Regelung eines inversen Pendels

Aufgabenstellung

Freiwilliges praktisches Projekt
Systemtheorie II

Ansprechpartner:

Manuel Pitz (manuel.pitz@eonerc.rwth-aachen.de)
Systemtheorie II Team (acs-teaching-sys2@eonerc.rwth-aachen.de)

Version: 20.11.2019

Inhaltsverzeichnis

1 Einleitung

Vorsicht: Es besteht Quetschungsgefahr. Stellen Sie zu allen Zeiten sicher, dass sich der an dem Motor befestigte Hebelarm frei bewegen kann.

Verwenden Sie die freie Fläche hinter dem Arduino, um den Aufbau mit einem Gewicht (z.B. einem Buch) zu stabilisieren.

Dieses Projekt ist zum Verständnis und zur Vertiefung der in der Vorlesung und Übung Systemtheorie II erlernten Inhalte gedacht. Bei erfolgreichem Bearbeiten des Projekts erhalten die Studierenden Bonuspunkte im Umfang von 0.3 Notenstufen für die Klausur Systemtheorie II im Wintersemester 2019/2020 bzw. im Sommersemester 2020. Die Bonuspunkte kommen nur zur Geltung, wenn die Klausur auch ohne Anrechnung der Bonuspunkte bestanden wurde. Teilweise gehen die Inhalte der Aufgaben über diejenigen der Vorlesung hinaus. In solchen Fällen werden entsprechende Hinweise oder Lösungshilfen gegeben.

Bei technischen Problemen können Sie sich gerne an Herrn Manuel Pitz (manuel.pitz@eonerc.rwth-aachen.de) wenden. Bei organisatorischen Rückfragen wenden Sie sich bitte an das Systemtheorie II Lehrteam (acs-teaching-sys2@eonerc.rwth-aachen.de).

Alle zur Verfügung gestellten Komponenten müssen unbeschädigt bis spätestens zum **31. Januar 2020** wieder am E.ON Energy Research Center in der Mathieustraße 10 bei Herrn Manuel Pitz abgegeben werden. **Es werden definierte Zeitfenster für die Rückgabe bekannt gegeben**. Bei Beschädigung oder Verlust der zur Verfügung gestellten Komponenten werden die Kosten zur Neubeschaffung dem Verursacher in Rechnung gestellt!

Die Lösung ist in schriftlicher Form pro Gruppe bestehend aus maximal 3 Studierenden bis zum **31. Januar 2020** einzureichen. Die Lösung muss je nach Aufgabenteil eine nachvollziehbare Lösung (mit Lösungsweg!), Screenshots von Plots, Simulink Schaltpläne und/oder Beschreibungen enthalten. Der Gesamtumfang der einzureichenden Dokumente soll 10 DIN A4 Seiten nicht überschreiten. Es wird ein entsprechendes Template in RWTH Moodle bereitgestellt. Zusätzlich zu der Ausarbeitung ist ein von jedem Gruppenmitglied zu unterschreibendes Deckblatt einzureichen.

Die Bekanntgabe über das Bestehen oder Nichtbestehen des Projekts erfolgt zusammen mit der Veröffentlichung der vorläufigen Klausurergebnisse vor Einsicht.

Wir wünschen Ihnen Vergnügen und Erfolg bei der Durchführung des Projekts!

1.1 Versuchsbeschreibung

In dieser Aufgabe soll mit Matlab Simulink eine Regelung für ein inverses Pendel ausgelegt, implementiert und dann in einem *Hardware in the Loop* (HiL) Aufbau getestet werden. Die Regelung wird in drei Schritten in die Praxis umgesetzt. Zuerst wird eine Steuerung des Motors ausgelegt und in Matlab Simulink implementiert. Im zweiten Schritt wird die Regelung der oberen Gleichgewichtslage durch einen PD-Regler entwickelt. Als letztes wird ein Zweipunktregler für das Aufschwingen des Pendels implementiert. Es wird empfohlen, die Aufgaben in dieser Reihenfolge zu bearbeiten.

Es wird ein Arduino und ein Servomotor[1], montiert auf einer Kunststoffplatte (Abbildung 1), bereitgestellt. An dem Servomotor ist ein Hebelarm mit Gewichten befestigt. Die Gewichte können sowohl verringert als auch vollständig abgenommen werden. Der Motor ist elektrisch in seinem maximalen Drehmoment begrenzt, um eine Zerstörung bei Überlastung zu vermeiden. Das Überschreiten der Drehmomentgrenze hat einen sogenannten *Brownout* der Motorsteuerung zur Folge. Bei einem Brownout schaltet die Motorsteuerung ab und nach wenigen Sekunden wieder ein. Der Brownout ist zu vermeiden, da sich der Motor trotz Schutzbeschaltung stark erhitzen kann!

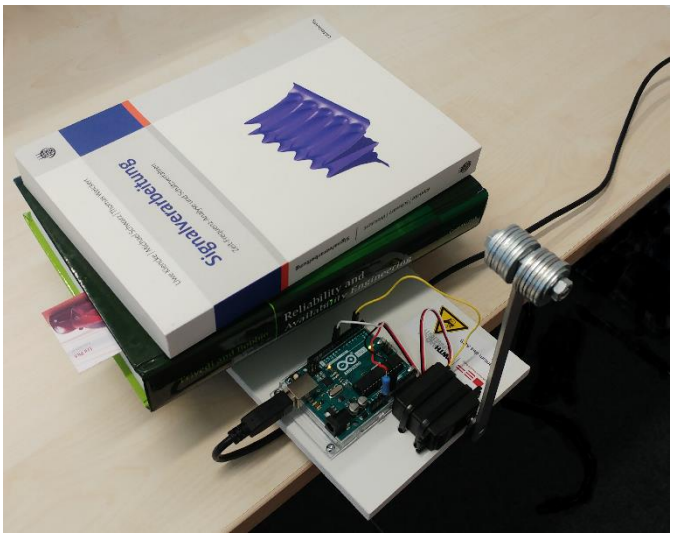


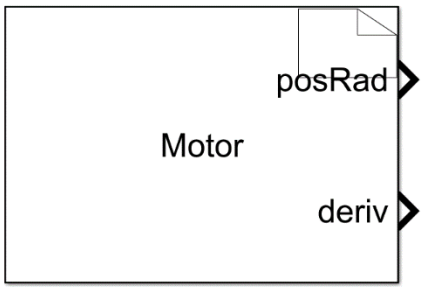
Abbildung 1: Inverses Pendel

[1] https://www.mouser.de/datasheet/2/321/900-00360-Feedback-360-HS-Servo-v1.2-1147206.pdf

1.2 Bereitgestellte Simulink Blöcke

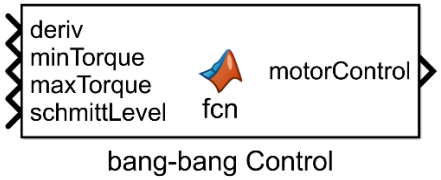
Die folgenden Simulink Blöcke werden im Online-Lernraum zum Download bereitgestellt.

1.2.1 Motor



Der *motorIn*-Block stellt die relative Position der Achse in Rad an *posRad* und den aktuellen Wert der Ableitung an *deriv* bereit. Der *motorIn*-Block filtert die Messung intern. Dies führt zu einer drehzahlabhängigen Verringerung der maximalen Positionsamplitude. Dieses Verhalten kann vernachlässigt werden.

1.2.2 bang-bang Control



Der bang-bang Control-Block schaltet den Ausgang zwischen dem an *minTorque* und *maxTorque* anliegenden Wert in Abhängigkeit von der Ableitung (*deriv* Eingang) um. Zusätzlich wird in dem Block ein Schmitt-Trigger implementiert, welcher das Umschalten aufgrund von Rauschen verhindert. Die beiden Schwellwerte des Schmitt-Triggers entsprechen *schmittLevel* und der Negation von *schmittLevel*. Mit Hilfe dieses Blocks kann ein Aufschwingen des Pendels realisiert werden.

Hinweis: Die beiden Blöcke können jeweils kopiert werden und in einem Blank-Model zusammengeführt werden.

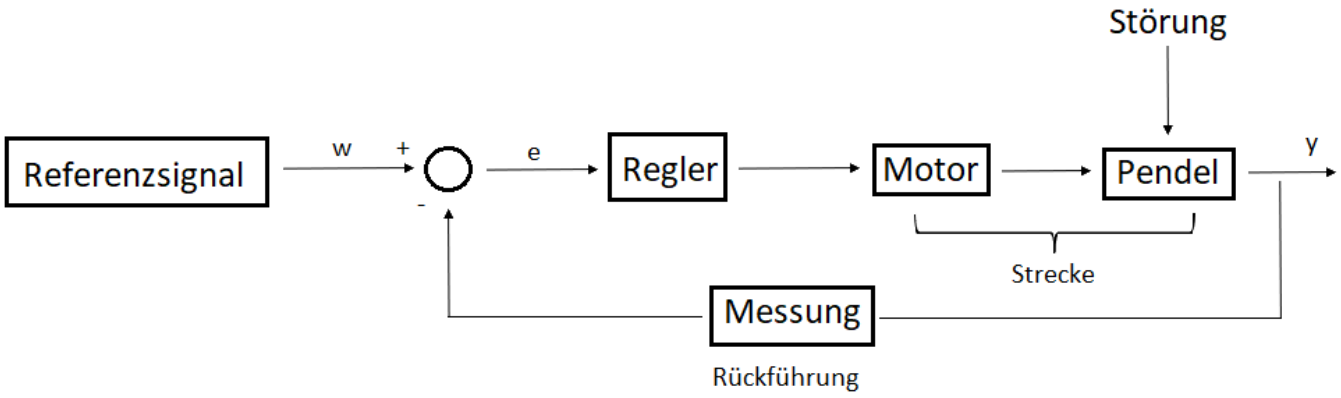
2 Aufgabenstellung

2.1 Aufbau und Motorsteuerung

Verbinden Sie den Servo-Motor entsprechend der folgenden Tabelle mit dem Arduino:

Servo-Motor	**Arduino**
Rot mit Widerstand	+5V
Schwarz	GND
Weis	PWM Ausgang
Gelb	Pin 3

2.1.1 Modellbildung



Der oben gezeigte Regelkreis soll die Aufgabe in diesem Projekt veranschaulichen. Für eine Untersuchung des inversen Pendels benötigt man im Wesentlichen ein Pendel und eine Messeinrichtung zur Detektion der Pendelposition (Winkelposition). Über eine Steuereinheit werden die Messsignale eingelesen und verarbeitet und dadurch der Motor angesteuert. Der Motor treibt das Pendel an. Der Regler soll in Simulink erarbeitet werden.

2.2 Rotation des Pendels

Für diesen Aufgabenteil wird der Motor ohne Gewichte verwendet. Es werden die entsprechenden Simulink-Blöcke für die Motorsteuerung sowie die Auswertung der Position in der MATLAB-Simulink-Umgebung miteinander verbunden.

Hinweise:

Verwenden Sie die Fläche hinter dem Arduino, um den Aufbau mit einem Gewicht (z.B. einem Buch) zu stabilisieren (siehe Abbildung 1).

Für die Messung der aktuellen Motorposition wird ein angepasster Simulink-Block (motorIn) bereitgestellt (Kapitel 1.2.1). Dieser kann ohne Änderungen verwendet werden. Beachten Sie, dass Pin 3 für das Positionssignal verwendet wird.

Zur Motorsteuerung kann der Arduino-Simulink-Block „**Arduino Continuous Servo Write**“ verwendet werden.

Die Drehmomenteinstellung kann aufgrund der Hardware unsymmetrisch sein.

1. Wählen und verbinden Sie die benötigten Blöcke, um eine Rotation zu erreichen.
2. Machen Sie sich mit dem Positionssignal vertraut (Form, Frequenz …) und notieren Sie einige Stichpunkte.
3. Prüfen Sie, ob das Positionssignal im Bereich der unteren Ruhelage maximal ist. Falls nicht, wenden Sie sich bitte an einen Betreuer des Projekts.
4. Untersuchen und dokumentieren Sie das Verhalten der Steuergröße:
    a. Welcher maximale/minimale Wert kann vorgegeben werden, bevor Brownout eintritt?
    b. Sind der maximale/minimale Wert betragsmäßig gleich? Begründen Sie Ihre Antwort.
5. Stellen Sie die Umlaufzeit auf 2 sec im Uhrzeigersinn ein und erstellen sie einen Plot.
6. Nehmen Sie einen Screenshot des Positionssignals auf und beschreiben Sie den Verlauf sowie markante Punkte.
7. Nehmen Sie einen Screenshot des Simulink-Blockschaltbilds auf, beschreiben Sie den Aufbau der Steuerung und geben Sie die relevanten Blockparameter an.

2.3 Regelung der oberen Gleichgewichtslage – PD-Regler

Ziel ist die Regelung der oberen Gleichgewichtslage. Zum Zeitpunkt $t = 0$ befindet sich der Hebelarm mit Gewicht bereits in der oberen Gleichgewichtslage. In dieser Aufgabe sollen nur Störungen der oberen Gleichgewichtslage ausgeregelt werden.

2.3.1 Mathematische Herleitung der Reglerparameter

In diesem Aufgabenteil werden die Reglerparameter mathematisch hergeleitet. Dokumentieren Sie den gesamten Lösungsweg.

Hinweis:

Es kann von einem idealen Pendel ausgegangen werden.

Ein Zustandsregler hat die Form $u = \mathrm{K}^T \cdot x$. Dabei ist x der Zustandsvektor der Regelstrecke, u stellt die skalare Stellgröße dar und K^T beinhaltet die frei wählbaren Reglerparameter.

1. Beschreiben Sie das Pendelverhalten mit Hilfe der einwirkenden Kräfte (Kräftediagramm zeichnen und Bewegungsgleichungen aufstellen).
2. Stellen Sie die Differentialgleichungen nach der höchsten Ordnung um.
3. Führen Sie eine Linearisierung um die obere Ruhelage durch.
4. Stellen Sie für die Regelstrecke die Laplace Übertragungsfunktion $G(s)$ auf und bestimmen Sie die Polstellen.
5. Beschreiben Sie das Systemverhalten in Bezug auf das Schwingverhalten.
6. Bestimmen Sie durch Koeffizientenvergleich die Reglerparameter eines Zustandsreglers, so dass die Eigenwerte des geschlossenen Regelkreises $\mu_1 = -5$ und $\mu_2 = -7$ sind.
7. Begründen Sie mathematisch, warum es sich um einen PD-Regler handelt und bestimmen Sie dessen Parameter (K_d, …).
8. Bestimmen Sie die Phasenreserve des geschlossenen Regelkreises.

Für alle folgenden Aufgaben gilt, dass K_d durch ein PT1-Element mit der Zeitkonstante $\tau = 10^{-2}$ gefiltert wird.

9. Überführen Sie den Regler in eine diskrete Darstellung. Nutzen Sie zur Diskretisierung das Euler-Verfahren und eine Abtastzeit von $10\,ms$.

2.3.2 Implementierung der Regelung auf dem Arduino

In diesem Aufgabenteil wird die in Aufgabe 2.3.1 berechnete **diskrete** Regelung in Simulink implementiert, auf den Arduino übertragen und ihr Verhalten untersucht.

Hinweise:

Verwenden Sie die Fläche hinter dem Arduino, um den Aufbau mit einem Gewicht (z.B. einem Buch) zu stabilisieren (siehe Abbildung 1).

Beachten Sie, dass der Motor nicht in der Lage ist, eigenständig die obere Gleichgewichtslage aus der unteren Gleichgewichtslage zu erreichen. Bringen Sie daher das Pendel von Hand in die obere Gleichgewichtslage.

Beachten Sie die unterschiedlichen Vorzeichen von Ableitung und Motorsteuersignal.

Beachten Sie, dass die obere Gleichgewichtslage am *posRad* Ausgang nicht 0 Rad entspricht.

1. Implementieren und dokumentieren Sie die in Aufgabe 2.3.1 Teil 9 berechnete Regelung in Matlab-Simulink.
2. Übertragen Sie die Regelung auf den Arduino.
3. Untersuchen Sie das Verhalten Ihrer Regelung durch Störung des Systems (händisches Auslenken des Pendels in der oberen Ruhelage) und beschreiben Sie kurz das Verhalten für die in Tabelle 1 vorgegebenen sieben Konfigurationen des Pendelgewichts.
4. Untersuchen sie den stationären Fehler des Winkels.
    a. Beschreiben Sie den stationären Fehler und mögliche Ursachen.
    b. Falls ein stationärer Fehler vorhanden ist, implementieren Sie eine Korrektur mit Simulink und dokumentieren Sie diese.

Tabelle 1: Pendelgewicht

Konfiguration	Schraube und Mutter	Kleine Unterlagscheiben	Große Unterlagscheiben
1	Nein	Nein	0
2	Ja	Nein	0
3	Ja	Ja	0
4	Ja	Ja	4
5	Ja	Ja	8
6	Ja	Ja	12
7	Ja	Ja	14

2.4 Aufschwingen des Pendels

Um das Pendel aus der unteren Gleichgewichtslage in den Einzugsbereich der PD-Regelung zu überführen, muss die potentielle Energie schrittweise durch Aufschwingen erhöht werden. Hierfür kann mit Hilfe von Simulink die Winkelgeschwindigkeit bestimmt werden. Ab einem Schwellenwert v_x wird das Drehmoment des Motors umgekehrt. Es wird zwischen zwei Drehmomenten umgeschaltet, weshalb man hier von einer Zweipunktregelung sprechen kann. Das Überschwingen wird durch Umschalten zwischen der Zweipunktregelung und der PD-Regelung unterbunden. Dafür wird basierend auf der Pendelposition zwischen den beiden Reglern umgeschaltet.

Hinweise:

Verwenden Sie die Fläche hinter dem Arduino, um den Aufbau mit einem Gewicht (z.B. einem Buch) zu stabilisieren (siehe Abbildung 1).

Beachten Sie, dass die obere Gleichgewichtslage am *posRad* Ausgang nicht 0 Rad entspricht.

Bearbeiten Sie diese Aufgabe mit Konfiguration Nr. 7 gemäß Tabelle 1.

1. Entwickeln Sie eine Simulink-Funktion, die in der Lage ist, auf Basis von Position/Positionsfehler zwischen der in Aufgabe 9 entwickelten Regelung und dem bereitgestellten bang-bang Control (Kapitel 1.2.2) umzuschalten. Die Umschaltung muss zwischen einem Aufschwingen und Balancieren des Pendels unterscheiden können. Dokumentieren Sie ihre Lösung.
2. Bestimmen Sie empirisch, ab welcher Position der PD-Regler in der Lage ist, die obere Gleichgewichtslage zu erreichen.
3. Bestimmen Sie den *minTorque*, *maxTorque* Wert empirisch und beschreiben Sie Ihr Vorgehen sowie das Ergebnis.
4. Beschreiben Sie das Verhalten des Gesamtsystems, inklusive eines Screenshots des Einschwingvorgangs der Ausgangsgrößen *posRad* und *deriv* sowie eines Screenshots des gesamten Simulink-Schaltplans.