1}{2} m_S v_S^2 \\ &= \frac{1}{2} m_P \cdot \left(\dot{x}_S^2 + 2 \dot{x}_S \dot{\varphi} \frac{l}{2} \cos \varphi + \left(\dot{\varphi} \frac{l}{2} \cos \varphi \right)^2 + \left(\dot{\varphi} \frac{l}{2} \sin \varphi \right)^2 + \frac{1}{12} l^2 \dot{\varphi}^2 \right) + \frac{1}{2} m_S v_S^2 \\ &= \frac{1}{2} m_P \cdot \left(\dot{x}_S^2 + \dot{x}_S \dot{\varphi} l \cos \varphi + l^2 \dot{\varphi}^2 \frac{1}{3} \right) + \frac{1}{2} m_S \dot{x}_S^2 . \end{aligned}$$

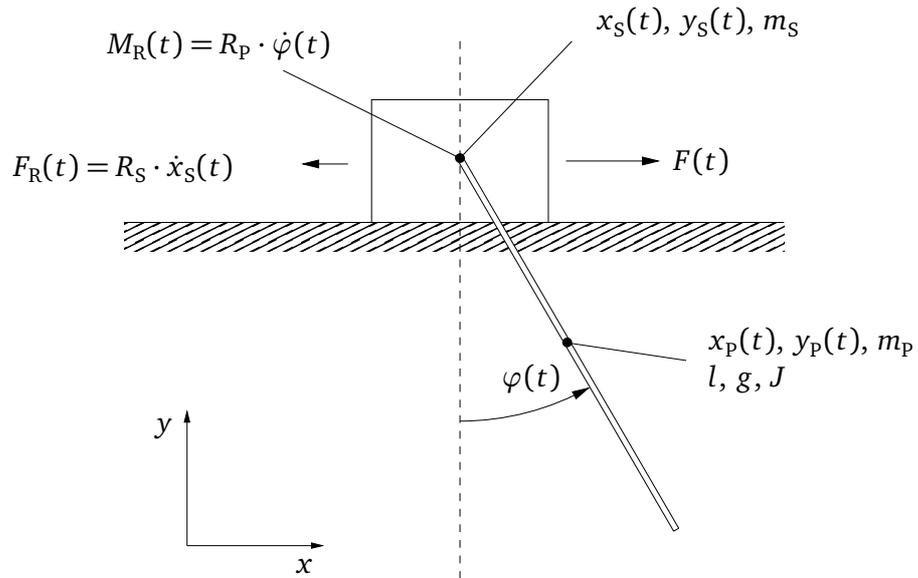


Abbildung 0.1: Darstellung des Schlitten-Pendel-Systems mit den zur Modellbildung zu verwendenden Größen.

Die potentielle Energie U des Pendels lässt sich mit

$$U = -m_P g \frac{l}{2} \cos \varphi$$

ausdrücken. Für die generalisierten Kräfte Q_x^* und Q_φ^* gilt:

$$Q_x^* = F \frac{\partial \dot{x}}{\partial \dot{x}} - F_R \frac{\partial \dot{x}}{\partial \dot{x}} = F - F_R$$

$$Q_\varphi^* = -M_R \frac{\partial \omega}{\partial \dot{\varphi}} = -M_R .$$

Mit der LAGRANGEschen Funktion

$$L = T - U$$

sind nun die LAGRANGEschen Gleichungen zweiter Art

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_n} - \frac{\partial L}{\partial q_n} = Q_n^* \quad n = 1, 2$$

aufzustellen. Als Zwischenschritt können folgende Beziehungen angegeben werden:

$$L = \frac{1}{2}m_p \cdot \left(\dot{x}_S^2 + \dot{x}_S \dot{\varphi} l \cos \varphi + \frac{1}{3}l^2 \dot{\varphi}^2 + gl \cos \varphi \right) + \frac{1}{2}m_s \dot{x}_S^2$$

$$\frac{\partial L}{\partial q_1} = \frac{\partial L}{\partial x_S} = 0$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_1} = \frac{d}{dt} \frac{\partial L}{\partial \dot{x}_S} = \frac{d}{dt} \left(\frac{1}{2}m_p \cdot (2\dot{x}_S + \dot{\varphi} l \cos \varphi) + m_s \dot{x}_S \right)$$

$$= (m_p + m_s)\ddot{x}_S + \left(\ddot{\varphi} \frac{l}{2} \cos \varphi - \dot{\varphi}^2 \frac{l}{2} \sin \varphi \right) \cdot m_p$$

$$\frac{\partial L}{\partial q_2} = \frac{\partial L}{\partial \varphi} = -\frac{1}{2}m_p \cdot (\dot{x}_S \dot{\varphi} l \sin \varphi + gl \sin \varphi)$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_2} = \frac{d}{dt} \frac{\partial L}{\partial \dot{\varphi}} = \frac{d}{dt} \left(\frac{1}{2}m_p \cdot \left(\dot{x}_S l \cos \varphi + \frac{2}{3}l^2 \dot{\varphi} \right) \right)$$

$$= m_p \cdot \left(\ddot{x}_S \frac{l}{2} \cos \varphi - \dot{x}_S \dot{\varphi} \frac{l}{2} \sin \varphi + \frac{l^2}{3} \ddot{\varphi} \right).$$

Die resultierenden Bewegungsgleichungen lauten:

$$(m_s + m_p) \cdot \ddot{x}_S + m_p \ddot{\varphi} \frac{l}{2} \cos \varphi - m_p \dot{\varphi}^2 \frac{l}{2} \sin \varphi = F - R_S \dot{x}_S \quad (0.1)$$

$$\frac{1}{2}l m_p \cdot \left(\ddot{x}_S \cos \varphi + \frac{2}{3}l \ddot{\varphi} + g \sin \varphi \right) = -R_P \dot{\varphi}. \quad (0.2)$$

Beide DGLs zweiter Ordnung sind verkoppelt. Wird (0.2) nach \ddot{x}_S aufgelöst und in (0.1) eingesetzt, so folgt für $\ddot{\varphi}$:

$$\ddot{\varphi} = \frac{(m_s + m_p) \cdot \left(g \sin \varphi + \frac{2R_P}{m_p l} \dot{\varphi} \right) + m_p \dot{\varphi}^2 \frac{l}{4} \sin(2\varphi) + \cos \varphi \cdot (F - R_S \dot{x}_S)}{m_p \frac{l}{2} \cos^2 \varphi - (m_s + m_p) \cdot \frac{2}{3}l}. \quad (0.3)$$

Die Position des Schlittens x_S kann nun durch zweimaliges Integrieren von

$$\ddot{x}_S = -\frac{1}{\cos \varphi} \cdot \left(g \sin \varphi + \frac{2R_P}{m_p l} \dot{\varphi} + \frac{2}{3}l \ddot{\varphi} \right) \quad (0.4)$$

ermittelt werden.

Gl. (0.3) und (0.4) stellen die gesuchten Bewegungsdifferentialgleichungen des Schlitten-Pendel-Systems dar. (Um eine Standardform zu erhalten, muss noch Gl. (0.3) in (0.4) eingesetzt werden. Um doppelte Auswertungen zu vermeiden, bietet es sich aber praktisch an, zunächst Gl. (0.3) und damit dann (0.4) zu berechnen.)

0.2 Aufgabe 1.2 (Vorbereitung): Linearisierung und Zustandsraumdarstellung

0.2.1 Linearisierung

Die Linearisierung der Gleichungen (0.1) und (0.2) um $\Phi_0 = 0$ rad liefert:

$$(m_s + m_p) \cdot \Delta \ddot{x}_S + m_p \frac{l}{2} \Delta \ddot{\varphi} - \Delta F + R_S \Delta \dot{x}_S = 0 \quad (0.5)$$

$$\frac{2}{3}l \Delta \ddot{\varphi} + \frac{2R_P}{m_p l} \Delta \dot{\varphi} + g \Delta \varphi + \Delta \ddot{x}_S = 0. \quad (0.6)$$

Analog lassen sich die im Arbeitspunkt $\Phi_0 = \pi$ rad linearisierten Gleichungen mit

$$(m_S + m_P) \cdot \Delta \ddot{x}_S - m_P \frac{l}{2} \Delta \ddot{\varphi} - \Delta F + R_S \Delta \dot{x}_S = 0 \quad (0.7)$$

$$\frac{2}{3} l \Delta \ddot{\varphi} + \frac{2R_P}{m_P l} \Delta \dot{\varphi} - g \Delta \varphi - \Delta \ddot{x}_S = 0 \quad (0.8)$$

angeben.

0.2.2 Zustandsraumdarstellung

Die Gleichungen (0.5) bis (0.8) sollen zunächst durch Differentialgleichungen erster Ordnung dargestellt werden. Dies erfolgt zunächst durch das Auflösen von (0.6) bzw. (0.8) nach $\Delta \ddot{x}_S$ und anschließendes Einsetzen in (0.5) bzw. (0.7). Für $\Phi_0 = 0$ rad kann folgende Zustandsraumdarstellung des linearisierten Schlitten-Pendel-Modells angegeben werden (angelehnt an [1]):

$$\begin{bmatrix} \Delta \dot{x}_S \\ \Delta \ddot{x}_S \\ \Delta \dot{\varphi} \\ \Delta \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-4R_S}{4m_S+m_P} & \frac{3gm_P}{4m_S+m_P} & \frac{6R_P}{l \cdot (4m_S+m_P)} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{6R_S}{l \cdot (4m_S+m_P)} & \frac{-6g \cdot (m_S+m_P)}{l \cdot (4m_S+m_P)} & \frac{-12 \cdot R_P \cdot (m_S+m_P)}{l^2 m_P \cdot (4m_S+m_P)} \end{bmatrix} \cdot \begin{bmatrix} \Delta x_S \\ \Delta \dot{x}_S \\ \Delta \varphi \\ \Delta \dot{\varphi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{4}{4m_S+m_P} \\ 0 \\ \frac{-6}{l \cdot (4m_S+m_P)} \end{bmatrix} \cdot \Delta F \quad (0.9)$$

$$\mathbf{y} = \begin{bmatrix} x_S \\ \varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta x_S \\ \Delta \dot{x}_S \\ \Delta \varphi \\ \Delta \dot{\varphi} \end{bmatrix}$$

Für den Arbeitspunkt $\Phi_0 = \pi$ rad folgt entsprechend:

$$\begin{bmatrix} \Delta \dot{x}_S \\ \Delta \ddot{x}_S \\ \Delta \dot{\varphi} \\ \Delta \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-4R_S}{4m_S+m_P} & \frac{3gm_P}{4m_S+m_P} & \frac{-6R_P}{l \cdot (4m_S+m_P)} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-6R_S}{l \cdot (4m_S+m_P)} & \frac{6g \cdot (m_S+m_P)}{l \cdot (4m_S+m_P)} & \frac{-12 \cdot R_P \cdot (m_S+m_P)}{l^2 m_P \cdot (4m_S+m_P)} \end{bmatrix} \cdot \begin{bmatrix} \Delta x_S \\ \Delta \dot{x}_S \\ \Delta \varphi \\ \Delta \dot{\varphi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{4}{4m_S+m_P} \\ 0 \\ \frac{6}{l \cdot (4m_S+m_P)} \end{bmatrix} \cdot \Delta F \quad (0.10)$$

$$\mathbf{y} = \begin{bmatrix} x_S \\ \varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta x_S \\ \Delta \dot{x}_S \\ \Delta \varphi \\ \Delta \dot{\varphi} \end{bmatrix}.$$

1 Versuchsdurchführung

Die folgenden Aufgaben sind teilweise während des Versuchs zu lösen und am nächsten Versuchsnachmittag oder spätestens eine Woche später mit dem Protokoll den Betreuern abzugeben. Die Aufgaben, die während des Versuchs zu lösen sind, sind handschriftlich in den umrandeten Feldern einzutragen.

1.1 Aufgaben

Legen Sie die Parameter des Pendels und ihre Zahlenwerte in einem Skript (`initModell.m`) ab, so dass diese geändert werden können, ohne die Simulink-Modelle und Funktionen anpassen zu müssen.

Aufgabe 1.3 (Durchführung/Nachbearbeitung):

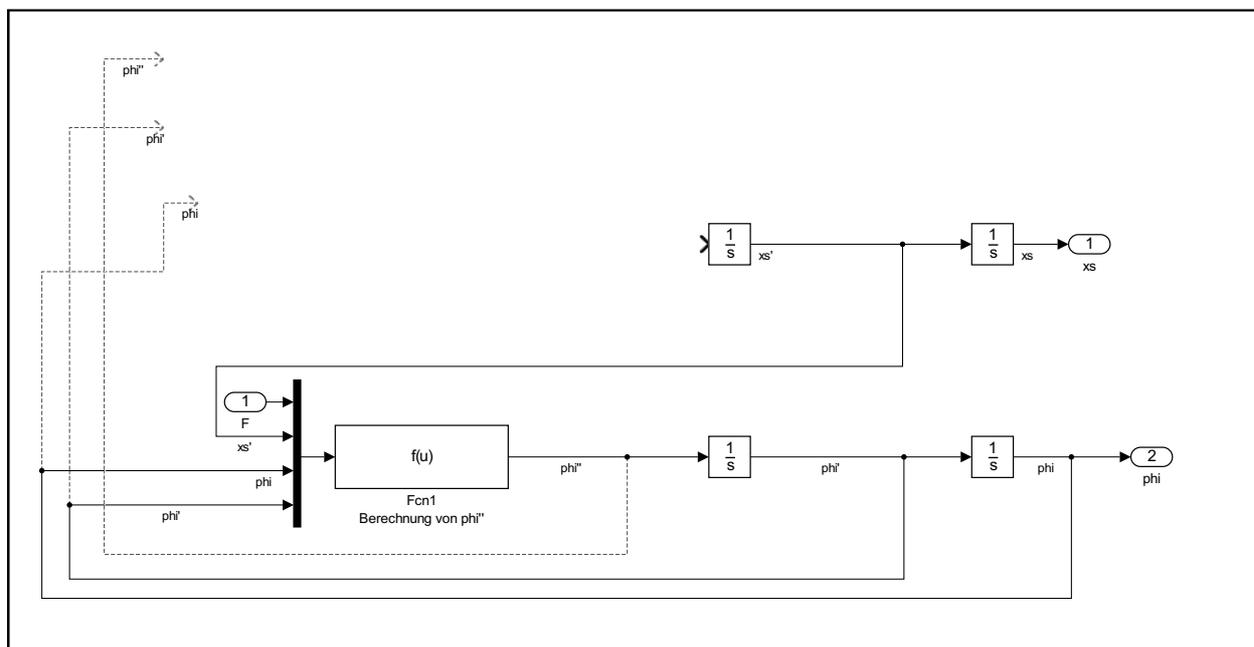
Das **nichtlineare** Modell soll jeweils mit Hilfe von

- elementaren Funktionsblöcken der Math-Operations-Library und Fcn-Blocks,
- MATLAB-Function-Blocks sowie
- einer M-File S-Function

realisiert werden. Bearbeiten Sie die folgenden Aufgaben und implementieren Sie die Modelle anschließend in Simulink.

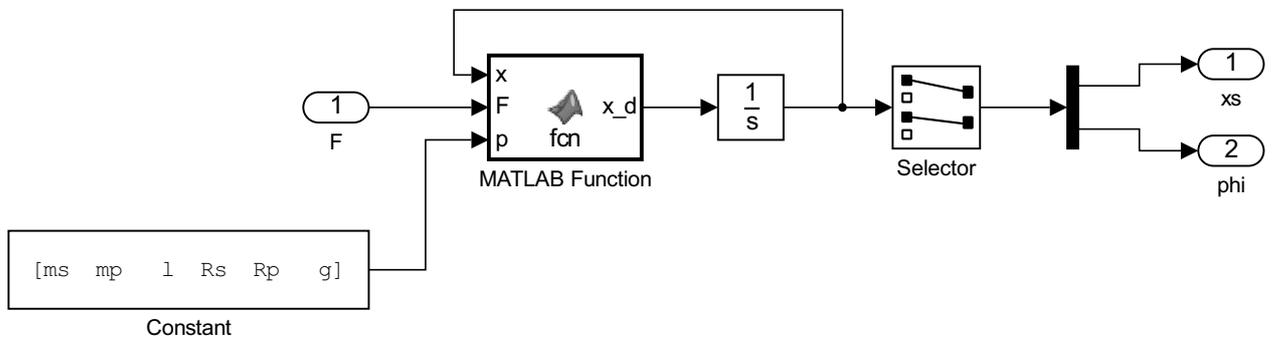
- Ergänzen Sie die Blöcke zur Berechnung der Position x_s und des Winkels φ und geben Sie die Funktion des Blocks Fcn1 an!

Gehen Sie hierzu von den verkoppelten Gleichungen (0.3) und (0.4) aus. Dabei soll (0.3) durch den Fcn-Block Fcn1 realisiert werden, und (0.4) soll durch elementare Funktionsblöcke aufgebaut werden.



Inhalt Fcn1:

- Wie lautet der Inhalt des MATLAB-Funktion-Blocks MATLAB Function? Wie müssen Sie den Anfangswert des Integrators wählen, damit der Pendelstab zu Beginn nach oben zeigt?



- Vervollständigen Sie das M-File an die vorliegende Aufgabenstellung an den entsprechenden Stellen.
Hinweis: Bei den Praktikumsunterlagen ist eine Vorlage für die S-Function zur Verfügung gestellt (siehe auch den Anhang zum Versuch 1 im Skript).

```

function sFunModell(block)

    setup(block);

end

% *****
% Initialisierung
% *****
function setup(block)

    % Anzahl der Ein-/Ausgänge
    block.NumInputPorts =
    block.NumOutputPorts =

    % Eigenschaften des Eingangs
    block.InputPort(1).Dimensions =
    block.InputPort(1).DatatypeID =
    block.InputPort(1).Complexity =
    block.InputPort(1).DirectFeedthrough =
    block.InputPort(1).SamplingMode =

    % Eigenschaften des 1. Ausgangs
    block.OutputPort(1).Dimensions =
    block.OutputPort(1).DatatypeID =
    block.OutputPort(1).Complexity =
    block.OutputPort(1).SamplingMode =

    % Eigenschaften des 2. Ausgangs
    block.OutputPort(2).Dimensions =
    block.OutputPort(2).DatatypeID =
    block.OutputPort(2).Complexity =
    block.OutputPort(2).SamplingMode =

    % Anzahl der Zustände
    block.NumContStates =

    % Anzahl der Parameter
    block.NumDialogPrms =

    % Abtastzeit definieren -> zeitkontinuierlich
    block.SampleTimes = [0 0];

    % weitere Methoden registrieren
    block.RegBlockMethod('InitializeConditions', @InitializeConditions);
    block.RegBlockMethod('Outputs', @Outputs);
    block.RegBlockMethod('Derivatives', @Derivatives);
    block.RegBlockMethod('Terminate', @Terminate);

end

```

```

% *****
% Anfangsbedingungen setzen
% *****
function InitializeConditions(block)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Funktion vollständig ergänzen!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

% *****
% Ausgänge berechnen
% *****
function Outputs(block)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Funktion vollständig ergänzen!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    block.OutputPort(1).Data =
    block.OutputPort(2).Data =

end

% *****
% Ableitungen berechnen
% *****
function Derivatives(block)

    % Parameter auslesen
    ms = block.DialogPrm(1).Data;
    Rs = block.DialogPrm(2).Data;
    mp = block.DialogPrm(3).Data;
    Rp = block.DialogPrm(4).Data;
    l = block.DialogPrm(5).Data;
    g = block.DialogPrm(6).Data;

    % Zustände auslesen
    x      = block.ContStates.Data;
    xs     = x(1);
    xs_d   = x(2);
    phi    = x(3);
    phi_d  = x(4);

    % Eingang auslesen
    F = block.InputPort(1).Data(1);

    % Ableitungen berechnen
    phi_dd = ( ...
                (ms + mp) * (g * sin(phi) + 2 * Rp/(mp*1) * phi_d) + ...

```

```

        mp * phi_d^2 * 1/4 * sin(2*phi) + ...
        cos(phi) * (F - Rs * xs_d) ...
    ) / (mp * 1/2 * cos(phi)^2 - (ms + mp) * 2/3 * l);

xs_dd = -1/cos(phi) * ...
        (g * sin(phi) + 2 * Rp/(mp * l) * phi_d + 2/3 * l * phi_dd);

% Ableitungen zuweisen
block.Derivatives.Data =

end

% *****
% Aufräumen (wenn nötig)
% *****
function Terminate(block)
end

```

- Welche Vor- und Nachteile bieten die unterschiedlichen Möglichkeiten der Realisierung nichtlinearer Modelle?

	Vorteile	Nachteile
Math-Operations-Library		
Fcn-Block		
MATLAB-Function-Block		
M-File S-Function		

Aufgabe 1.4 (Durchführung/Nachbearbeitung):

Das **lineare** Modell soll als Zustandsraummodell implementiert werden. Bearbeiten Sie die Aufgabe und implementieren Sie das Modell anschließend in Simulink. Der Block *Integrator* und die Bibliotheken *Sinks* und *Sources* können dabei ohne Einschränkung verwendet werden.

- Vervollständigen Sie die Funktion `initLinear`, welche die Matrizen **A**, **B**, **C** und **D** für das ZRM in Abhängigkeit des Arbeitspunktes initialisiert.

```

function [A, B, C, D] = initLinear( l, mp, ms, Rs, Rp, g, AP )
% function [A, B, C, D] = initLinear( lPendel, mPendel, mSchlitten→
    ←, RSchlitten, RPendel, g, arbeitspunkt)
% Arbeitspunkt:
% arbeitspunkt = 0; oder
% arbeitspunkt = pi;
%
% Zustände:

```

```

% x = [ xs, xs_d, phi, phi_d ]'
%

if ( AP == 0 )
    % Hängendes Pendel
    A =

    B =

    C =

elseif ( AP == pi )
    % Stehendes Pendel
    A =

    B =

    C =

else
    % Andere Werte für AP entweder nicht sinnvoll, da kein
    % AP oder schlicht Vielfache von pi und daher nicht relevant.
    error( ['Wert_arbeitspunkt_nicht_unterstützt.' ] );
end

D =

end

```

Die nachfolgenden Aufgaben sind in Ihrem Protokoll zu beantworten.

Verwenden Sie für die nachfolgenden Simulationen den *ode45 (Dormand-Prince)*-Solver und eine geeignete maximale Schrittweite (auf die MATLAB-Ausgabe während der Simulation achten). Verringern Sie die maximale Schrittweite, wenn die Graphen sehr kantig wirken.

Aufgabe 1.5 (Durchführung/Nachbearbeitung):

Untersuchen Sie das Verhalten bzw. die Reaktion der erstellten Modelle anhand der Auswertung der Signale $x_s(t)$ und $\varphi(t)$ auf eine sinusförmige Anregung mit

$$F(t) = \sin(\omega \cdot t) \quad \text{und} \quad \omega = 1 \text{ rad/s} \quad \text{für die Zeitspanne} \quad t = 0 \text{ s} \quad \text{bis} \quad t = 20 \text{ s}.$$

- Sind die Simulationsergebnisse aller drei nichtlinearen Modelle identisch ($\varphi(t=0) = 0 \text{ rad}$)? Fertigen Sie dazu folgenden Plot mit Hilfe von `subplot` und `linkaxes` an: Stellen Sie im ersten Subplot die anregende Kraft, im zweiten die drei absoluten Positionen des Schlittens und im dritten die absoluten Winkel dar. Vergessen Sie nicht die sinnvolle Achsenskalierung, die Achsenbeschriftung und die Verwendung unterschiedlicher Strichtypen!
(Für die Erstellung der Plots beachten Sie bitte Abschnitt **Physikalische Modellbildung** → **Simulation** → **Darstellung der Ergebnisse im Skript von Versuch 1**. Die Nichtbeachtung der Richtlinien führt zu Punktabzug!)

Hinweis:

Suchen Sie für die folgenden Aufgaben ein geeignetes nichtlineares Modell zum Vergleich mit dem linearisierten Modell aus.

- Stellen Sie die Simulationsergebnisse von $F(t)$, $x_s(t)$ und $\varphi(t)$ des linearisierten und eines nichtlinearen Modells (Absolutwerte angeben! Siehe Hinweis im Abschnitt Simulation im Skript zum Versuch 1.) jeweils für $\varphi(t=0) = 0 \text{ rad}$ und $\varphi(t=0) = \pi \text{ rad}$ dar. Passen Sie die Anfangsbedingungen dem Arbeitspunkt an.
- Ist das Verhalten des Pendels (bzw. des Winkels $\varphi(t)$) plausibel? Beschreiben Sie dazu die erwartete und die simulierte Reaktion des Modells für die ersten zwei Sekunden der Simulation. (Zu beachten sind $F(t)$, $x_s(t)$ und $\varphi(t)$.)

Hinweise:

Beachten Sie bei der Simulation, dass *Arbeitspunkt* und *Anfangsbedingungen* korrekt gewählt sind. Vergewissern Sie sich, dass der Arbeitspunkt und die Anfangsbedingungen des linearen Modells zu den Anfangsbedingungen des nichtlinearen Modells passen.

Um die einzelnen Größen miteinander vergleichen zu können, sollen diese entsprechend in gleiche Diagramme gezeichnet werden. Achten Sie darauf, wenn Sie mehrere Graphen in ein Diagramm zeichnen, dass diese auch im Schwarz-Weiß-Druck unterscheidbar dargestellt sind (verschiedene Marker/Strichtypen verwenden).

- Lässt sich anhand der Grafiken feststellen, ob die Linearisierung richtig durchgeführt wurde? Vergrößern Sie den für diese Aussage relevanten Bereich. Beschreiben Sie das Verhalten des linearisierten Modells.

Aufgabe 1.6 (Durchführung/Nachbearbeitung):

Nun ist eine rechteckförmige Anregung gemäß Abbildung 1.1 anzusetzen.

- Vergleichen Sie das Verhalten des linearisierten und eines nichtlinearen Modells. Betrachten Sie auch hier beide Arbeitspunkte und erstellen Sie zu beiden Arbeitspunkten Grafiken.

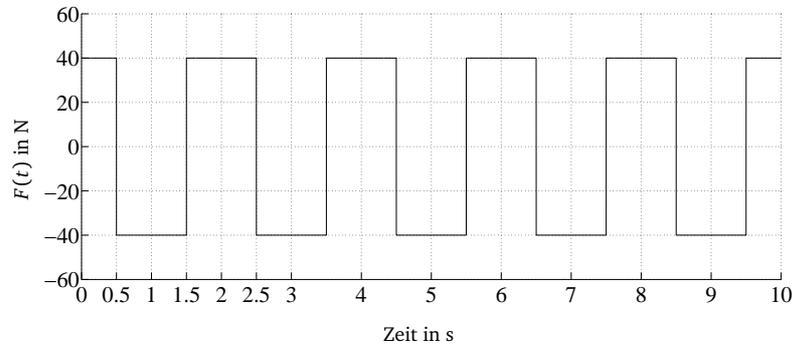


Abbildung 1.1: Rechteck-Signal $F(t)$.

Aufgabe 1.7 (Durchführung/Nachbearbeitung):

Schließlich ist eine rechteckförmige Anregung gemäß Abbildung 1.2, welche bis auf eine Phasenverschiebung dem Signal in Abbildung 1.1 entspricht, anzunehmen.

- Welche Auswirkung hat diese Phasenverschiebung auf das Verhalten von $x_S(t)$ und $\varphi(t)$? Betrachten Sie auch hier beide Arbeitspunkte und erstellen zu jedem der beiden Arbeitspunkte Grafiken.

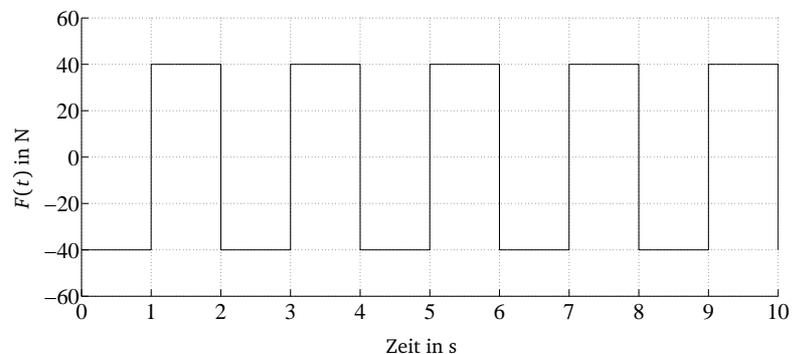


Abbildung 1.2: Rechteck-Signal $F(t)$.

1.2 WICHTIG: Hinweis zur Erstellung des Versuchsberichtes

Der Versuchsbericht (einer pro Gruppe) ist anhand der oben aufgeführten Fragen anzufertigen und innerhalb von einer Woche abzugeben.

Sowohl die Versuchsvorbereitung als auch die Versuchsdurchführung müssen vollständig im Bericht erscheinen. Dabei sollte ein besonderer Wert auf die Aussagekraft der Diagramme und der Screenshots gelegt werden. Die Simulationsergebnisse sollen als Diagramme mit der `plot`-Funktion (siehe Abschnitt Simulation, Erzeugen von Plots im Skript zu Versuch 1) dargestellt werden. Screenshots der *Scope*-Blöcke von Simulink sind für die schriftliche Dokumentation aus optischen Gründen ungeeignet.

Skalieren Sie die Diagramme sinnvoll, so dass das Ablesen bzw. Erkennen der interessierenden Größen möglich ist. In den Diagrammen ist eine eindeutige Achsenbezeichnung mit entsprechenden Größen und Einheiten unabdingbar. Bedenken Sie bei der Erstellung der Diagramme, dass diese auch schwarz/weiß ausgedruckt gut erkennbar und die einzelnen Graphen unterscheidbar sein müssen.

Strukturieren Sie die Protokolle so, dass auch nach einer längeren Zeitpause ein rascher Einstieg in die Thematik möglich ist. Dies kann im späteren Berufsleben eine erhebliche Zeitersparnis mit sich bringen.



Versuch 2

Steuerbarkeit und Beobachtbarkeit

2 Versuchsdurchführung

Die für die Durchführung der Versuche notwendige Zustandsraumdarstellung liegt bereits aus dem ersten Versuch vor (Modelle 1.0 und 1. π).

Zum Vergleich mit diesen Modellen soll die viskose Reibung des Pendelstabs vernachlässigt werden (Modelle 2.0 und 2. π). Des Weiteren soll sämtliche Reibung vernachlässigt werden (Modelle 3.0 und 3. π).

Aufgabe 2.1 (Durchführung/Nachbearbeitung):

Vergleichen Sie die Zustandsraummodelle miteinander:

- Notieren Sie die Matrizen **A** und **B** aller Modelle.

$$\mathbf{A}_{1.0} =$$

$$\mathbf{B}_{1.0} =$$

$$\mathbf{A}_{1.\pi} =$$

$$\mathbf{B}_{1.\pi} =$$

$$\mathbf{A}_{2.0} =$$

$$\mathbf{B}_{2.0} =$$

$$\mathbf{A}_{2.\pi} =$$

$$\mathbf{B}_{2.\pi} =$$

$$\mathbf{A}_{3.0} =$$

$$\mathbf{B}_{3.0} =$$

$$\mathbf{A}_{3.\pi} =$$

$$\mathbf{B}_{3.\pi} =$$

- Wodurch unterscheiden sich die Zustandsraumdarstellungen der Modelle?

Aufgabe 2.2 (Durchführung/Nachbearbeitung):

Vergleichen Sie die Eigenwerte der Zustandsraummodelle miteinander:

- Notieren sie die Eigenwerte aller Systeme.

$\lambda_{1,0} =$	$\lambda_{1,\pi} =$
$\lambda_{2,0} =$	$\lambda_{2,\pi} =$
$\lambda_{3,0} =$	$\lambda_{3,\pi} =$

- Welche Unterschiede liegen vor und worauf sind diese zurückzuführen? Gehen Sie dabei auf Realteil und Imaginärteil der Eigenwerte ein. Welche Aussagen über das Systemverhalten können Sie daraus ableiten?

Aufgabe 2.3 (Durchführung/Nachbearbeitung):

Normalformen des Zustandsraummodells:

- Schreiben Sie eine Funktion
 $[AD, BD, CD, DD] = \text{diagonalForm}(A, B, C, D)$
die ein gegebenes System in Diagonalform transformiert (wenn möglich) und dokumentieren Sie diese in ihrem **Protokoll**.
- Transformieren Sie das System 1.π mit dieser Funktion auf Diagonalform.

$A_D =$	$B_D =$
$C_D =$	$D_D =$

- Transformieren Sie das System 1.π mit der Funktion canon auf Diagonalform.

$\mathbf{A}_D =$	$\mathbf{B}_D =$
$\mathbf{C}_D =$	$\mathbf{D}_D =$

- Welche Unterschiede sind zu erkennen? Wie lassen Sie sich erklären?

- Transformieren Sie das System 1.0 auf Modalform.

$\mathbf{A}_D =$	$\mathbf{B}_D =$
$\mathbf{C}_D =$	$\mathbf{D}_D =$

Die nachfolgenden Aufgaben sind in Ihrem Protokoll zu beantworten.

Aufgabe 2.4 (Durchführung/Nachbearbeitung):

Die Zustandsraumbeschreibung des Schlitten-Pendel-Systems 1. π am oberen Arbeitspunkt soll nun anhand der beschriebenen Kriterien auf die vollständige Steuer- und Beobachtbarkeit untersucht werden. Hierzu sollen die angegebenen formellen Zusammenhänge verwendet werden. Schreiben Sie die Funktionen, so dass sie *möglichst allgemein* verwendbar sind, also z. B. Systeme beliebiger Ordnung verarbeiten können. Ebenfalls sollen von MATLAB zur Verfügung gestellte Funktionen angewendet werden.

- Prüfen Sie die Steuer- und Beobachtbarkeit des Systems 1. π nach Kalman. Schreiben Sie hierzu die Funktionen

- `checkCtrbKalman(A, B)` und
- `checkObsvKalman(A, C)`

und verifizieren Sie die Ergebnisse mit den von MATLAB zur Verfügung gestellten Funktionen `obsv` und `ctrb`. Dokumentieren Sie sowohl die beiden Funktionen als auch die Ergebnisse des Vergleichs in Ihrem Protokoll.

- Prüfen Sie die Steuer- und Beobachtbarkeit des Systems 1. π nach Gilbert. Schreiben Sie hierzu die Funktionen

- `checkCtrbGilbert(A, B)` und
- `checkObsvGilbert(A, C)`.

Dokumentieren Sie die beiden Funktionen in Ihrem Protokoll.

- Prüfen Sie die Steuer- und Beobachtbarkeit des Systems 1. π nach Hautus. Schreiben Sie hierzu die Funktionen

- `checkCtrbHautus(A, B)` und
- `checkObsvHautus(A, C)`.

Dokumentieren Sie die beiden Funktionen in Ihrem Protokoll.

- Welche Aussagen lassen sich über das System machen?
- Können die Kriterien auch auf die Systeme 2. π und 3. π angewendet werden?

Versuch 3

LQ-Regelung und Animation

Aufgabe 3.2 (Durchführung/Nachbearbeitung):

- Vervollständigen Sie die Funktion

```
function [A, B, C, D] = linPendelZR(stPendel, AP),
```

welche in Abhängigkeit der Pendeldata und des Arbeitspunkts (untere oder obere Ruhelage) die in Versuch 1 linearisierten Matrizen **A**, **B**, **C** und **D** zurückgibt. (AP kann dabei 0 oder π sein.)

Diese Funktion dient dazu, später schnell den Arbeitspunkt wechseln zu können, ohne anzufangen im Code Teile aus- und einzukommentieren.

Wichtig: Der Zustandsvektor soll folgendermaßen aufgebaut sein:

$$\mathbf{x} = \begin{bmatrix} \text{Weg Schlitten} \\ \text{Geschwindigkeit Schlitten} \\ \text{Winkel Pendel} \\ \text{Winkelgeschwindigkeit Pendel} \end{bmatrix}.$$

```
function [A, B, C, D] = linPendelZR(stPendel, AP)

    if ( )

        fprintf('-----\n');
        fprintf('Es wurde ein falscher Arbeitspunkt gewählt!!\n');
        fprintf('Möglicher Arbeitspunkt 0 bzw pi\n');
        fprintf('-----\n');

        A = 'Error';
        B = 'Error';
        C = 'Error';
        D = 'Error';

        return;
    end;

    % Bestimmung der ZR-Darstellung
    % Abkürzungen
    mP =
    mS =
    lP =
    g =

    % Abfrage um welchen Punkt linearisiert wird.
    switch (AP)
        case 0

            A =
```

```

        B =

        C =

        D =

    case pi

        A =

        B =

        C =

        D =

    end % switch (AP)

end % function linPendelZR

```

Aufgabe 3.3 (Durchführung/Nachbearbeitung):

- Vervollständigen Sie die Funktion

$$[K, \text{poleRK}] = \text{berechneLQR}(A, B, Q, R),$$

die die Reglermatrix K mittels des LQR-Verfahren berechnet und diese zurückgibt. In Hinblick auf Versuch 4 soll die Funktion ebenfalls die Pole des *geschlossenen* Regelkreises zurückgeben.

```

function [K, poleRK] = berechneLQR(A, B, Q, R)

% Fehlerabfragen:
K      = 'Error';
poleRK = [];

% Steuerbarkeit

```

```

% Test auf Symmetrie von Q:

% Test auf positive Definitheit von Q:

% Test auf positive Definitheit von R:

% Reglerberechnung:
[K, ~, poleRK] =
    % poleRK: Pole geschlossener Regelkreis

end % function berechneLQR

```

Aufgabe 3.4 (Durchführung/Nachbearbeitung):

- Implementieren Sie die Regelung in Simulink und vervollständigen Sie das unten stehende Strukturbild. Für das Schlitten-Pendel-System soll dabei das nichtlineare Modell verwendet werden. Verwenden Sie dazu die gegebene s-Function „systemPendel_V3“. Diese unterscheidet sich in folgenden Punkten von der s-Function, die Sie in Versuch 1 erstellt haben:
 - Es werden alle vier Zustände zurückgegeben. (In der in Aufgabe 2 angegebenen Reihenfolge!)
 - In den Parametern wird für die Pendeldaten die oben beschriebene Struktur benutzt.
 - Es wird keine viskose Reibung berücksichtigt.
 - Es wird in den Parametern auch der Anfangswert angegeben.

Natürlich können Sie diese Änderungen auch selber vornehmen. In Abbildung 3.1 ist die Verwendung gezeigt.

Das Modell sollte die Reglermatrix K , die Pendeldaten, den Arbeitspunkt und den Anfangszustand \mathbf{x}_0 aus dem Workspace lesen und den Verlauf der Zustände $\mathbf{x}(t)$ mit dem Zeitvektor in den Workspace schreiben.

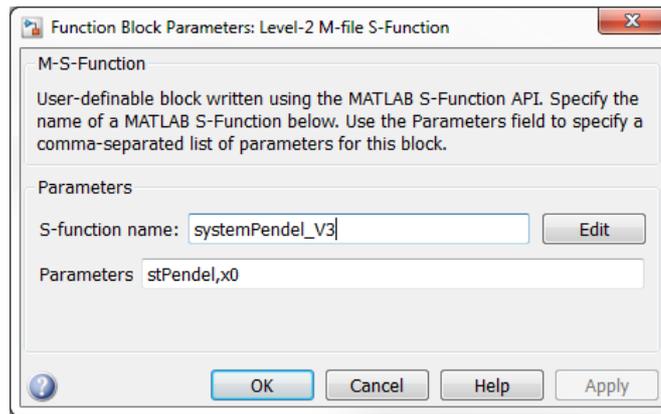


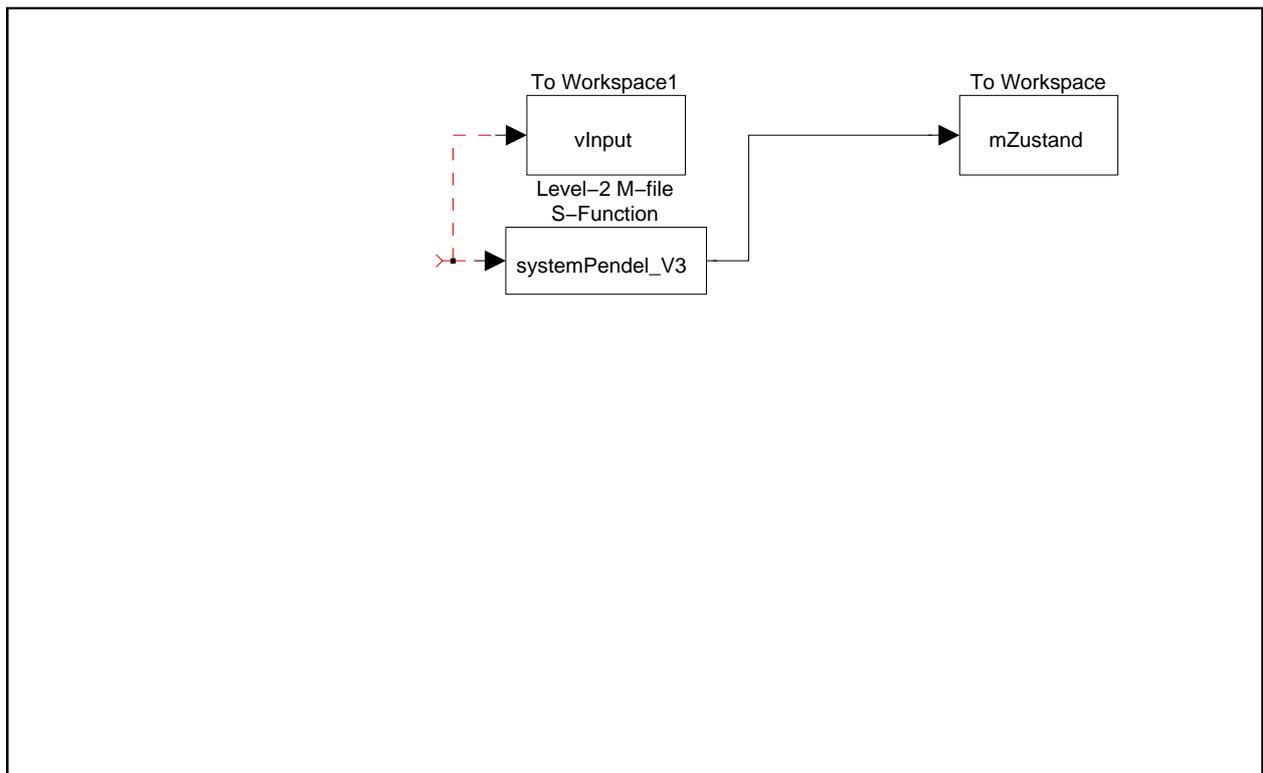
Abbildung 3.1: Verwendung der s-Funktion „systemPendel_V3“

Hinweise:

Überlegen Sie, welche Werte das Modell des Systems für die einzelnen Arbeitspunkte zurückgibt, und welche der Regler erwartet.

Um einen Sollgrößensprung auf das System zu geben, benutzen Sie die Anfangswerte. Geregelt wird das System immer in die Lage $x = 0$.

Testen Sie ihr Modell dadurch, dass Sie einen Anfangswert ungleich 0 vorgeben und interpretieren Sie den Ausgang.



Die nachfolgenden Aufgaben sind in Ihrem Protokoll zu beantworten.

Aufgabe 3.5 (Durchführung/Nachbearbeitung):

- Schreiben Sie eine Funktion

`[vT, mX] = runPendel(stPendel, AP, K, x0)`

die die Pendeldaten, die Reglermatrix und die Anfangswerte des Systems übergeben bekommt, und das Simulinkmodell ausführt. Diese Funktion soll den Verlauf der Zustandsgrößen mit dem dazugehörigen Zeitvektor zurückgeben.

Fügen Sie bitte (unkommentiert) den Code Ihrer Funktion `runPendel` Ihrem Protokoll bei.

Aufgabe 3.6 (Durchführung/Nachbearbeitung):

- Schreiben Sie eine Funktion

`animierePendel(vT, mX, stPendel, hAxes)`

zur Animation des Pendelschlittens. Diese Funktion soll den Verlauf der Zustandsgrößen sowie den dazugehörigen Zeitvektor und die Pendeldaten übernehmen. Zudem soll diese Funktion in Hinblick auf Übung 4 auch das Handle des Achsensystems übernehmen, in das die Animation gezeichnet werden soll. Wenn diese Variable leer ist, dann soll ein neues Figure mit Achsensystem erstellt werden.

Testen Sie Ihre Funktion mit Simulationsergebnissen.

Hinweise:

Mit `isempty(Variablenname)` kann überprüft werden, ob eine Variable eine leere Matrix enthält.

Orientieren Sie sich beim Programmieren an dem Beispiel `animierePunkt` von Listing 11.3.

Es wäre praktisch, im Titel des Achsensystems die aktuelle Simulationszeit anzuzeigen.

Betrachten Sie sich den Zeitvektor `vT`. Wurde ein Löser mit variabler Schrittweite verwendet, so sind die Simulationszeitpunkte nicht äquidistant, die Pause zwischen zwei Bildern der Animation müsste für jedes Bild neu bestimmt werden. Es ist sinnvoll, die Daten für die Animation in einer vernünftigen konstanten Bildrate vorliegen zu haben. Hierzu kann die Funktion `xi = interp1(x,y,xi)` verwendet werden. Ein Vektor mit äquidistanten Zeitpunkten kann leicht mit `vTAnim = 0:pause:vT(end)` erzeugt werden.

In Abbildung 3.2 ist beispielhaft gezeigt, wie so eine Animation aussehen könnte. Es muss allerdings nicht so aufwendig (mit Kästen für Schlitten) ausgeführt werden, es reicht auch eine einzige bewegte Linie als Pendelstab aus.

Aufgabe 3.7 (Durchführung/Nachbearbeitung):

- Verändern Sie die Einträge in den Matrizen **Q** und **R** und interpretieren Sie das sich daraus ändernde Systemverhalten. Achten Sie hierbei darauf, dass die Anforderungen an die Matrizen **Q** und **R** (Symmetrie, positive Definitheit) stets erfüllt sind.

Hinweise:

Um die Einflüsse der einzelnen Parameter auf den Reglerentwurf beurteilen zu können, sollten Sie nur jeweils einen Parameter verändern.

Für die direkte Interpretation der Ergebnisse stellt die Animation eine Hilfe dar. Für das Protokoll sind jedoch Plots der Zustände über der Zeit sinnvoller.

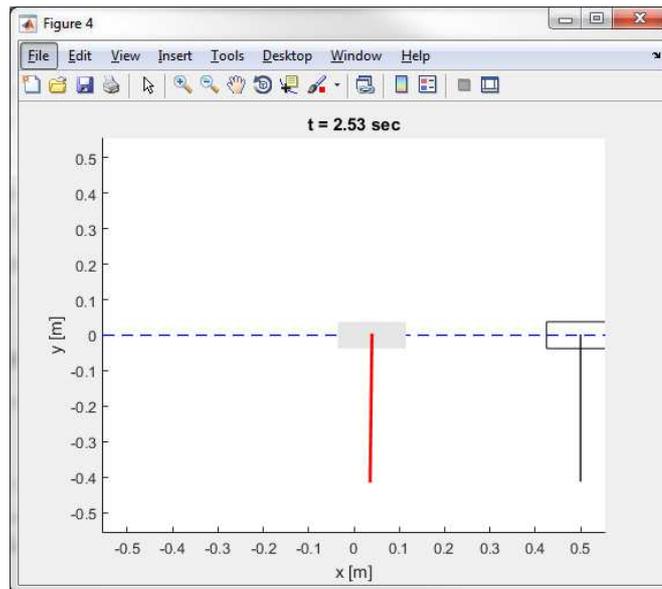


Abbildung 3.2: Beispiel für Animation

Simulieren Sie das geregelte System für die untere und obere Ruhelage und wählen Sie jeweils mindestens zwei aussagekräftige Simulationsergebnisse bei verschiedenen Gewichtungsmatrizen aus. Stellen Sie die Simulationsergebnisse von $F(t)$, $x_s(t)$ und $\varphi(t)$ dar und kommentieren Sie diese.

Aufgabe 3.8 (Durchführung/Nachbearbeitung):

- Erweitern Sie Ihre Funktion `animierePendel`, so dass, wenn gewünscht, ein avi-Video generiert wird. Dazu können Sie ein weiteres Argument hinzufügen. Bedenken Sie dabei, dass die Funktion dennoch mit dem oben angegebenen Aufruf funktionieren sollte, da dies im nächsten Versuch benötigt wird.

Hinweis:

Mit dem Befehl `nargin` können Sie in einer Funktion die Anzahl der tatsächlich übergebenen Parameter bestimmen.

Fügen Sie bitte (unkommentiert) den Code Ihrer Funktion `animierePendel` und einen Screenshot der Animation Ihrem Protokoll bei.

Versuch 4

Beobachterentwurf – Benutzeroberflächen

4 Versuchsdurchführung

4.1 GUI 1. Teil

Aufgabe 4.1 (Durchführung/Nachbearbeitung):

- Kopieren Sie die gegebenen Dateien „simGUI.fig“ und „simGUI.m“ in ein Verzeichnis mit den im letzten Versuch erstellten Programmen.

Wenn Sie sich im letzten Versuch an die vorgegebenen Funktionsnamen und Syntax gehalten haben, dann sollte es möglich sein, die gegebene GUI zu starten und den Button „Berechne K“ zu betätigen.

Die berechneten Werte für K sollten dann im entsprechenden Feld angezeigt werden. Falls nicht, müssen Ihre Funktionen entsprechend angepasst werden.

Aufgabe 4.2 (Durchführung/Nachbearbeitung):

- Erweitern Sie die vorgegebene GUI in einem ersten Schritt dadurch, dass Sie neben den Reglerparametern auch die Pole des geschlossenen Regelkreises in einem Textfeld ausgeben.

Aufgabe 4.3 (Durchführung/Nachbearbeitung):

- Erweitern Sie die vorgegebene GUI in einem weiteren Schritt durch
 - Eingabefelder für die Anfangswerte des Systems und
 - Hinzufügen eines Buttons, durch den die Simulation ausgeführt wird. Die Animation des simulierten Systems soll in dem schon vorhandenen Achsensystem `axes1` gezeigt werden.

Hinweis: Hierbei sollen natürlich die Funktionen aus Versuch 3 verwendet werden.

Aufgabe 4.4 (Durchführung/Nachbearbeitung):

- Fügen Sie einen Button STOP hinzu, mit dem Sie den Ablauf der Animation stoppen können.

Hinweis: Sie können hierzu eine globale Variable verwenden, die in der Callback-Funktion des Stop-Button gesetzt wird und in der `for`-Schleife abgefragt wird.

4.2 Beobachter

Aufgabe 4.5 (Durchführung/Nachbearbeitung):

- Schreiben Sie eine Funktion

```
function L = berechneBeobachter(A, C, poleBeobachter)
```

die den Beobachter nach vorgebbaren Polen auslegt.

Aufgabe 4.6 (Durchführung/Nachbearbeitung):

- Erweitern Sie Ihr Simulink-Modell aus Versuch 3, so dass alternativ (über eine Variable wählbar) die Regelung über Zustandsrückführung (Versuch 3) oder Ausgangsrückführung und Beobachter erfolgt. Dazu könnten Sie bspw. einen „Switch“-Block benutzen.
- Erweitern sie Ihre Funktion `runPendel` so, dass alternativ der Beobachter verwendet werden kann. Die neue Syntax der Funktion könnte so aussehen

$$[vT, mX, mXobs] = \text{runPendel}(\text{stPendel}, AP, K, x0, \text{stObs})$$

wobei die Struktur `stObs` alle notwendigen Daten enthalten würde, um den Beobachter in Simulink benutzen zu können. Die Entscheidung, ob der Beobachter verwendet werden soll (und keine Zustandsrückführung) würde in diesem Fall durch die Anzahl der tatsächlich übergebenen Argumente (5 oder 4) getroffen werden.

Der Rückgabewert `mXobs` enthält die geschätzten Zustände.

- Testen Sie die neue Funktionalität zunächst ohne GUI.

Hinweise:

Wird der Beobachter verwendet, muss der Ausgang der s-Function transformiert werden, da dieser alle Systemzustände enthält. Dazu könnte entweder der Ausgang mit einer geeigneten Matrix multipliziert oder der Block „Selector“ aus der Gruppe „Signal Routing“ verwendet werden.

Die Funktion `runPendel` soll alle notwendigen Schritte zum Starten des Simulink-Modells beinhalten. D. h. diese Funktion sollte auch funktionieren, wenn vorher alle Variablen im Workspace gelöscht werden.

4.3 GUI 2. Teil

Aufgabe 4.7 (Durchführung/Nachbearbeitung):

- Erweitern Sie Ihre GUI um eine Auswahlmöglichkeit „Beobachter oder Zustandsrückführung“. Sehen Sie außerdem Eingabefelder für die Beobachtereigenwerte und die Anfangswerte für den Beobachter vor.
- Erweitern Sie die Callback-Funktion der Schaltfläche derart, dass je nach Wahl ggfs. die Beobachtermatrix `L` berechnet und die Funktion `runPendel` entsprechend aufgerufen wird.

In Abbildung 4.1 ist ein Beispiel gezeigt, wie die Oberfläche am Ende aussehen könnte.

4.4 Einfluss des Beobachters

Aufgabe 4.8 (Durchführung/Nachbearbeitung):

- Untersuchen Sie anhand von ein paar Beispielen und geeigneten Graphen den Einfluss des Beobachters und der Beobachtereigenwerte auf die geschätzten Zustände und das Verhalten des Regelkreises.

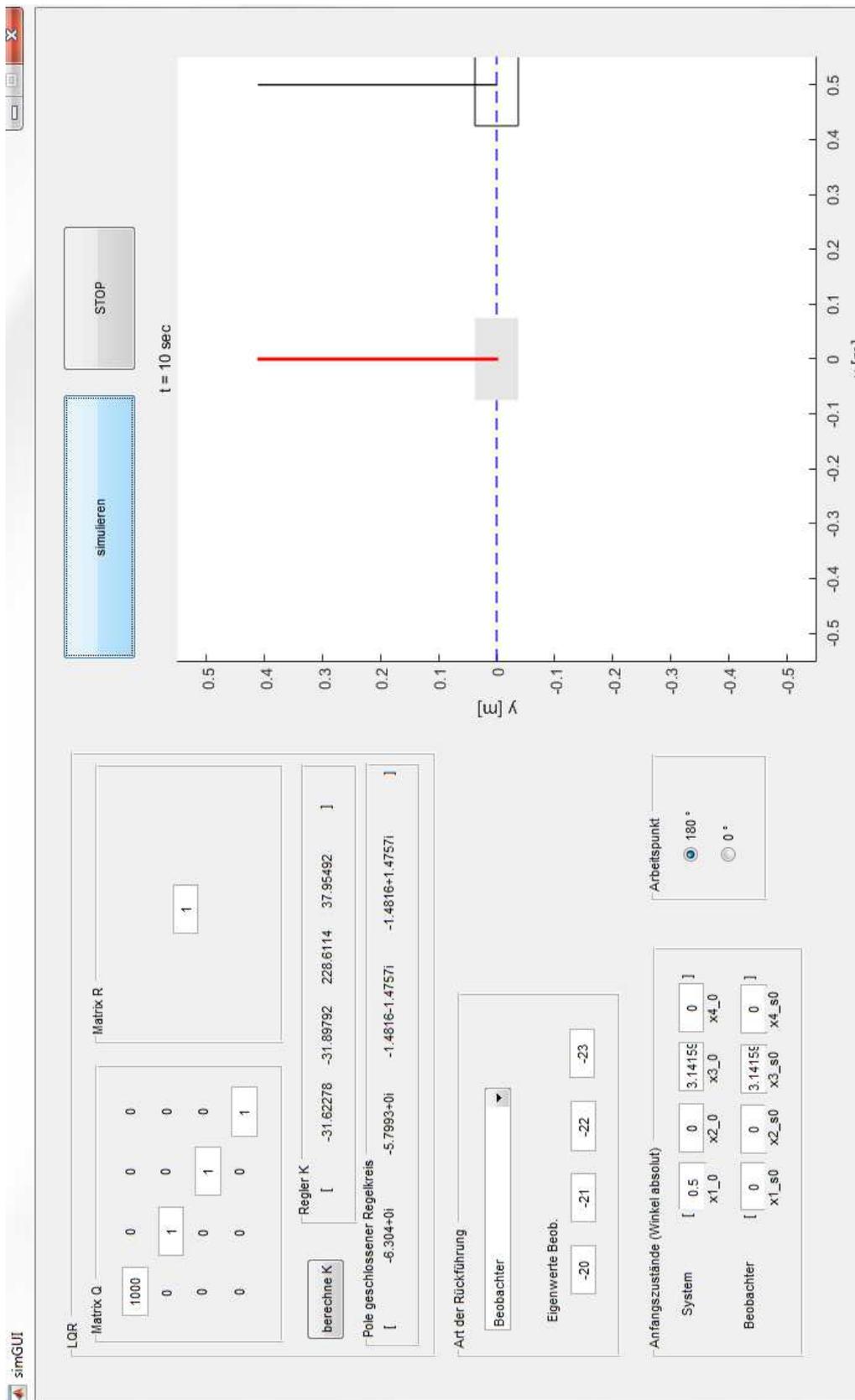


Abbildung 4.1: Mögliches Aussehen der GUI mit allen Funktionen

-
- Was fällt bei den Werten für die gemessenen Zustände x_1 und x_3 auf? Wie könnte das Verhalten verbessert werden?

4.5 Protokoll

Das Protokoll soll die Ergebnisse der Aufgabe 8 beinhalten.

Fügen Sie bitte außerdem

- alle Screenshots des Simulink-Modells mit Beobachter,
- sowie (unkommentiert) den Quellcode der relevanten Callback-Funktionen,
- sowie der Funktionen `berechneBeobachter` und `runPendel`

Ihrem Protokoll bei.



Versuch 5

Aufschwungsteuerung für das Pendel

5 Versuchsdurchführung

5.1 Berechnen von $u^*(t)$ und $x^*(t)$

5.1.1 Programmierung

Aufgabe 5.1 (Durchführung/Nachbearbeitung):

- Erstellen Sie eine Funktion

`stTraj = berechneTrajektorie(stPendel, T)`

die aus den Pendeldaten (Struktur wie in Versuch 3 und 4) und der vorzugebenden Übergangszeit T die Verläufe der Eingangsgröße $u^*(t)$ und der Zustände $x^*(t)$ nach dem vorgestellten Verfahren mit `bvp4c` berechnet. Es bietet sich an, als Rückgabewert eine Struktur zu haben, in denen der Zeitvektor t mit der Steuerfolge u und den Zuständen x zurückgegeben werden. Außerdem könnte noch explizit die Übergangszeit T angegeben werden:

```
stTraj.T
      .vT
      .vU
      .mX
```

Um diese Funktion zu realisieren, können Sie sich an folgenden Schritten orientieren:

- Schreiben Sie eine Funktion `RandwertproblemDGL`, die das DGL-System (16.6) des Randwertproblems beschreibt:

`dx = RandwertproblemDGL(t, x, P, stPendel, T)`

Die Funktion hat als Eingangsparameter zum eine die von `bvp4c` geforderten Daten: Zeit t , Zustandsvektor x und den Vektor P (enthält die freien Parameter p_1 und p_2). Zudem sollen dieser Funktion auch die Struktur mit den Pendeldaten und die gewünschte Übergangszeit T übergeben werden können. Da `bvp4c` nur die ersten drei Parameter benutzt, würde dies zum Absturz führen, d. h. es muss möglich sein, diese Funktion auch mit drei Parametern aufzurufen. Im Abschnitt 17.3 wurden dazu zwei Möglichkeiten vorgestellt, von denen Sie sich eine aussuchen können. Stimmen Sie die Funktion `RandwertproblemDGL` entsprechend auf Ihre Wahl ab.

- Schreiben Sie eine Funktion `RandwertproblemRB`:

`deltaRB = RandwertproblemRB(x0, xT, P)`

Sie gibt einen Spaltenvektor `deltaRB` mit der Abweichung der Werte bei $t = 0$ und $t = T$ zu den gegebenen Randwerten (16.7) (Auch wenn im vorliegenden Problem die Randwerte unabhängig von den Parametern p_1 und p_2 sind, muss der Parametervektor P als Argument angegeben werden, da dieser vom Löser übergeben wird.)

- Erstellen Sie in der Funktion `berechneTrajektorie` die Struktur `solinit` mit den Startwerten für den Löser. Geben Sie zunächst eine Aufteilung für t in 1000 Intervallen an. Für $y(t)$ geben Sie den linearen Verlauf der beiden Zustände von Anfangs- zum Endwert vor. Für p_1 und p_2 geben Sie jeweils Null vor.

-
- Lösen sie das Randwertproblem mit `bvp4c`.
 - Berechnen bzw. entnehmen Sie aus der von `bvp4c` zurückgegebenen Struktur den Verlauf für die Eingangsgröße $u^*(t)$ sowie den Verlauf aller vier Zustände $\mathbf{x}^*(t)$ und geben Sie diese in der oben angegebenen Struktur `stTraj` zurück.

5.1.2 Auswertung

Aufgabe 5.2 (Durchführung/Nachbearbeitung):

- Wählen Sie zum Testen Ihrer Funktion zunächst für T einen Wert zwischen 1,2 und 1,5 s.
- Betrachten Sie für verschiedene Übergangszeiten T die Trajektorie $y^*(t)$ und die Verläufe von $u^*(t)$ und $\mathbf{x}^*(t)$ und überprüfen Sie die Lösung. Plotten sie den Verlauf der Zustände $\mathbf{x}^*(t)$ und der Stellgröße $u^*(t)$.
- In welchem Wertebereich für T können Lösungen gefunden werden? Begründen sie über die Anschauung, ob das Nichtauffinden von Lösungen für große T am Löser oder an der Physik liegen wird.
- Versuchen Sie durch Variation der Übergangszeit T die Beschränkungen

$$|x| = 0,5 \text{ m}$$

$$|\dot{x}| = 1,6 \text{ m/s}$$

$$|\ddot{x}| = 12 \text{ m/s}^2$$

möglichst knapp einzuhalten. (x ist die Position des Schlittens.)

5.2 Simulation

Aufgabe 5.3 (Durchführung/Nachbearbeitung):

- Bauen Sie die Steuerung unter Simulink auf.

Benutzen Sie die s-Function `systemPendel_V5` als Modell für das System. Sie beschreibt das Schlitten-Pendel-System nach (16.2). (Im Unterschied zu `systemPendel_V3` wird hier als Eingang u die Beschleunigung des Schlittens und nicht die Kraft auf den Schlitten verwendet.) Als Parameter werden die Pendelraten und die Anfangsbedingung des Systems erwartet (siehe Abbildung 5.1). Der Ausgang dieser s-Function entspricht wie auch bei `systemPendel_V3` den Zuständen.

5.2.1 Auswertung

Aufgabe 5.4 (Durchführung/Nachbearbeitung):

- Vergleichen Sie anhand der Zeitplots (und der Animation) das Ergebnis der Simulation mit den theoretisch berechneten Trajektorien. Lassen Sie die Simulation etwa eine Sekunde länger als die Übergangszeit T laufen.

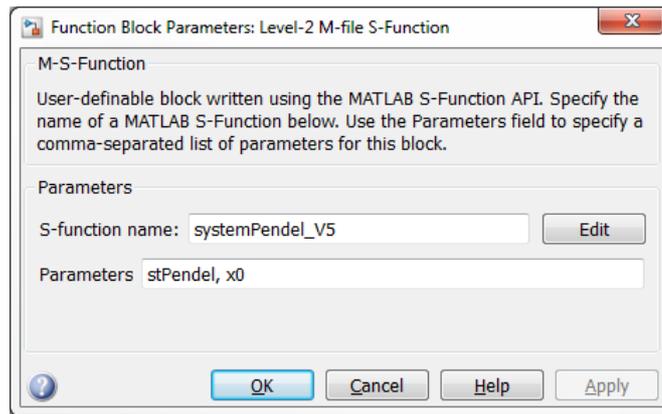


Abbildung 5.1: Verwendung der s-Function systemPendel_V5

- Variieren Sie bei der Simulation auch die Pendelparameter. (Berechnen Sie aber nicht eine neue Steuerung!) Welche Pendelparameter haben einen großen Einfluss auf das Ergebnis?

5.3 Protokoll

Das Protokoll soll die unter den Punkten „Auswertung“ angesprochenen Themen behandeln und die gestellten Fragen beantworten.

Außerdem fügen Sie bitte einen Screenshot des Simulink-Modells sowie (unkommentiert) den Code Ihrer Funktionen dem Protokoll bei.

Versuch 6

Trajektorienfolgeregelung

6 Versuchsdurchführung

6.1 Linearisierung

Es wird auch in diesem Versuch (zunächst) von den im Skript zum letzten Versuch gegebenen Systemgleichungen

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ 0 \\ x_4 \\ -\frac{m_p \frac{l}{2} g \sin x_3}{m_p \frac{l^2}{4} + J} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ -\frac{m_p \frac{l}{2} \cos x_3}{m_p \frac{l^2}{4} + J} \end{bmatrix} u \quad (6.1)$$

$$y = x_1 \quad (6.2)$$

ausgegangen ($J = \frac{1}{12} m_p l^2$). D. h. die Eingangsgröße ist die Beschleunigung des Schlittens.

Bestimmen Sie analytisch die Matrizen

$$\mathbf{A}(\mathbf{x}, u) = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}, u} \quad (6.3)$$

$$\mathbf{B}(\mathbf{x}, u) = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_{\mathbf{x}, u} . \quad (6.4)$$

Schreiben Sie eine Funktion

$$[\mathbf{A}, \mathbf{B}] = \text{linPendelZR2}(\text{stPendel}, \mathbf{x}, u)$$

die die Matrizen $\mathbf{A}(\mathbf{x}, u)$ und $\mathbf{B}(\mathbf{x}, u)$ des um den gegebenen Arbeitspunkt \mathbf{x} und u linearisierten Systems zurückgibt.

6.2 Berechnen von $\mathbf{K}(t)$

Schreiben Sie eine Funktion

$$\text{vPdot} = \text{RiccatiDGL}(t, \text{vP}, \text{stPendel}, \text{stTraj}, Q, R)$$

die die RICCATIdifferentialgleichung (20.10) implementiert.

Diese Funktion muss vom DGL-Löser in der Form $\text{vPdot} = \text{RiccatiDGL}(t, \text{vP})$ aufrufbar sein. (Methoden dazu wurden in Versuch 5 vorgestellt.)

Dazu müssen in der Funktion `RiccatiDGL` folgende Schritte implementiert werden:

- Aus den in `stTraj` gespeicherten Daten der berechneten Trajektorie den zum Zeitpunkt t gehörenden Sollzustand und Eingangswert ermitteln (interpolieren).

- Die Systemmatrizen des linearisierten Modelles mit `linPendelZR2` berechnen.
- $\dot{\mathbf{P}}$ mit (20.10) berechnen. Dabei beachten, dass die Matrix \mathbf{P} in `vP` als Vektor übergeben wird und die Matrix $\dot{\mathbf{P}}$ als Vektor in `vPdot` zurückgegeben werden muss.

Erstellen Sie eine Funktion

$$[\mathbf{vTK}, \mathbf{mK}] = \text{berechneK}(\text{stPendel}, \text{stTraj}, Q, R)$$

die in `mK` den Reglervektor für die in `vTK` angegebenen Zeitpunkten zurückgibt.

Dazu müssen in dieser Funktion folgende Schritte implementiert werden:

- Berechnen des Anfangswertes $\mathbf{P}(T)$ durch Lösen von (20.11).
- Berechnen des Verlaufs von $\mathbf{P}(t)$ durch Lösen der RICCATI-Differentialgleichung (20.10). (Hierzu wird die Funktion `RiccatiDGL` benötigt.)
- Berechnen des Verlaufes von $\mathbf{K}(t)$ mit (20.9).

6.3 Folgeregelung unter Simulink aufbauen

Erweitern Sie Ihr Simulink-Modell aus der vorherigen Übung um die Folgeregelung.

Es ist empfehlenswert, das Modell so aufzubauen, dass die Regelung schnell über eine Variable oder einen manuellen Schalter aktiviert und deaktiviert werden kann. So kann das unterschiedliche Verhalten schnell verglichen werden.

Simulieren Sie auch jetzt etwa eine Sekunde länger als die Übergangszeit T . Verändern Sie auch wieder die Pendelparameter des Simulationsmodells (ohne die Steuerung und Regelung neu zu berechnen).

Welche Unterschiede stellen Sie beim Vergleich der reinen Steuerung und der Steuerung mit Folgeregelung fest?

6.4 Einfluss des Antriebs

Bei der Berechnung der Trajektorien und zur Bestimmung der zeitvarianten Reglermatrix $\mathbf{K}(t)$ wurde zur Vereinfachung angenommen, dass der Systemeingang u des Schlitten-Pendel-Systems die Schlittenbeschleunigung \ddot{x} ist. Praktisch erfolgt die Steuerung/Regelung jedoch über einen Elektromotor, der eine Spannung U als Eingangsgröße besitzt und eine Kraft F auf den Schlitten ausübt.

Um diesen Motor möglichst einfach zu berücksichtigen, verwenden Sie in einem ersten Schritt wieder die `s`-Function `systemPendel_V3` aus Versuch 3 und 4, die als Eingangsgröße die Kraft F erwartet. Die Steuerung (die nicht neu hergeleitet/berechnet werden soll) gibt aber natürlich immer noch eine Beschleunigung u^* vor, und auch der Reglerausgang Δu ist eine Beschleunigung. Um also die durch Steuerung und Regler geforderte Beschleunigung $u(t) = u^*(t) + \Delta u(t)$ in eine Kraft umzurechnen, multiplizieren Sie diese mit der Gesamtmasse des Schlitten-Pendel-Systems:

$$F(t) = u(t) \cdot (m_s + m_p).$$

(Dabei sind die Massen zu benutzen, die auch zur Berechnung der Trajektorien benutzt wurden.)

Vergleichen Sie das neue Systemverhalten mit dem aus Abschnitt 6.3, wieder ohne und mit Trajektorienfolgeregelung.

Im letzten Versuch 5 hat sich gezeigt, dass die Pendelmasse $m = m_p$ keinen Einfluss auf die Steuerung hat, wenn der Systemeingang die Beschleunigung ist. Wie verhält es sich wenn die Kraft F der Systemeingang ist?

Ist die Umrechnung der Schlittenbeschleunigung u in die Kraft F , die auf den Schlitten aufgebracht werden muss, exakt, wenn davon ausgegangen wird, dass die Pendelparameter m_s und m_p genau bekannt sind?

Tatsächlich wird aber auch keine Kraft vorgegeben, sondern die Spannung U am Motor. Ein Elektromotor ist aber wiederum ein dynamisches System. Modellieren Sie dieses näherungsweise durch ein PT_1 -Glied mit der Zeitkonstanten T_M .

Probieren Sie für verschiedene Werte für T_M aus, wie sich das gesteuerte System ohne und mit Folgeregelung verhält.

6.5 Protokoll

Gehen Sie bitte auf die angesprochenen Punkte und Fragen ein. Verwenden Sie zur Erläuterung geeignete Plots.

Fügen Sie bitte Screenshots der beiden Simulink-Modelle (mit `systemPendel_V3` und `systemPendel_V5`) sowie (unkommentiert) Ihre in diesem Versuch erstellen Funktionen.

Literaturverzeichnis

- [1] ABEL, D. und A. BOLLIG: *Rapid Control Prototyping, Methoden und Anwendungen*. Springer, 2006.
- [2] MARKERT, R.: *Technische Mechanik, Teil B*. Fachbereich Mechanik, Technische Universität Darmstadt, 2002.