

DIPLOMARBEIT

Modernisierung des Praktikumsversuchs
“Magnetischer Schwebekörper” –
Streckenanalyse und Reglersynthese mit
MATLAB/Simulink

Diplomand: Andreas Bernhardt

Fachbereich: Elektro- und Informationstechnik

Fachrichtung: Kommunikationstechnik

Erstprüfer: Prof. Dr.-Ing. Wolfgang Schittenhelm

Zweitprüfer: Dipl.-Ing.(FH) Peter Viehhauser

Abgabedatum: 31. März 2010

Erklärung gemäß § 31,5 RaPo

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit

**Modernisierung des Praktikumsversuchs “Magnetischer Schwebekörper” –
Streckenanalyse und Reglersynthese mit MATLAB/Simulink**

selbständig verfasst, noch nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die vorgegebenen Quellen und Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum

Andreas Bernhardt

Danksagung

Dank gebührt an erster Stelle Herrn Dipl.-Ing.(FH) Peter Viehhauser. Er hat den Gegenstand dieser Diplomarbeit vorgeschlagen und mich bei deren Durchführung stets hilfsbereit mit Rat und Tat begleitet. Auch die Aufgabe des Zweitprüfers wurde von Herrn Viehhauser liebenswürdigerweise übernommen.

Die unkoplierten Nutzungsbedingungen für das Mess- und Regelungstechniklabor der Hochschule Rosenheim verdanke ich ihm, sowie Herrn Prof. Dr.-Ing. M.Sc. Peter Zentgraf. Herr Professor Zentgraf trug ausserdem durch seine didaktisch wertvollen Ideen wesentlich zu einer verständlicheren Gestaltung der Praktikumsanleitung bei.

Herrn Prof. Dr.-Ing. Wolfgang Schittenhelm danke ich vielmals für seine spontane Zusage bei der Frage nach einem Erstprüfer. Durch seine fachliche Führung und vielfältigen Hilfestellungen fiel mir ein zielgerichtetes Bearbeiten des Themas recht leicht.

Besonderer Dank gilt meiner Frau Dipl.oec.troph. Marion Bernhardt für das zur Verfügung Stellen ihrer Bürogeräte, ihr Coaching beim Schreiben der Arbeit und die psychische Betreuung.

Auch den übrigen Familienmitgliedern, vor allem meiner Mutter, danke ich sehr für ihre Unterstützung während der Studienjahre.

Rosenheim, März 2010

Inhaltsverzeichnis

Erklärung gemäß § 31,5 RaPo	I
Danksagung	II
Inhaltsverzeichnis	III
Abkürzungsverzeichnis	VII
1. Einleitung	1
1.1. Beschreibung des ursprünglichen Versuchs	1
1.1.1. Experimentaufbau	1
1.1.2. PC-Experimentsteuerung	3
1.2. Aufgabenstellung	4
2. Beschreibung der verwendeten Software	5
2.1. MATLAB	5
2.2. Simulink	5
2.3. Toolboxen	5
2.3.1. Data Acquisition Toolbox	6
2.3.2. System Identification Toolbox	6
2.3.3. Control System Toolbox	6
2.3.4. Real-Time Workshop	7
2.3.5. Real-Time Windows Target	7
2.4. Neu erstellte MATLAB-Skriptdateien, -Funktionen und Simulink-Modelle	8
2.4.1. load_conv_param.m	8
2.4.2. get_static_data.m	8
2.4.3. plot_static_data.m	9
2.4.4. get_step_data.mdl	9
2.4.5. prep_step_data.m	9
2.4.6. plot_step_lin.m und plot_step_log.m	9
2.4.7. get_state_param.m	10
2.4.8. comp_id_re.m	10
2.4.9. loop_sim.mdl	10
2.4.10. loop_real.mdl	10
2.4.11. reset_PCI6014.m	10
3. Hardware	11
3.1. Signale am Experimentaufbau	11
3.1.1. Stellsignal	12
3.1.2. Messsignal	12
3.2. Externer Sollwertgeber	13
3.3. Datenerfassungskarte und PC	13

4. Streckenanalyse	14
4.1. Mathematisches Modell der Regelstrecke	14
4.2. Klassische Ermittlung der Streckenparameter	16
4.2.1. Statische Kennlinie	17
4.2.2. Sprungantwort	19
4.2.3. Vergleich: reale Sprungantwort mit theoretischem Verlauf	23
4.3. Ermittlung der Streckenparameter mit der System Identification Toolbox	24
4.3.1. Grey-Box-Modell zur Parameterschätzung	24
4.3.2. Vergleich: reale Sprungantwort mit theoretischem Verlauf	27
5. Reglerdesign mit der Control System Toolbox	28
5.1. Initialisierung	28
5.1.1. Erzeugung der Streckenübertragungsfunktion	28
5.1.2. Vorbereitung der Benutzeroberfläche	29
5.2. Entwurf	29
5.2.1. PD-Regler	30
5.2.2. PID-Regler	32
6. Simulation des geschlossenen Regelkreises	35
6.1. Vorbereitende Überlegungen und Berechnungen	36
6.1.1. Realitätsnahe Modellierung	36
6.1.2. Konvertierung der Reglerübertragungsfunktionen	36
6.2. Verifizierung der Ergebnisse aus dem Reglerentwurf	37
6.2.1. PD-Regler	37
6.2.2. PID-Regler	38
6.2.3. Resumee	39
6.3. Bestimmung und Erweiterung des regelbaren Bereichs	40
6.4. Echtzeit-Simulation	40
7. Echtzeituntersuchung des geschlossenen Regelkreises mit realer Strecke	42
7.1. Regelkreis mit PD-Regler	43
7.2. Regelkreis mit PID-Regler	45
7.3. Resumee	47
7.4. Sollwertvorgabe durch externe Hardware	48
8. Zusammenfassung und Ausblick	49
Literaturverzeichnis	51
A. Anhang	53
A.1. Abbildungsverzeichnis	54
A.2. Pläne und Schaltbilder	56
A.2.1. Leistungsverstärker	56
A.2.2. Messelektronik	58
A.2.3. Schnittstellen	60
A.2.4. Blockschaltplan des neuen Versuchsaufbaus	63

A.2.5. Verdrahtungsplan des Schiebepotentiometers	64
A.3. m-Files	65
A.3.1. Listing: load_conv_param.m	65
A.3.2. Listing: get_static_data.m	65
A.3.3. Listing: plot_static_data.m	69
A.3.4. Listing: prep_step_data.m	70
A.3.5. Listing: plot_step_lin.m	71
A.3.6. Listing: plot_step_log.m	72
A.3.7. Listing: get_state_param.m	73
A.3.8. Listing: comp_id_re.m	74
A.3.9. Listing: reset_PCI6014.m	77
A.4. Kennlinien	78
A.4.1. Statische Kennlinie der Regelstrecke	78
A.4.2. Aufbereitete Sprungantwort der Regelstrecke	79
A.4.3. Näherungsgerade zur Auswertung der halblogarithmisch aufge- tragenen Sprungantwort	80
A.4.4. Vergleich von realer Sprungantwort mit idealen Verläufen aus klassischer Streckenanalyse und Grey-Box-Modellierung	81
A.4.5. Ausdrücke der Ergebnisse des ursprünglichen Praktikumsversuchs	82
Praktikumsanleitung	86
Teil A:	
Grundlagen	87
1. Versuchsaufbau	87
2. Versuchsbeschreibung	88
3. Verwendete Hard- und Software	89
3.1. Datenerfassungskarte	89
3.2. Software	89
4. Mathematisches Modell der Regelstrecke	93
Teil B1:	
Aufnahme der Sprungantwort der Regelstrecke	96
1. Aufnahme der Sprungantwort	96
2. Sichtung und Aufbereitung der Daten	97
Teil B2:	
Klassische Streckenanalyse	99
1. Sprungantwort	99
1.1. Bestimmung von p	100
2. Statische Kennlinie	100
2.1. Aufnahme der statischen Kennlinie	100
2.2. Bestimmung der Steigung und des Werts von k	101
3. Vergleich der realen Sprungantwort mit dem theoretischem Verlauf . . .	101

Teil B3:**Parameterschätzung mittels****Grey-Box-Modellierung****102**

1. Sprungantwort 103
2. Erzeugung eines `iddata`-Objekts der Sprungantwort 103
3. Erzeugung eines `idgrey`-Modells der Regelstrecke 104
4. Schätzung der Streckenparameter 104
5. Vergleich der realen Sprungantwort mit dem theoretischen Verlauf . . . 104

Teil C:**Reglerentwurf mit der****Control System Toolbox****105**

1. Erzeugung der Streckenübertragungsfunktion G_S 105
2. Initialisierung von `sisotool` 107
3. Reglerdesign 107
 - 3.1. PD-Regler 108
 - 3.2. PID-Regler 109

Teil D:**Simulation des geschlossenen Regelkreises****110**

1. Fertigstellung des Modells 110
2. PD-Regler 111
 - 2.1. Antwort des Regelkreises auf den Einheitssprung 111
 - 2.2. Optimierung der Reglerparameter 111
3. PID-Regler 112
4. Simulation in Echtzeit 112

Teil E:**Echtzeitexperiment****114**

1. Setup 114
2. Echtzeitregelung der realen Strecke 114
 - 2.1. PD-Regler 114
 - 2.2. PID-Regler 115
3. Einbindung des externen Sollwertgebers 115

Abkürzungsverzeichnis

Abkürzung Definition

ADC	Analog-to-Digital-Converter
A/D	Analog to Digital
AI	Analog Input
AO	Analog Output
API	Application Program Interface (Programmierschnittstelle)
CPU	Central Processing Unit
DAC	Digital-to-Analog-Converter
D/A	Digital to Analog
FET	Feldeffekttransistor
GND	Ground (elektrisches Massepotential)
GUI	Graphical User Interface
I/O	Input Output
MIMO	Multiple Input Multiple Output
MS-DOS	Microsoft Disk Operating System
NRSE	Nonreferenced Single-Ended (vgl. [19])
PCI	Peripheral Component Interconnect (PCI-Bus)
RAM	Random Access Memory (Arbeitsspeicher)
SISO	Single Input Single Output

Formelzeichen Dimension Größe

a	$\frac{m}{s^2}$	Beschleunigung
C		Reglerübertragungsfunktion (Compensator)
f	$Hz (= \frac{1}{s})$	Frequenz
G		Übertragungsfunktion allgemein
i	A	Strom
k	$\frac{mm^3}{mA \cdot s^2}$	Streckenparameter
K		Verstärkung
m	kg	Masse
ω	$\frac{1}{s}$	Kreisfrequenz ($\omega = 2 \cdot \pi \cdot f$)
p	$\frac{mm^2}{s^2}$	Streckenparameter
s	$\frac{1}{s}$	Parameter der Laplace-Transformierten ($s = \sigma + j \cdot \omega$)
t	s	Zeit
T	s	Periodendauer ($T = \frac{1}{f}$)
T	s	Zeitkonstante ($T = \frac{1}{\omega} = \frac{1}{2 \cdot \pi \cdot f}$)
u	V	Spannung

Index Definition

hilfs	Hilfsstromquelle bzw. Hilfsspule
i, m, n	allgemeine Zählindizes
leist	Leistungsverstärker
N	Nullstelle
P	Pol
PD	für PD-Regler (-Elemente)
PID	für PID-Regler (-Elemente)
R	Regler
S	Strecke
steuer	Steuergröße
W	kennzeichnet die Übertragungsfunktion des geschlossenen Regelkreises
x	Messgröße der Körperposition in vertikaler Richtung (Höhe)

1. Einleitung

Das Labor für Mess- und Regelungstechnik der Hochschule Rosenheim bietet unter anderem diverse Praktikumsversuche an. Hier haben Studenten der Fakultät Ingenieurwissenschaften die Möglichkeit, ihr theoretisches Wissen auszubauen und diese Kenntnisse auch praktisch anzuwenden.

Softwaregesteuerte Regelungen gehören zum derzeitigen Stand der Technik. Gängige Praxis ist ebenfalls die computergestützte Analyse von Systemen, sowie Simulation während der Entwicklung. Im Laufe des Studiums ist es also sehr wichtig, auch mit solchen wissenschaftlichen Lösungswegen vertraut zu werden.

Der Praktikumsversuch "Magnetischer Schwebekörper" verfolgt zwar diese Ansätze, allerdings hat dessen Software und ein Teil der Hardware erheblich an Aktualität eingebüßt. Ziel dieser Arbeit ist es deshalb, den Versuchsablauf mittels Verwendung von MATLAB/Simulink an die heutigen technischen Stand anzupassen.

1.1. Beschreibung des ursprünglichen Versuchs

Anfang der 1990er Jahre entwickelte der Lehrstuhl für Steuerungs- und Regelungstechnik der Technischen Universität München das Lehr- und Experimentiermodell "Magnetischer Schwebekörper". Es existieren mehrere dieser Aufbauten. Einer davon wurde der Hochschule Rosenheim mit zugehöriger Software zur Verfügung gestellt und ist dort seit etwas mehr als 10 Jahren im Einsatz.

Das Modell besteht aus zwei Einheiten, dem eigentlichen Experimentaufbau und der PC-Experimentsteuerung [6].

1.1.1. Experimentaufbau

Im steuerungs- und regelungstechnischen Sinn handelt es sich bei der Regelstrecke um ein nichtlineares, zeitinvariantes und instabiles System mit einer Ein- und einer Ausgangsgröße¹.

¹SISO: Single Input Single Output

Abbildung 1.1 zeigt die Frontansicht des Aufbaus.

Der höhenverstellbare Scherentisch ((5)) dient als Ablage für den ferromagnetischen Körper ((2)). Die Spulenanordnung ((1)) stellt den Arbeitspunkt durch eine Grundmagnetisierung, die eine mittels Konstantstromquelle gespeiste Hilfsspule erzeugt. Mit der vom Leistungsverstärker versorgten Steuerspule wird ein magnetisches Feld additiv überlagert. Um der Erwärmung der Magnetspulen entgegenzuwirken, sorgt das Lüftersystem ((6)) für ausreichende Kühlung. Die Ermittlung der Position des Körpers übernimmt das elektro-optische Messsystem (Lichtquelle: (4), Sensor: (3)). An den analogen Kontrollanzeigen ((7) und (8)) sind die aktuellen Werte von Messsignal u_x und Steuerstrom i_{steuer} ablesbar. Die Kontroll-LEDs ((9) und (10)) signalisieren eingestellten Arbeitspunkt (Grundmagnetisierung) und den Netzbetrieb der Anordnung.

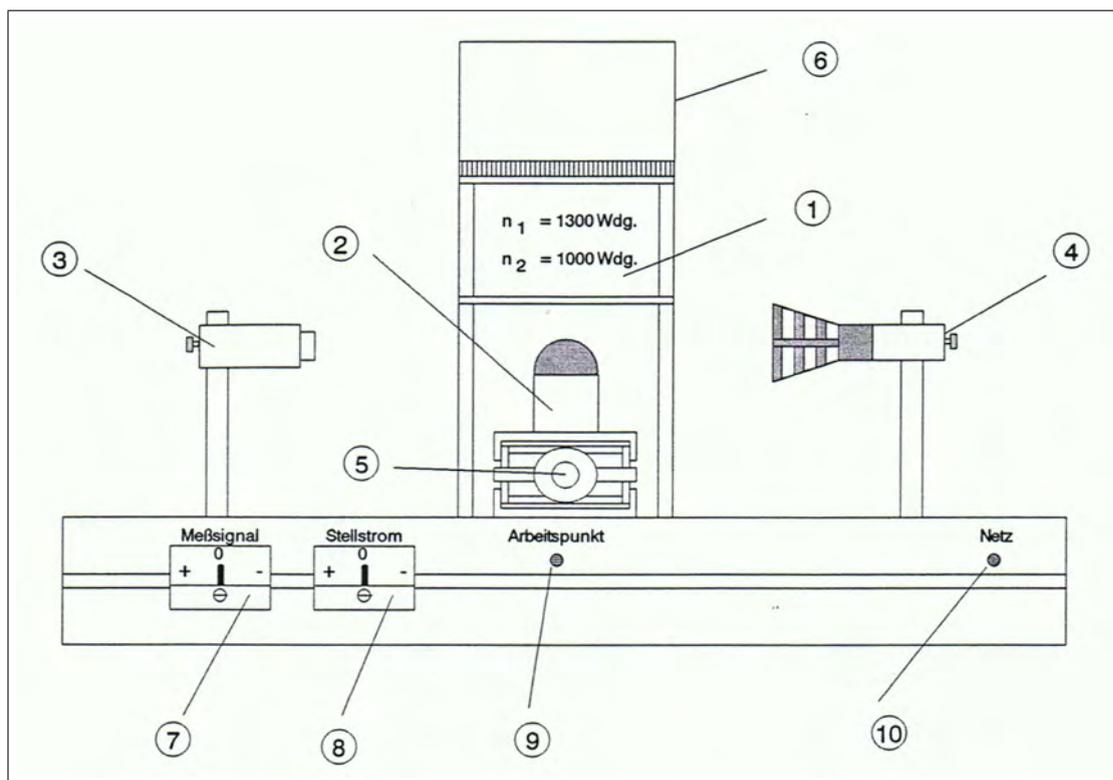


Abbildung 1.1: Experimentaufbau, Frontansicht (Quelle: [6])

An den D/A²- bzw. A/D³-Wandlern der Datenerfassungskarte Keithley Metrabyte DAS-16/1600 werden die Schnittstellensignale des Experimentaufbaus mit dem PC ausgetauscht. 12-Bit-Werte repräsentieren dabei PC-intern das eingelesene Messsignal u_x zur Positionsbestimmung und die ausgegebene Steuerspannung u_{steuer} für die Einstellung des Spulenstroms i_{steuer} , jeweils im Bereich $-10 \dots +10 \text{ V}$ [6].

Genauere Ausführungen zu den Signalen und deren Umsetzung finden sich in Abschnitt 3.1 ab Seite 11.

²Digital to Analog

³Analog to Digital

1.1.2. PC-Experimentsteuerung

Die in Turbo Pascal und C geschriebene Software läuft auf einem PC mit 133MHz Prozessor unter MS DOS 3.1, einem der ersten Betriebssystem von Microsoft für x86-PCs. Ein MATLAB-Interpreter dient der graphischen Darstellung.

In Abbildung 1.2 ist das Hauptmenü des Programms dargestellt. Wie man sieht, gliedert sich die Baumstruktur in zwei Hauptgruppen.

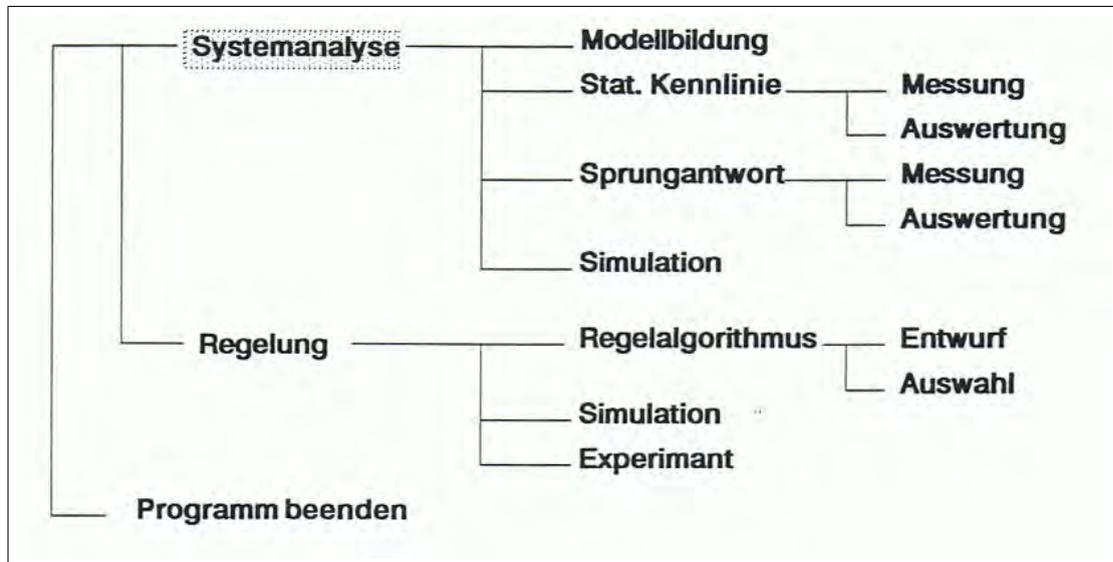


Abbildung 1.2: PC-Experimentsteuerung, Menü (Quelle: [6])

Während der **Systemanalyse** werden die Parameter der Regelstrecke anhand einer selbst aufzunehmenden statischen Kennlinie und Sprungantwort bestimmt. Die Vermittlung theoretischer Grundlagen über Bildschirmausgaben ergänzt die Menüführung. Verwendet wird eine Differentialgleichung zweiter Ordnung, welche die Strecke in linearisierter Form mathematisch modelliert.

Die **Regelung** bietet vorimplementierte Algorithmen der Reglertypen P, PD und PID. So wird mit den vorher ermittelten Streckenparametern der geschlossene Regelkreis theoretisch, durch Simulation, sowie im echtzeitgeregelten Experiment untersucht.

Ein quasi-paralleler Programmablauf ermöglicht auch während aktiver Regelung beispielsweise die Manipulation der Regelparameter oder die Anzeige von Diagrammen auf dem Bildschirm.

In beiden Menügruppen werden theoretische Ergebnisse den experimentellen gegenübergestellt. Diagramme können, etwa zur weiteren Analyse oder graphischen Auswertung, über einen am PC angeschlossenen Drucker ausgegeben werden [6].

1.2. Aufgabenstellung

Der Experimentaufbau ist robust und im Prinzip verschleissfrei. Im Hinblick auf Energieeffizienz wäre zwar eine Optimierung des Leistungsverstärkerteils und des Kühlsystems der Spulen wünschenswert, darauf soll jedoch im Rahmen dieser Diplomarbeit verzichtet werden.

Für die Regelung des Aufbaus mit aktueller Software auf einem zeitgerechten PC ist hardwaretechnisch noch eine passende Datenerfassungskarte notwendig. Hierfür wird die im Labor bereits vorhandene Multifunktions-Datenerfassungskarte NI PCI-6014 von National Instruments verwendet. Diese ist unter anderem durch ihre spezifizierten Ein- und Ausgangsdatenraten weit über denen der ursprünglichen Regelung (1KHz!) mehr als geeignet. Es verbleibt noch die Verkabelung der Schnittstellen des Experimentaufbaus mit den jeweiligen analogen Ein- und Ausgängen der NI PCI-6014.

Für den Reglerentwurf soll nun die verbreitete Softwareplattform MATLAB/Simulink eingesetzt werden. Ziel dabei ist zum einen, die Studierenden in einem didaktisch angepassten Rahmen in gängigen modernen Analyse- und Synthese-Methoden der Steuerungs- und Regelungstechnik zu schulen. Zum anderen soll der Umgang mit MATLAB/Simulink prinzipiell vertrauter werden. Zur Unterstützung der Versuchsdurchführung wurden in der Diplomarbeit auch MATLAB-Skript-Dateien und -Funktionen, sowie Simulink-Modelle vorbereitet.

Der Hauptteil der Arbeit beschäftigt sich auch mit der Erstellung einer Praktikumsanleitung, die durch einen modularen Aufbau den jeweiligen Lernzielen der Studiengänge Elektro- und Informationstechnik als auch Produktionstechnik entsprechend zusammengestellt werden kann.

Am Beispiel der Strecke "Magnetischer Schwebekörper" wird sukzessiv der Weg von der Systemanalyse und -Simulation zur effizienten Reglersynthese, Simulation des geschlossenen Regelkreises und letztendlich dem Echtzeitexperiment mittels vorher untersuchter Regelparameter softwaregestützt mit den Studenten gegangen.

2. Beschreibung der verwendeten Software

2.1. MATLAB

MATLAB¹ ist eine mächtige Entwicklungsumgebung für die numerische Berechnung und graphische Darstellung auch sehr komplexer mathematischer Zusammenhänge. Weltweit schätzen viele Wissenschaftler und Ingenieure diese Software. Durch diverse Erweiterungspakete, so genannte Toolboxen, kann der Funktionsumfang auf spezifische Aufgabenstellungen, beispielsweise solche der digitalen Signalverarbeitung oder der Mess- und Regelungstechnik, angepasst werden.

Für diese Diplomarbeit wurde die MATLAB-Version R2008a mit den unter Abschnitt 2.3 aufgeführten Toolboxen verwendet.

2.2. Simulink

Simulink bietet dem Benutzer ein graphisches Interface, über das dynamische Systeme mittels Drag-And-Drop-Verfahren schnell modellier- und simulierbar sind. Mit entsprechenden Toolboxen (vgl. Abschnitt 2.3.4) und geeigneter zusätzlicher Hardware (beispielsweise der hier verwendeten, in Abschnitt 3.3 vorgestellten) ist ausserdem das Einlesen und Ausgeben von Signalen und damit eine Verwirklichung von Steuerungen und Regelungen realer Systeme, auch in Echtzeit, möglich.

Als Schnittstelle zu MATLAB dient der Workspace, über den Daten zwischen beiden Modulen ausgetauscht werden.

Regelungstechnische Aufgaben sind sowohl mit MATLAB als auch mit Simulink lösbar. Im MATLAB-Code können Daten sehr gezielt manipuliert werden, die große Stärke von Simulink-Modellen ist die graphisch bedingte Übersichtlichkeit der Modelle [2].

2.3. Toolboxen

Im Folgenden wird kurz auf die unter MATLAB/Simulink verwendeten Erweiterungspakete und deren Funktionalität eingegangen.

¹MATrix LABoratory

Detaillierte Informationen zu allen Toolboxen und deren Funktionen finden sich in der MATLAB-Hilfe unter *Help*→*Product Help*.

2.3.1. Data Acquisition Toolbox

Die Data Acquisition Toolbox enthält eine Vielzahl von Werkzeugen, die das analoge und digitale Einlesen und Ausgeben von Signalen über PC-kompatible Geräte wesentlich erleichtern.

In MATLAB/Simulink erzeugte Daten können dabei als Signale auf einfachem Wege über entsprechende Hardware ausgegeben werden. Daten eingelesener Signale stehen zur Weiterverarbeitung in MATLAB/Simulink zur Verfügung. Gerade Signal-I/O² mit National Instruments Hardware wird durch eine effiziente API³ benutzerfreundlich unterstützt [14].

Für diese Diplomarbeit wurde die Version 2.12 der Data Acquisition Toolbox verwendet.

2.3.2. System Identification Toolbox

Die System Identification Toolbox erstellt unter MATLAB/Simulink lineare und nicht-lineare mathematische Modelle von dynamischen Systemen auf Basis gemessener Ein- und Ausgangsdaten.

Es können einerseits Koeffizienten von gewöhnlichen Differentialgleichungen und Differenzgleichungen, welche mittels physikalischer Grundgesetze modelliert wurden, berechnet werden. Der datengetriebene Ansatz macht aber auch Systeme beschreibbar, welche sich nicht auf diese Art repräsentieren lassen, wie beispielsweise thermische Prozesse eines Fluidums und elektromechanische Systeme. Das resultierende Modell kann unter anderem bei der Simulation eines Systemausgangs bei einem bestimmten Eingangssignal Verwendung finden, oder beim Design einer Regelung helfen [5, 17].

Für diese Diplomarbeit wurde die Version 7 der System Identification Toolbox verwendet.

2.3.3. Control System Toolbox

Die Control System Toolbox bietet Werkzeuge für die systematische Analyse, den Entwurf und letztlich die Optimierung steuerungs- und regelungstechnischer Systeme.

Zunächst wird ein System in Form eines linearen Modells als Übertragungsfunktion, Pol-Nullstellen-Diagramm oder in der Zustandsraumdarstellung definiert. Eine Konvertierung zwischen den verschiedenen Repräsentationen ist problemlos möglich. Nach

²Input Output

³Application Program Interface

der Modellierung kann das Antwortverhalten sowohl im Zeit- als auch im Frequenzbereich untersucht und manipuliert werden. Auf Workflow basierende graphische Benutzeroberflächen führen den Anwender durch die einzelnen Schritte der Analyse- und Entwurfsprozesse. Die automatisierten, interaktiven Techniken erleichtern typische Entwicklungsaufgaben der Steuerungs- und Regelungstechnik, wie zum Beispiel das Optimieren von Reglerparametern [9, 13].

Für diese Diplomarbeit wurde die Version 8 der Control System Toolbox verwendet.

2.3.4. Real-Time Workshop

Mit dem Real-Time Workshop lässt sich aus MATLAB-Code oder Simulink-Modellen automatisch eigenständig lauffähiger C-Quellcode für Echtzeit- und Nicht-Echtzeit-Anwendungen generieren, packen und kompilieren.

Nachdem der generierte Code auf geeignete Hardware-Zielplattformen übertragen wurde und dort ausgeführt wird, ist dessen Beeinflussung aus der MATLAB/Simulink-Umgebung immer noch möglich. So eröffnet sich ein schneller und direkter Weg vom System-Design zur Implementierung, den man etwa für Rapid Prototyping oder Hardware-in-the-Loop-Tests nutzen kann [10, 15].

Für diese Diplomarbeit wurde die Version 7 des Real-Time Workshops verwendet.

2.3.5. Real-Time Windows Target

Dieses Paket wird benötigt, da die verwendete Datenerfassungskarte über keinen eigenen Prozessor verfügt, auf dem der mittels Simulink und Real-Time Workshop erzeugte Code eigenständig laufen könnte.

Durch Real-Time Windows Target kann ein und derselbe PC gleichzeitig als Host und als Target genutzt werden. Dabei wird ein Teil der CPU⁴-Zeit des PCs als Echtzeit-Kernel mit maximaler Prozesspriorität genutzt, wobei der Systemtakt als primäre Zeitreferenz dient. Die mitgelieferten I/O-Gerätetreiber erleichtern die Einbindung von Aktoren und Sensoren in eine Echtzeitregelung. Die Verbindung des Echtzeit-Kernels mit dem zugrunde liegenden Simulink-Modell ermöglicht ausserdem eine sich direkt auf den laufenden Code auswirkende Beeinflussung der Modellparameter. Da die PC-Aktivitäten dabei auf ein Minimum reduziert werden, sind Abtastraten bis zu 5kHz erreichbar.

Um Real-Time Windows Target nutzen zu können, muss zuerst der Echtzeit-Kernel durch das MATLAB-Skript `rtwintgt.m` installiert werden. Des Weiteren ist für jedes Simulink-Modell, in dem dieses Target benutzt wird, die Standardkonfiguration der Simulationsparameter mit dem Befehl `rtwinconfigset('modelname')` zu laden. Zuvor sollte das Modell unter diesem Modellnamen abgespeichert worden sein [16].

Für diese Diplomarbeit wurde die Version 3.1 des Real-Time Windows Target verwendet.

⁴Central Processing Unit

2.4. Neu erstellte MATLAB-Skriptdateien, -Funktionen und Simulink-Modelle

Zur Unterstützung der Durchführung des Praktikumsversuchs wurden die im diesem Abschnitt in der Reihenfolge ihrer Verwendung aufgeführten m-Files (*.m) und Simulink-Modelle (*.mdl) erstellt. Sie sind als Grundmodelle für die verschiedenen, während des Praktikumsablaufs zu lösenden Aufgaben anzusehen. Die Programme finden sich auf dem beiliegendem Datenträger, die Listings der ausführlich mit Kommentaren versehenen m-Files zusätzlich im Anhang A.3 ab Seite 65.

Auf dem Datenträger gibt es zusätzliche Versionen dieser Programme. Von den Studenten zu bearbeitende Teile sind am Dateinamensende `_todo` zu erkennen. Für jeden Praktikumsabschnitt gibt es Musterlösungen, die analog zur Beschreibung in der Praktikumsanleitung (siehe Anhang ab Seite 1) benannt sind.

Alle MATLAB-Skripte und -Funktionen sind so geschrieben, dass nach der Eingabe von `help programmname` in der MATLAB-Kommandozeile ein Hilfetext mit Informationen über den Zweck des Programms, Funktions- bzw. Skriptaufruf und die verwendeten Ein- und Ausgabevariablen erscheint. Dabei wurde darauf geachtet, dass die Form der Hilfetexte dem MATLAB-Standard entspricht. Die erstellten Simulink-Modelle beinhalten kommentierende Textfelder, um beispielsweise Signalfluss und -anpassung zu verdeutlichen. An allen signifikanten Stellen im Modell sind zusätzlich Visualisierungsmöglichkeiten für die jeweiligen Signale vorgesehen. Logisch zusammen gehörende Blöcke sind farblich gekennzeichnet. Analoge Ein- und Ausgänge wurden zum besseren Verständnis in einen mit dem Bild des Experimentaufbaus versehenen Subsystemblock ausgelagert. Die Rate für ausgegebene und eingelesene Signale beträgt in den betreffenden Modellen und m-Files jeweils 2 kHz .

Die Anwendung aller Programme wird anhand eines beispielhaften Versuchsablaufs in den Abschnitten 4.2 und 4.3 ab Seite 16, sowie in den Kapiteln 6 und 7 ab Seite 35 erläutert.

2.4.1. `load_conv_param.m`

Das MATLAB-Skript lädt die Konvertierungsparameter für die Umrechnung der Spannungswerte an den analogen Ein- und Ausgängen der Datenerfassungskarte in die Positions- bzw. Spulenstromwerte des Systemkerns der Regelstrecke in den Workspace.

2.4.2. `get_static_data.m`

Diese Funktion dient der dialoggeführten Aufnahme von maximal 20 Messpunkten für die Ermittlung der statischen Kennlinie der Regelstrecke "Magnetischer Schwebekörper".

Das Programm initialisiert mit Unterstützung von Funktionen der Data Acquisition Toolbox die Kommunikation mit der Datenerfassungskarte und führt den Benutzer dann über Ausgaben und Abfragen in der MATLAB-Kommandozeile sukzessiv durch die Messpunktaufnahme. Alle möglichen Fehleingaben des Benutzers, wie beispielsweise Wertebereichsüberschreitungen, sind dabei u.a. durch die Ausgabe von Fehlermeldungen abgefangen.

2.4.3. `plot_static_data.m`

Die Funktion `plot_static_data` stellt die mit `get_static_data` aufgenommenen Daten der statischen Kennlinie der Regelstrecke für die Parameterermittlung graphisch dar. Sie entspricht einer Erweiterung der MATLAB-eigenen `plot`-Funktion um aussagekräftige Titel und Achsenbeschriftungen.

2.4.4. `get_step_data.mdl`

Dieses Simulink-Modell ermöglicht die Aufnahme der Sprungantwort der Regelstrecke "Magnetischer Schwebekörper".

Die vorher im MATLAB-Workspace erstellte Variable `stepsize` legt die Höhe des Stromsprungs in *mA* fest. Nach der Ausführung des Modells liegen hier auch bei entsprechenden Einstellungen im Modell die aufgenommenen Daten zur weiteren Verarbeitung bereit. Blöcke aus der Data Acquisition Toolbox werden für die Kommunikation mit der Datenerfassungskarte genutzt.

2.4.5. `prep_step_data.m`

`get_step_data` zeichnet zwangsweise auch das Anschlagen des Körpers am unteren Ende des Spulenkerns auf. `prep_step_data` entfernt diese, bei den folgenden Analysen störenden Werte aus den aufgenommenen Daten und strukturiert diese für eine einfachere Handhabung neu. Die Minimalwerte der Ein- und Ausgangsgrößen werden außerdem auf den Wert 0 gezogen.

2.4.6. `plot_step_lin.m` und `plot_step_log.m`

Mit diesen Funktionen werden die mit `prep_step_data` aufbereiteten Daten der Sprungantwort linear bzw. halblogarithmisch graphisch aufgetragen.

Beide Funktionen entsprechen wieder, wie `plot_static_data`, Erweiterungen der Standard-`plot`-Funktion. `plot_step_log` schiebt zusätzlich die Messwerte soweit in den positiven Bereich, dass negative Logarithmuswerte vermieden werden und so die graphische Parameterermittlung erleichtert wird.

2.4.7. `get_state_param.m`

Diese Funktion erzeugt die Matrizen für die Zustandsraumdarstellung des mathematischen Modells der Regelstrecke. Sie wird für die MATLAB-Funktion `idgrey` zur Grey-Box-Modellierung mit anschließender Parameterschätzung benötigt.

2.4.8. `comp_id_re.m`

`comp_id_re` trägt die experimentell ermittelte Sprungantwort und deren theoretischen Idealverlauf aus dem mathematischen Modell der Strecke so nebeneinander graphisch auf, dass diese verglichen werden können. Es können ein oder zwei theoretische Verläufe geplottet werden, da die Funktion polymorph implementiert ist.

2.4.9. `loop_sim.mdl`

Das Simulink-Modell ermöglicht die Simulation des geschlossenen Regelkreises nach dem Reglerentwurf. Im Praktikum wird die Regelung mit PD- und PID-Regler auch in Echtzeit simuliert.

2.4.10. `loop_real.mdl`

`loop_real` nutzt die Funktionalitäten von Real-Time Workshop und Real-Time Windows Target, um die Strecke "Magnetischer Schwebekörper" in Echtzeit zu regeln. Die Signal-I/O wird auch hier mittels Data Acquisition Toolbox bereitgestellter Blöcke realisiert. Durch erweiternde Modifikationen dieses Modells ist auch die Vorgabe des Sollwertes durch den in Abschnitt 7.4 ab Seite 48 beschriebenen externen Aktor möglich.

2.4.11. `reset_PCI6014.m`

Mit `reset_PCI6014` kann die Datenerfassungskarte im Fehlerfall zurückgesetzt werden.

3. Hardware

Die neue Softwaresteuerung bedarf einiger Anpassungen im Hardware-Bereich. Auf diese wird im folgenden näher eingegangen.

Detaillierte Pläne des Leistungsverstärkerteils, der Messelektronik, des externen Sollwertgebers sowie des Pinouts der Datenerfassungskarte und der neuen Verkabelung befinden sich im Anhang A.2 ab Seite 56.

3.1. Signale am Experimentaufbau

Abbildung 3.1 zeigt eine Übersicht der grundsätzlichen Funktionsweise des Experimentaufbaus, sowie der Signale, die für die Regelung der Strecke von Bedeutung sind.

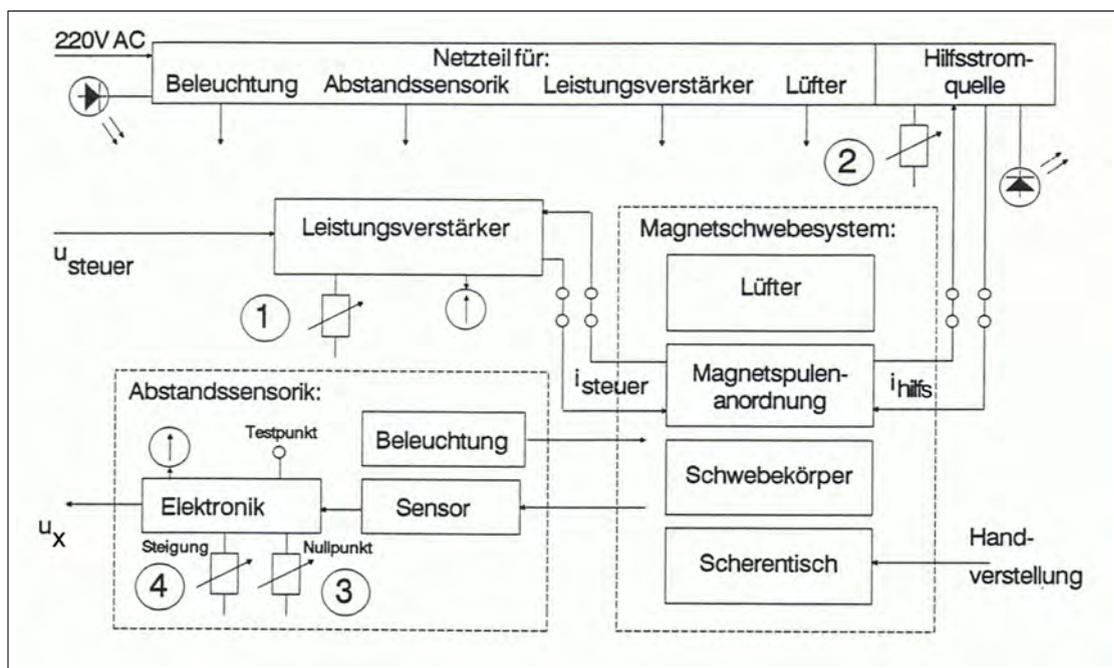


Abbildung 3.1: Experimentaufbau, Funktionsplan (Quelle: [6])

Die Potentiometer (1) bis (4) sind auf der Rückseite des Experimentaufbaus für die Justierung zugänglich (vgl. Abbildung A.5 auf Seite 60 im Anhang).

3.1.1. Stellsignal

Wie bereits im Abschnitt 1.1.1 auf Seite 2 erwähnt, wird der Strom der Steuerspule i_{steuer} durch die Spannung u_{steuer} am Eingang des Leistungsverstärkers eingestellt. Diese Spannung liefert einer der beiden D/A-Wandler-Ausgänge der in Abschnitt 3.3 vorgestellten PCI¹-Datenerfassungskarte. Das Signal wird über einen 68-poligen Stecker (Typ D) an der Karte abgegriffen. Eine zweiadrige Leitung führt es zum 25-poligen Sub-D-Stecker auf den entsprechenden Anschluss des Experimentaufbaus. Der als stromgegekoppelter Operationsverstärker mit einer FET-Leistungsstufe konzipierte Verstärker erzeugt in Abhängigkeit von diesem Stellsignal den Steuerspulenstrom. Über einen Eingangsspannungsteiler ((1) in Abbildung 3.1) kann die Verstärkung der Spannungs-Strom-Wandlung eingestellt werden [6]. Das Stellsignal ist zusätzlich noch direkt am 68-poligen Stecker der Datenerfassungskarte auf einen A/D-Wandler-Eingang für die Erfassung zur weiteren Datenverarbeitung mit einer Drahtbrücke verbunden.

Messungen beim Erstellen des Programms für die statische Kennlinie (vgl. Abschnitt 2 auf Seite 100) haben einen annähernd linearen Zusammenhang zwischen dem Stellsignal und den jeweiligen Positionen des Schwebekörpers, bei denen ein Gleichgewichtszustand zwischen der Gewichtskraft des Körpers und der auf ihn wirkenden magnetischen Kraft herrscht, gezeigt. In dem der Strecke zugrundeliegenden mathematischen Modell wird das Stellsignal deshalb durch Linearisierung genähert. Diese Näherung ist für die Streckenanalyse sowie den folgenden Reglerentwurf ausreichend genau.

3.1.2. Messsignal

Abbildung 1.1 auf Seite 2 zeigt unter anderem das elektro-optische Messsystem. Das von einer handelsüblichen Niedervolt-Halogen-Reflektorlampe (12V, 20W, 36°Abstrahlwinkel) erzeugte Licht schirmt der Schwebekörper positionsabhängig unterschiedlich stark ab. Als Sensor für dieses optische Signal wird die Photodiode BPY47 verwendet. Deren Photostrom wandelt die in Abbildung A.3 auf Seite 58 dargestellte Messelektronik in eine Spannung u_x um, die ebenfalls über eine Sub-D-Steckverbindung am Gehäuse des Experimentaufbaus verfügbar ist. Dieses Messsignal wird durch eine geeignete zweiadrige Leitung und den 68-poligen Stecker zu einem A/D-Wandler-Eingang der unter 3.3 beschriebenen Datenerfassungskarte im PC geführt. Potentiometer ermöglichen das Justieren des Messsignals ((3) in obiger Abbildung 3.1) auf 0 V am Arbeitspunkt, d.h. bei einer Höhe des Schwebekörpers von 0 mm auf der am Experimentaufbau angebrachten Höhenskala, sowie die Einstellung der Steigung des Messspannungssignals ((4) in Abbildung 3.1) [6].

¹Peripheral Component Interconnect (PCI-Bus)

Der Zusammenhang zwischen der Höhe des Körpers und dem durch den Photostrom und die nachgeschaltete Messelektronik erzeugten Positionssignal u_x ist näherungsweise linear und wird im mathematischen Streckenmodell ebenfalls durch Linearisierung genähert, was auch hier wieder für Streckenanalyse und Reglerentwurf ausreichende Genauigkeit bringt.

3.2. Externer Sollwertgeber

Eine weitere Schnittstelle wurde im Rahmen dieser Diplomarbeit für die Option der Vorgabe eines Sollwertes am geschlossenen Regelkreis durch externe Hardware geschaffen. Hierzu sind ein analoger Ausgangskanal, ein analoger Eingangskanal und die Massereferenz der Datenerfassungskarte über eine passende Leitung auf einen 4-poligen Gerätestecker (vgl. Abbildung A.7) geführt.

Für die Durchführung des Praktikumsversuchs findet dabei das in Abschnitt 7.4 ab Seite 48 beschriebene Schiebepotentiometer Verwendung. Für den Anschluss denkbar sind jedoch beliebige Aktoren, wie beispielsweise ein vierstufiger Drehschalter zum Umschalten verschiedener Sprünge oder dergleichen.

3.3. Datenerfassungskarte und PC

Wie schon erwähnt, ist die Multifunktions-Datenerfassungskarte NI PCI-6014 im Labor bereits vorhanden und durch die angegebenen Spezifikationen für die Regelung bestens geeignet (vgl. [18]). Einer der beiden analogen Ausgangskanäle (*AO 0*, maximale Ausgaberate 10 kHz) erzeugt die Steuerspannung u_{steuer} mit einer Rate von 2 kHz und 16 Bit Auflösung. Der zweite Ausgangskanal (*AO 1*) versorgt den Aktor mit einer Betriebsspannung von 10 V . Über zwei der 16 analogen Eintaakteingänge (Summenabtastrate 200 kHz) der Karte, *AI 0* und *AI 1*, werden die Signale u_x und u_{steuer} mit ebenfalls 2 kHz abgetastet und 16-Bit-aufgelöst zur Verarbeitung in den PC eingelesen. Ein dritter analoger Eingang (*AI 2*) liest mit selber Abtastrate und Auflösung das Aktorenausgangssignal ein.

Als Massepotentiale dienen grundsätzlich *AO GND* für Ausgangssignale und *AI GND* für Eingangssignale. Diese beiden Massereferenzen sind jedoch auf der Karte intern verbunden und somit identisch. Aus diesem Grund wurde an die Schnittstelle des externen Sollwertgebers lediglich *AO GND* gelegt.

Zusätzlich müssen im NRSE-Modus *AI GND* und *AI SENSE* verbunden sein (vgl. [19]), was direkt am 68-poligen Stecker an der NI PCI-6014 durch eine Brücke realisiert ist.

Ein aktueller Standard-Labor-PC mit Pentium 4 Prozessor, 2 GByte RAM, Betriebssystem MS Windows XP (32 Bit) mit SP2 und der in Abschnitt 2 ab Seite 5 beschriebenen, für den Praktikumsversuch nötigen Software nimmt die Datenerfassungskarte in seinen PCI-Slot auf.

4. Streckenanalyse

Um die nichtlineare, instabile Regelstrecke mathematisch handhabbar zu machen, sind zuerst die im nächsten Abschnitt beschriebenen physikalischen Grundüberlegungen und Vereinfachungen nötig. So entsteht eine für die Regelung ausreichend genaue Bewegungsgleichung des magnetischen Körpers in differentieller Form mit zwei unbekanntem, zu ermittelnden Parametern. Nach der Laplace-Transformation dieser einfachen linearen Differentialgleichung liegt die Streckenübertragungsfunktion vor.

Die Ermittlung der Streckenparameter erfolgt auf den in den Abschnitten 4.2 und 4.3 angegebenen Wegen. Nachgestellt ist jeweils ein Vergleich der experimentell gemessenen Sprungantwort mit deren theoretischem, aus dem mathematischen Modell errechnetem Verlauf.

4.1. Mathematisches Modell der Regelstrecke

Ausgangspunkt bei den Überlegungen sind die in Abbildung A.3 dargestellten, am magnetischen Körper wirkenden Kräfte.

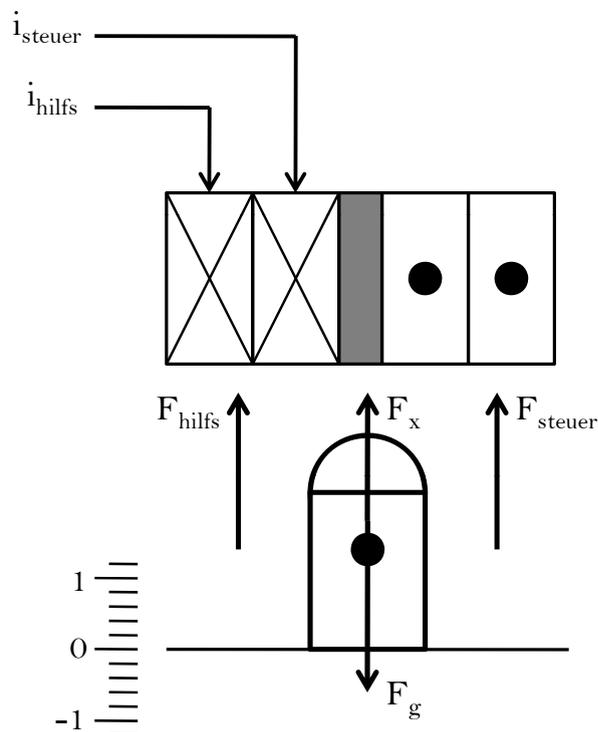


Abbildung 4.1: Kräfte am magnetischen Körper (Quelle: Eigene Darstellung nach [6])

Ist die Position x in vertikaler Richtung, wie in Abbildung A.3 angedeutet, nach oben positiv definiert, ergibt sich folgende Kräftebilanz am Körper:

$$F_x = F_{\text{steuer}} + F_{\text{hilfs}} - F_g \quad (4.1)$$

Diese wird mit $m \cdot \ddot{x} = F_x$ und $F_{\text{mag}} = F_{\text{steuer}} + F_{\text{hilfs}}$ zu

$$m \cdot \ddot{x} = F_{\text{mag}} - F_g = f(i_{\text{steuer}}, x) - F_g. \quad (4.2)$$

Die resultierende Kraft auf den Körper in x-Richtung ergibt sich aus der Summe der durch die Funktion f beschriebene Magnetkraft F_{mag} und der entgegengesetzt dazu wirkenden Gewichtskraft F_g . f ist nichtlinear und liegt nur als experimentell gewinnbares Kennlinienfeld vor.

Die weitere Behandlung wird wesentlich vereinfacht, in dem man sich auf die Betrachtung des Systems in der Umgebung des Betriebspunktes bei $x = x_0 = 0$ beschränkt. An diesem Punkt wirkt nur die durch die Hilfsspule bereit gestellte Grundmagnetisierung ($i_{\text{steuer}} = i_{\text{steuer } 0} = 0$). Magnetische Kraft und Gewichtskraft heben sich gerade auf, wodurch letztere aus Gleichung A.2 entfällt, die an diesem Punkt zu

$$m \cdot \ddot{x} = F_{\text{mag}} \quad (4.3)$$

wird. Bei einer linearisierenden Näherung ist somit nur noch F_{mag} zu behandeln:

$$\Delta F_{\text{mag}} = \left. \frac{\delta f}{\delta i_{\text{steuer}}} \right|_{i_{\text{steuer } 0}, x_0} \cdot \Delta i_{\text{steuer}} + \left. \frac{\delta f}{\delta x} \right|_{i_{\text{steuer } 0}, x_0} \cdot \Delta x \quad (4.4)$$

Mit den Abkürzungen

$$f_i = \left. \frac{\delta f}{\delta i_{\text{steuer}}} \right|_{i_{\text{steuer } 0}, x_0}, \quad f_x = \left. \frac{\delta f}{\delta x} \right|_{i_{\text{steuer } 0}, x_0} \quad (4.5)$$

und dem Weglassen des die lineare Beziehung anzeigenden Δ wird diese Gleichung zu

$$m \cdot \ddot{x} = f_i \cdot i_{\text{steuer}} + f_x \cdot x. \quad (4.6)$$

Teilt man weiter noch durch die Masse m , stellt um und kürzt $p = \frac{f_x}{m}$ und $k = \frac{f_i}{m}$ ab, erhält man die Differentialgleichung 2. Ordnung

$$\boxed{\ddot{x} - p \cdot x = k \cdot i_{\text{steuer}}} \quad (4.7)$$

zur Beschreibung des ungerichteten Systems.

Die Streckenübertragungsfunktion

$$G_S(s) = \frac{I_{\text{steuer}}(s)}{X(s)} = \frac{k}{s^2 - p} \quad (4.8)$$

ergibt sich aus der Laplace-Transformation der Differentialgleichung in den Frequenzbereich und anschließendem Umstellen. Die Konstanten p (in $\frac{\text{mm}^2}{\text{s}^2}$) und k (in $\frac{\text{mm}^3}{\text{mA} \cdot \text{s}^2}$) sind als Streckenparameter vor dem Entwurf einer Regelung zu bestimmen [8].

Da die Steuer- und Regelstreckenglieder, der Leistungsverstärker und die Abstandssensorik des Experimentaufbaus mit hoher Bandbreite ausgelegt sind, kann bei einem vereinfachten dynamischen Systemmodell von deren proportionaler Wirkung ausgegangen werden [6]. Wie bereits am Ende der Abschnitte 3.1.1 und 3.1.2 erwähnt, verifizieren entsprechende Messreihen diese Annahme. Für die Umrechnung der Größen i_{steuer} und x des Systemkerns auf die Schnittstellensignale u_{steuer} und u_x (vgl. Abbildung A.8 im Anhang auf Seite 63) ergeben sich so die in der Versuchsdurchführung verwendeten, experimentell ermittelten Konvertierungsparameter

$$\text{volt2curr} = \frac{i_{\text{steuer}}}{u_{\text{steuer}}} = 283,0 \frac{\text{mA}}{\text{V}} \quad (4.9)$$

und

$$\text{curr2volt} = \frac{1}{\text{volt2curr}} \quad (4.10)$$

bei Konvertierungen von Steuerspulenstrom und Stellsignal, sowie

$$\text{volt2pos} = \frac{x}{u_x} = -1,0 \frac{\text{mm}}{\text{V}} \quad (4.11)$$

zur Errechnung der Position des Körpers aus dem Messsignal. Diese Konstanten sind grundsätzlich bei Versuchsbeginn, beispielsweise durch Ausführung des Skripts `load_conv_param` in den Workspace zu laden.

4.2. Klassische Ermittlung der Streckenparameter

Angelehnt an die ursprüngliche PC-Experimentsteuerung erfolgt die Ermittlung der die Streckenparameter in diesem Abschnitt durch die Aufnahme der statischen Kennlinie, sowie der Sprungantwort. Die beiden entstehenden Graphen entsprechen zwei Gleichungen eines bestimmten Gleichungssystems, welche zur eindeutigen Berechnung der zwei unbekanntenen Konstanten p und k aus dem mathematischen Modell mindestens nötig sind.

4.2.1. Statische Kennlinie

Untersucht man Positionen der Ruhelagen des Körpers bei verschiedenen Steuerpulvenströmen, an denen sich magnetische Kraft und Gewichtskraft genau aufheben, erhält man die Punkte einer statischen Kennlinie der Regelstrecke. Im mathematischen Modell ist $\ddot{x} = 0$ zu setzen, was Gleichung A.7 nach Umstellen auf

$$x(i_{\text{steuer}}) = -\frac{k}{p} \cdot i_{\text{steuer}} \quad (4.12)$$

reduziert. Aus der Steigung der statischen Kennlinie ist also das Verhältnis $\frac{k}{p}$ bestimmbar [6, 8].

Sobald die Konvertierungsparameter mit dem Skriptaufruf in der MATLAB-Kommandozeile `load_conv_param` geladen worden sind, kann die Messung beginnen. `static_data = get_static_data(volt2curr, curr2volt, volt2pos)` initialisiert die dialoggeführte Messpunktaufnahme und speichert die aufgenommenen Daten in der Strukturvariablen `static_data`.

Nach Eingabe eines Startwertes des aktuellen Messpunkts für den Steuerpulvenstrom ist der magnetische Körper durch Verstellen der Ablage in die Position, bei der er gerade noch nicht abhebt zu bringen und dies im Dialog der Kommandozeile zu bestätigen. Jetzt wird der Spulvenstrom in einer Sekunde rampenförmig um 500 mA erhöht. Sobald die Magnetkraft größer als die Gewichtskraft ist, hebt der Körper ab. Das Programm sucht in den aufgenommenen Daten den Punkt, an dem das Positionssignal über einen Schwellenwert wächst, so dass von einem Abheben des Körpers auszugehen ist. Das Messwertepaar aus Steuerpulvenstrom und zugehörigem Positionswert wird dann der Ausgabestrukturvariablen `static_data` angehängt, die nach Beendigung der Messpunktaufnahme im Workspace zur weiteren Verarbeitung bereit steht.

Der genaue Programmaufbau und -ablauf ist aus dem Listing der kommentierten Skriptdatei im Anhang A.3.2 ab Seite 65 ersichtlich.

Als sinnvoll herausgestellt hat sich die Aufnahme von sieben Messpunkten, bei Steuerpulvenstromwerten von -750 mA bis $+750 \text{ mA}$ in Schritten von 250 mA (ähnlich wie in [8]). Mit diesen Werten erscheint nach Aufrufen der Funktion `plot_static_data(static_data.Ist, static_data.x)` mit den Zeilenvektoren der Ausgabestrukturvariablen von `get_static_data` als Parameter, die in Abbildung 4.2 auf der nächsten Seite gezeigte graphische Darstellung der Messreihe auf dem Bildschirm. Nach Ausgabe dieser Grafik über einen am PC verfügbaren Drucker sind die Messpunkte mit einem Lineal durch eine Gerade zu nähern (vgl. Abbildung A.10 auf Seite 78 im Anhang). Die Steigung dieser Geraden entspricht dem Verhältnis $\frac{k}{p}$ mit negativem Vorzeichen:

$$\frac{\Delta x}{\Delta i_{\text{steuer}}} = -\frac{k}{p}. \quad (4.13)$$

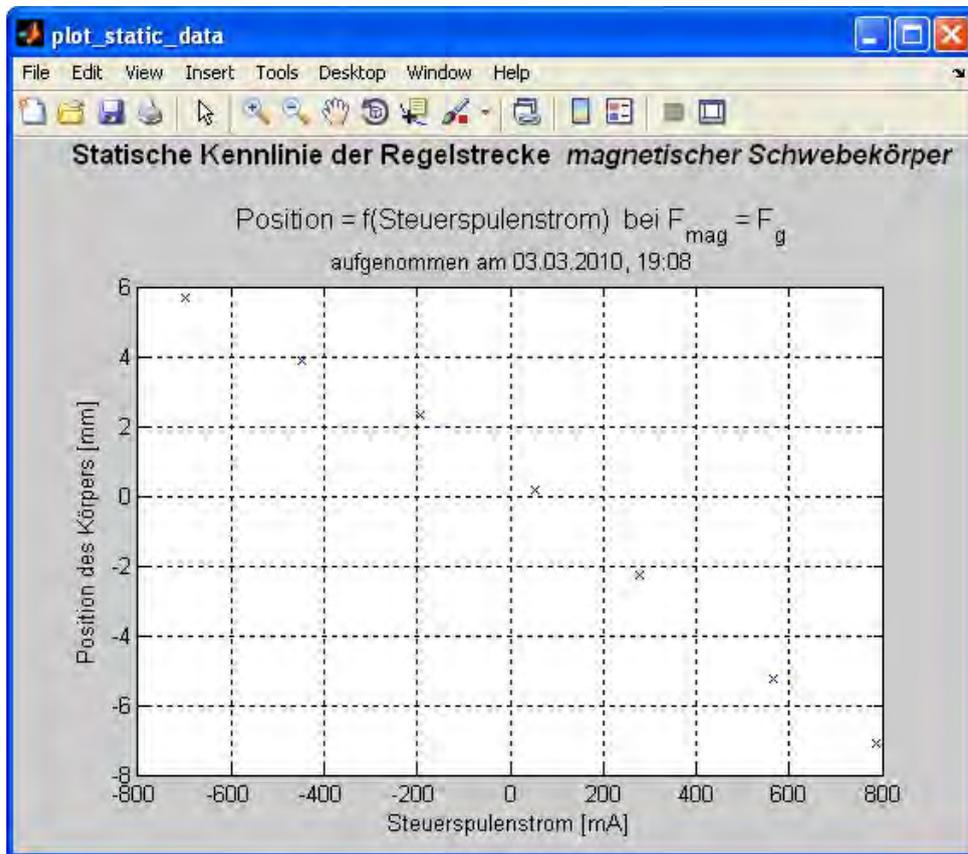


Abbildung 4.2: graphische Darstellung der Messpunkte zur Ermittlung der statischen Kennlinie

Eine Wahl der Geradenpunkte $x(500 \text{ mA}) = -4,2 \text{ mm}$ und $x(-500 \text{ mA}) = 4,4 \text{ mm}$ (wie in Abb. A.10) ergibt ein Verhältnis

$$\frac{k}{p} = -\frac{\Delta x}{\Delta i_{\text{steuer}}} = \frac{4,2 \text{ mm} + 4,4 \text{ mm}}{500 \text{ mA} - (-500 \text{ mA})} = \frac{8,6 \text{ mm}}{1000 \text{ mA}} = 8,6 \cdot 10^{-3} \frac{\text{mm}}{\text{mA}}. \quad (4.14)$$

Abhängig von der Wahl der Näherungsgeraden variiert dieser Wert. Auch der Zustand des Experimentaufbaus, beispielsweise der Grad von dessen Erwärmung, haben einen geringen Einfluss darauf. Der in Gleichung 1.1 angegebene Zahlenwert des Verhältnisses $\frac{k}{p}$ entspricht in etwa einem Mittel mehrerer Versuchsreihen, bei dem aus genannten Gründen Abweichungen von bis zu 10% möglich sind.

4.2.2. Sprungantwort

Löst man Gleichung A.7 mit den Gesetzen der Differentialrechnung oder über die Sätze der Laplace-Transformation, erhält man die Sprungantwort der Regelstrecke als Ansatz

$$x(t) = i_{\text{steuer}} \cdot \frac{k}{p} \cdot 0,5 \cdot [e^{\sqrt{p} \cdot t} + e^{-\sqrt{p} \cdot t} - 2] \quad (4.15)$$

für die Bestimmung der Konstanten p , mit i_{steuer} als Sprunghöhe. Die anklingende Exponentialfunktion überwiegt bei größeren Werten von $\sqrt{p} \cdot t$ und $e^{-\sqrt{p} \cdot t}$ kann vernachlässigt werden. Nach Umstellen ergibt sich so

$$x(t) + i_{\text{steuer}} \cdot \frac{k}{p} = i_{\text{steuer}} \cdot \frac{k}{p} \cdot 0,5 \cdot e^{\sqrt{p} \cdot t} \quad (4.16)$$

und nach Bildung des Logarithmus

$$\ln [x(t) + i_{\text{steuer}} \cdot \frac{k}{p}] = \sqrt{p} \cdot t + \ln [i_{\text{steuer}} \cdot \frac{k}{p} \cdot 0,5]. \quad (4.17)$$

Im halblogarithmischen Maßstab entspricht das einer Gleichung der Form $y = ax + b$, wobei $a = \sqrt{p}$ die Steigung der Funktion nach dem Einschwingen ist. Mit dem Ergebnis aus Gleichung B2.1 sind die gesuchten Größen p und k nun zahlenmäßig berechenbar [8].

Das in Abbildung 4.3 auf der folgenden Seite gezeigte Simulink-Modell `get_step_data` dient der Aufnahme der Sprungantwort. Das Programm benötigt die in beiden *Step*-Blöcken verwendete Variable `stepsize` im Workspace, welche die Sprunghöhe in *mA* angibt. Zu einem zweckmäßigen Wert von 280 mA lautet das Kommandozeilenstatement `stepsize = 280`. Auch die Konvertierungsparameter müssen geladen sein. Die Standardkonfiguration für das Real-Time Windows Target ist bereits vorhanden. Über *Tools* → *External Mode Control Panel* → *Signal & Triggering* erfolgt im Feld *Duration* die Einstellung der Größe des Datenpuffers bei der Aufnahme. Ein Wert von mindestens 600 ist notwendig, da Signale 300 Millisekunden lang mit 2 kHz abgetastet werden. Um die aufgenommenen Daten auch im Workspace zu speichern, ist nach Doppelklicken auf den *Scope*-Baustein (rechts unten im Blockschaltbild) der Button *Parameters* in der Symbolleiste zu drücken und im sich öffnenden Fenster der Reiter *Data history* zu wählen. Nach Setzen des Hakens in der Checkbox *Save data to workspace*, wird *Structure with time* als *Format* gewählt und als *Variable name* beispielsweise `step280` vergeben. Dies ist der Variablenname, unter dem die Datenstruktur nach der Ausführung im Workspace erscheint.

Vor dem Beginn der Messung ist der magnetische Körper noch durch Verstellen des Scherentisches in den Betriebspunkt zu bringen, so, dass er gerade noch nicht abhebt.

Der C-Code des Modells wird generiert, indem man entweder aus dem Menü *Tools*→*Real-Time Workshop*→*Build Model* wählt oder *Strg+B* drückt. Nachdem die in der MATLAB-Kommandozeile verfolgbare Generierung und Kompilierung des Codes abgeschlossen ist, wird die Datenaufnahme mit dem Menüaufruf *Simulation*→*Connect To Target*, dem Button *Connect To Target* in der Symbolleiste (links neben der Anzeige der *Simulation stop time*) oder mit *Strg+T* initialisiert und mit dem Button *Start real-time code*, ebenfalls in der Symbolleiste (links neben *Connect To Target*-Button), gestartet. Der nun auf dem Target laufende Code gibt einen der Sprunghöhe des Steuerspulenstroms entsprechenden Spannungssprung über *AO 0* aus und liest diesen Spannungsverlauf und das Positionssignal auf *AI 0* und *AI 1* ein.

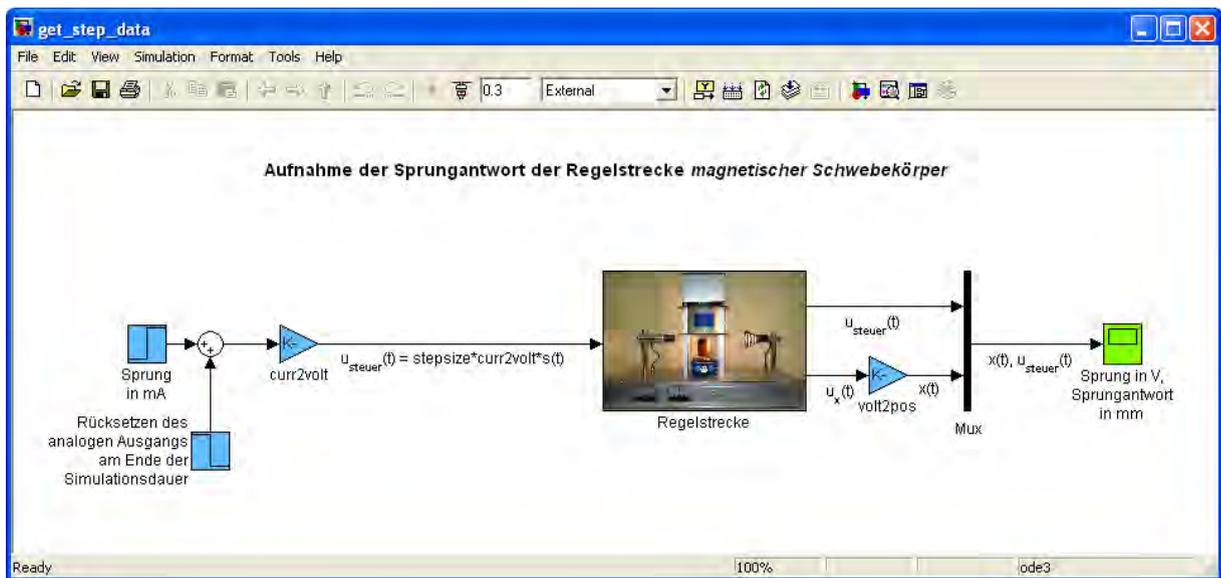


Abbildung 4.3: Blockschaltbild des Simulink-Modells `get_step_data`

Alle übrigen Einstellungen sind nach Doppelklick auf die jeweiligen Blöcke des Modells leicht einsehbar. Zu beachten ist, dass die Nummerierung der Ein- und Ausgangskanäle der Datenerfassungskarte von der bei *Analog Input* und *Analog Output* verwendeten abweicht. Es gilt: $AI/O\ i = Input/Output\ Channel\ (i + 1)$.

Die eingelesenen Daten mit bereits konvertiertem Positionssignal sind nach Doppelklick auf den *Scope*-Block und dem Drücken des *Autoscale*-Buttons in der Symbolleiste des geöffneten Fensters *Sprung in V, Sprungantwort in mm* sichtbar. Diese Daten wurden im Workspace in der Strukturvariablen `step280` abgelegt. Mit der Funktion `plot_step_lin` kann in MATLAB die Sprungantwort durch den Kommandozeilenaufruf `plot_step_lin(step280.time, step280.signals.values(:,2), 280)`, wie in Abbildung 4.4 auf der folgenden Seite gezeigt, dargestellt werden. Hierbei entspricht das Teilstatement `step280.signals.values(:,2)` der zweiten Spalte der Unterstruktur `values`, in dem sich die Positionswerte befinden.

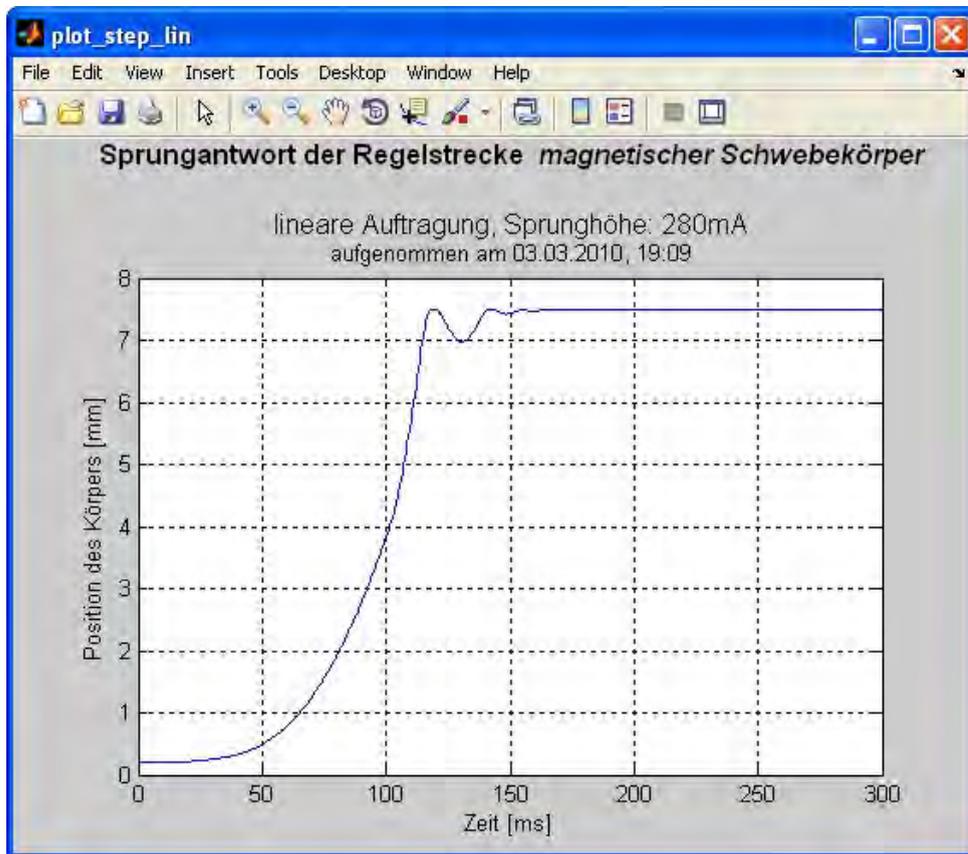


Abbildung 4.4: Rohdaten der Sprungantwort

Man sieht, dass auch das Anschlagen des Körpers am unteren Ende des Spulensystems aufgezeichnet wurde. Diese Werte stören bei der Parameterermittlung. Der Funktionsaufruf `prep_step280 = prep_step_data(step280, volt2curr)` extrahiert die relevanten Daten aus der Struktur `step280`, bereitet sie auf und strukturiert sie in der Ausgabevariablen `prep_step280` neu. Das Statement `plot_step_lin(prepare_step280.t, prepare_step280.x, 280)` zeigt dann den für die weitere Betrachtung notwendigen Teil der aufgenommenen Daten (vgl. Abbildung A.11 auf Seite 79 im Anhang) in Analogie zu Gleichung 2.2.

Die Studenten führen die in `prep_step_data` implementierte Extraktion und Aufbereitung der Daten selbst Schritt für Schritt durch. Sie lagern diese Funktionalität danach in die zu vervollständigende Version der o.g. Funktion, `prep_step_data_todo`, aus und testen diese nach erneuter Aufnahme einer Sprungantwort.

Die graphische Ermittlung der Konstanten p bzw. von \sqrt{p} erfordert nach Gleichung 4.17 eine halblogarithmische Auftragung der Sprungantwort. Dies geschieht durch das Statement `plot_step_log(prepare_step280.t, prepare_step280.x, 280)`. Ein Fenster mit der Darstellung aus Abbildung 4.5 auf der nächsten Seite öffnet sich. Nach Ausdrucken ist dieser Graph im linearen Bereich durch Anlegen einer Geraden zu nähern (vgl. Abbildung A.12 im Anhang, Seite 80).

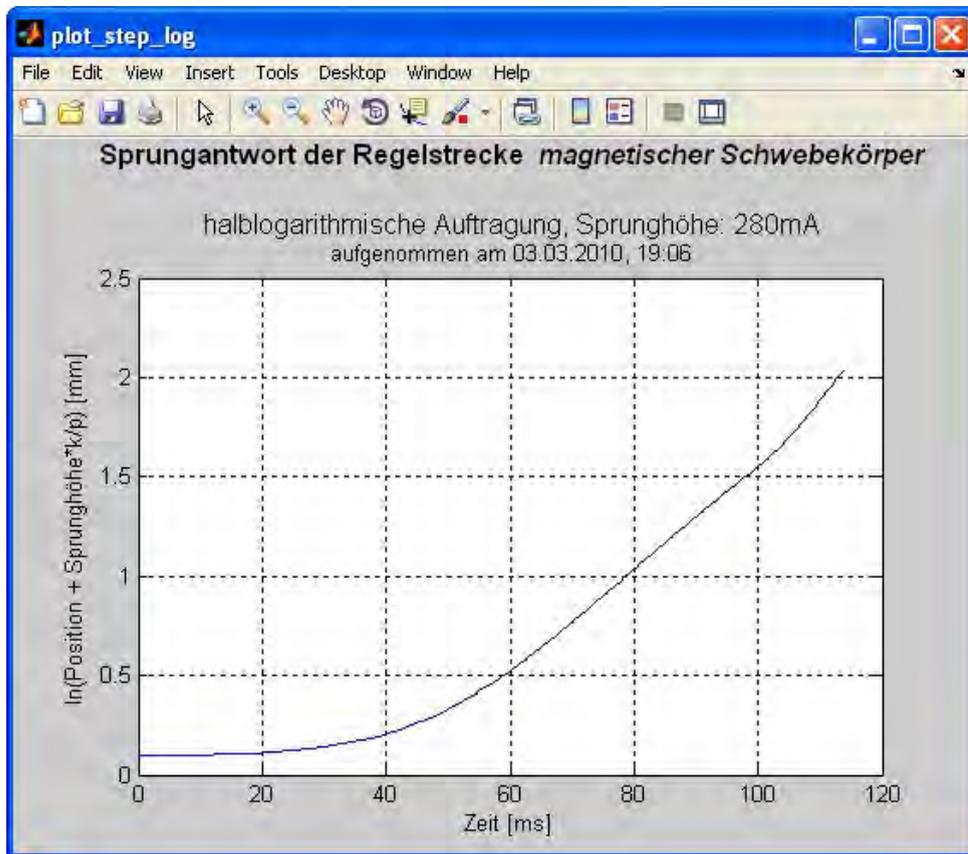


Abbildung 4.5: Halblogarithmische Auftragung der Sprungantwort

Wählt man die Punkte $x(0,10\text{ s}) = 1,58\text{ mm}$ und $x(0,05\text{ s}) = 0,28\text{ mm}$ dieser Geraden (wie in in Abb. A.12), wird

$$\sqrt{p} = \frac{\Delta \ln [x(t) + i_{\text{steuer}} \cdot \frac{k}{p}]}{\Delta t} = \frac{1,58\text{ mm} - 0,28\text{ mm}}{0,10\text{ s} - 0,05\text{ s}} = \frac{1,30\text{ mm}}{0,05\text{ s}} = 26,0 \frac{\text{mm}}{\text{s}}$$

bzw.

$$p = \left(26,0 \frac{\text{mm}}{\text{s}}\right)^2 = 676 \frac{\text{mm}^2}{\text{s}^2}. \quad (4.18)$$

Mit Gleichung 1.1 ist so

$$k = p \cdot \frac{k}{p} = 676 \frac{\text{mm}^2}{\text{s}^2} \cdot 8,6 \cdot 10^{-3} \frac{\text{mm}}{\text{mA}} = 5,81 \frac{\text{mm}^3}{\text{mA} \cdot \text{s}^2}. \quad (4.19)$$

Auch der auf diese Weise bestimmte Wert von p variiert ein wenig, wodurch wiederum der Wert von k weiter beeinflusst wird. Die Variationen haben jedoch keinen merklichen Effekt auf die später mit Hilfe dieser Streckenparameter entworfene Regelung.

4.2.3. Vergleich: reale Sprungantwort mit theoretischem Verlauf

Mit `comp_id_re` und deren Parametern Sprunghöhe, den Konstanten p und k , sowie den aufbereiteten Daten der Sprungantwort ist ein Vergleich der Sprungantwort aus Gleichung 2.2 mit den experimentellen Daten möglich. Der Aufruf `comp_id_re (prep_step280.t, prep_step280.x, 676, 5.81, 280)` öffnet das in Abbildung 4.6 gezeigte Fenster.

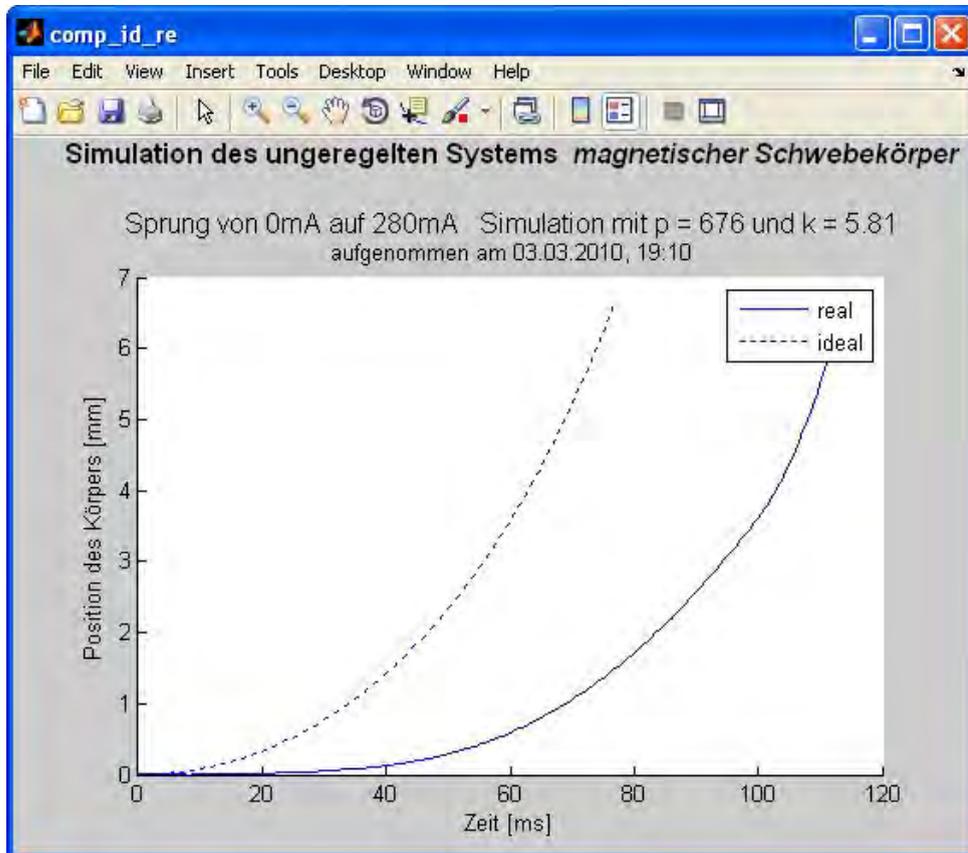


Abbildung 4.6: Vergleich der Sprungantwort aus theoretischem Modell mit realen Daten bei klassischer Streckenanalyse

Wie man sehen kann, weichen die im Prinzip ähnlichen Kurvenverläufe voneinander ab. Das reale System ist langsamer. Der Grund hierfür ist zum einen, dass sich Magnetfelder erst in bestimmter Zeit aufbauen. Zum anderen müsste für die Erzeugung eines echten Stromsprungs eine unendlich hohe Spannung angelegt werden ($u = L \cdot \frac{di}{dt}$)¹.

¹Vermerk von Prof. Dr. Schneider, Hochschule Rosenheim, zur Musterlösung des ehemaligen Versuchs

4.3. Ermittlung der Streckenparameter mit der System Identification Toolbox

Von den vielen durch die System Identification Toolbox gebotenen Ansätzen hat die Parameterschätzung mittels Grey-Box-Modellierung die mit Abstand besten Resultate gebracht. Beispielsweise bei Vorgabe lediglich einer Differentialgleichung zweiter Ordnung zur Näherung des Graphen der Sprungantwort mittels eines linearen, parametrischen Modells über die GUI des *ident*-Tools der Toolbox, wichen die Ergebnisse meist erheblich von den gewünschten bzw. erwarteten ab. Die Laplacetransformierten wiesen beispielsweise nicht vernachlässigbare Koeffizienten der ersten Potenz von s im Nenner der Streckenübertragungsfunktion $G_S(s)$ und Parameterwerte von k mit negativem Vorzeichen auf (vgl. Gleichung A.8). Aus diesem Grund und um den Rahmen der Diplomarbeit nicht zu sprengen wird auf eine Beschreibung derartiger Verfahren verzichtet.

4.3.1. Grey-Box-Modell zur Parameterschätzung

Begleitende Literatur zur System Identification Toolbox (z.B. [5] und [11]) empfiehlt immer dann die Anwendung eines Grey-Box-Modells, wenn die physikalischen Gegebenheiten des zu untersuchenden Systems einsehbar sind. So kann MATLAB die Struktur des mathematischen Streckenmodells vorgegeben werden, was den Rechenaufwand bei der Parameterschätzung reduziert und, wie auch aus einem Vergleich der Grafiken in den Abschnitten 5 und 4.3.2 ersichtlich ist, sehr gute mathematische Näherungen des realen Systemverhaltens liefert.

Die im folgenden beschriebene Methode zur Ermittlung der Streckenparameter erfordert ein *iddata*-Objekt mit den wie in Abschnitt 1 aufgenommenen und aufbereiteten Daten der Sprungantwort. Des Weiteren ist die Übertragung von Gleichung A.7 in den Zustandsraum, sowie ein Zeilenvektor mit den Startwerten der Parametern p und k für den Schätzalgorithmus nötig, um damit ein *idgrey*-Modell der Strecke zu erstellen. Mit *iddata*-Objekt und *idgrey*-Modell kann dann die Parameterschätzung initialisiert werden [11].

```
step280_obj = iddata(prepare_step280.x,prepare_step280.Ist,1/2000)
```

 erzeugt im Workspace ein *iddata*-Objekt mit den Daten der aufbereiteten Sprungantwort der Regelstrecke aus Abschnitt 1 unter dem Namen `step280_obj`. `1/2000` ($= 0,0005$) ist die Dauer des Sampleintervalls in Sekunden. `param = [700 7]` legt den Zeilenvektor `param` mit den Startwerten bei der Schätzung der Streckenparameter an. Versuche haben gezeigt, dass diese Werte beliebig wählbar sind, ohne dass sich das Ergebnis der Parameterschätzung merklich ändert. Die Wahl von `param(1) = 1000` ($= p$) und `param(2) = 10` ($= k$) hat einzig und allein den Zweck der Anschaulichkeit, da der Rechenaufwand für das im folgenden beschriebene, relativ einfache Modell ohnehin gering ist und von der Wertewahl so kaum beeinflusst wird.

Eine Zustandsraumdarstellung ermöglicht die Überführung einer Differentialgleichung n-ter Ordnung in ein Gleichungssystem mit n Differentialgleichungen 1. Ordnung. Um Gleichung A.7 in den Zustandsraum übertragen zu können, definiert man

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = p \cdot x_1 + k \cdot i_{\text{steuer}}. \quad (4.20)$$

Es ergibt sich so die Vektordifferentialgleichung 1. Ordnung

$$\dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{b} \cdot i_{\text{steuer}} \quad (4.21)$$

mit Zustandsmatrix $\underline{A} = \begin{pmatrix} 0 & 1 \\ p & 0 \end{pmatrix}$, Steuervektor $\underline{b} = \begin{pmatrix} 0 \\ k \end{pmatrix}$ und Zustandsvektor $\underline{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ in Matrixschreibweise

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ p & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ k \end{pmatrix} \cdot i_{\text{steuer}}. \quad (4.22)$$

Üblicherweise wird im Regelkreis die Eingangsgröße mit u , die Ausgangsgröße mit y bezeichnet. Es gilt $y = x$, und

$$y = \underline{c}^T \cdot \underline{x} \quad (4.23)$$

wird mit dem Ausgangsvektor $\underline{c}^T = (1 \ 0)$ in Matrixschreibweise zu

$$y = x_1 = (1 \ 0) \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \quad (4.24)$$

Es sei darauf hingewiesen, dass die Position x des magnetischen Körpers nicht mit dem Zustandsvektor \underline{x} oder dessen Elementen zu verwechseln ist (mathematische Ausführungen in diesem Abschnitt nach [6, 7, 11]).

Im allgemeinen, auch für MIMO²-Systeme geltenden Fall, schreibt man die Gleichungen 4 und C.3 mit der üblichen Notation als

$$\dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{B} \cdot \underline{u} \quad (4.25)$$

und

$$y = \underline{C} \cdot \underline{x} \quad (4.26)$$

²Multiple Input Multiple Output

Die dabei verwendeten Bezeichnungen der Zustandsraummatrizen finden sich auch in der MATLAB-Dokumentation der System Identification Toolbox, genauer, bei der Beschreibung des Vorgehens zur Grey-Box-Modellierung wieder. Die Erzeugung eines linearen *idgrey*-Modells erfordert hiernach die Bereitstellung dieser Matrizen als Ausgangsvariable einer entsprechenden Funktion vorgegebener Form. Das ist die Funktion `get_state_param`. The MathWorks formuliert die Zustandsraumdarstellung durch Vorgabe eines Fehlervektors, von Rückwirkungen des Systemausgangs auf dessen Eingang und von Anfangswerten für \underline{x} noch allgemeiner. Für die Umsetzung der Gleichungen 4 und C.4 wurden die dafür vorgesehenen Variablen K , D und x_0 auf Null gesetzt (vgl. Listing im Anhang A.3.7 auf Seite 73).

Die Funktion *idgrey* wird mit `get_state_param` als Funktionsparameter zum Anlegen eines Grey-Box-Modells aufgerufen. Weitere Parameter beim Aufruf sind der Vektor `param`, die Angabe, ob es sich um ein zeitdiskretes oder -kontinuierliches Modell handeln soll (continuous: 'c'), die Länge des Abtastintervalls (bei einem zeitkontinuierlichen Modell gleich 0 zu setzen) und eventuell von `get_state_param` benötigte Hilfsvariablen (keine Hilfsvariablen, also beliebiger Wert; verwendeter Wert: 0). Das Statement lautet insgesamt, mit der Ausgabevariablen `mag_grey_mod`:
`mag_grey_mod = idgrey('get_state_param',param,'c',0,0).`

Da nun sowohl ein *iddata*-Objekt der Sprungantwort als auch ein *idgrey*-Modell der Strecke im Workspace vorhanden sind, kann die Schätzfunktion `pem`³ mit diesen beiden Parametern aufgerufen werden. Die Kommandozeileingabe zur Speicherung des geschätzten Modells in der Variablen `mag_est_mod` lautet `mag_est_mod = pem(step280_obj,mag_grey_mod)` und liefert als Ergebnis für die Streckenparameter

$$\mathbf{p} = 1356,8 \quad (4.27)$$

und

$$\mathbf{k} = 1,4138 . \quad (4.28)$$

Diese Werte variieren ebenfalls leicht bei verschiedenen Aufnahmen der Sprungantwort. Auf die Qualität der mit Ihrer Hilfe entworfenen Regelung hat das jedoch auch hier keinen merklichen Einfluss.

Das Ergebnis weicht signifikant von dem aus Abschnitt 4.2 ab. Eine Beurteilung dieser Abweichung wird durch die im folgenden gezeigte graphische Auswertung und deren Vergleich mit dem in Abbildung 4.6 auf Seite 23 veranschaulichten Resultat ermöglicht.

³prediction-error minimization (ein Algorithmus von MATLAB zur iterativen Näherung eines Kurvenverlaufs)

4.3.2. Vergleich: reale Sprungantwort mit theoretischem Verlauf

Der Aufruf `comp_id_re (prep_step280.t, prep_step280.x, 1357, 1.414, 280)` öffnet das unten abgebildete Fenster, analog zu Abschnitt 5,

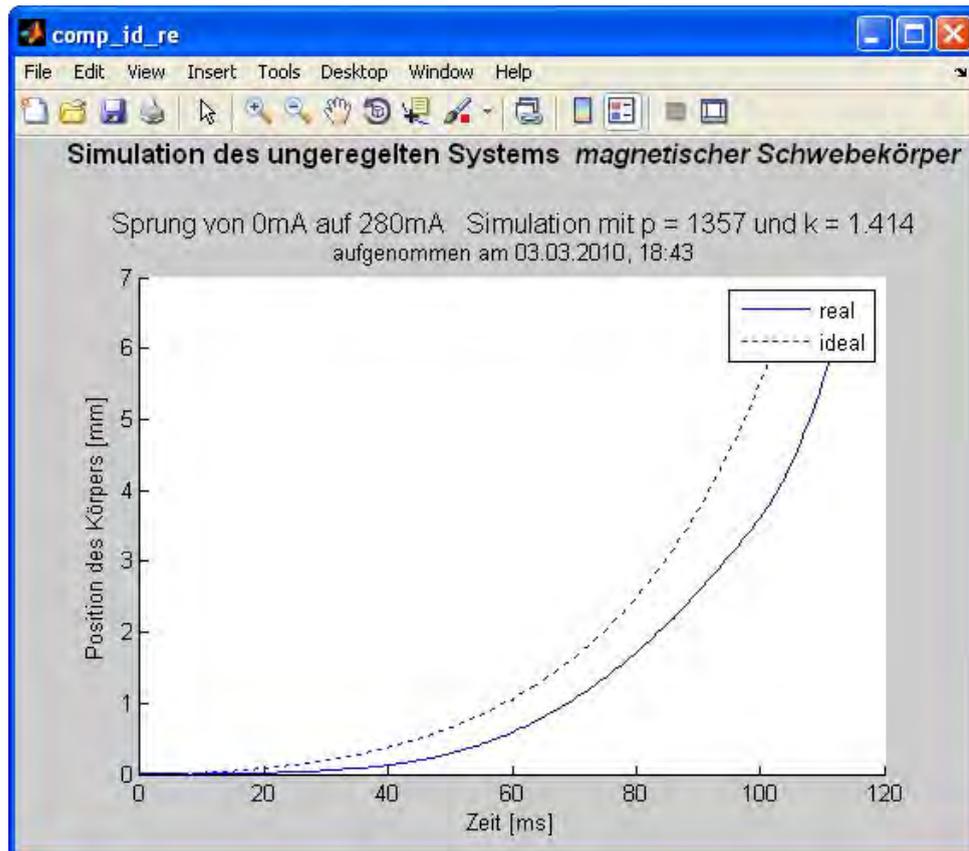


Abbildung 4.7: Vergleich der Sprungantwort aus theoretischem Modell mit realen Daten bei Grey-Box-Modellierung

Es stellt sich heraus, dass die Parameterermittlung mit Unterstützung der durch die System Identification Toolbox bereit gestellten Werkzeuge wesentlich bessere Näherungen des realen Systemverhaltens liefert, als die klassische, in Abschnitt 4.2 erörterte Vorgehensweise (vgl. dazu auch Abbildung A.13 auf Seite 81 im Anhang).

Die folgenden Ausführungen und Analysen verwenden die mittels Grey-Box-Modellierung gewonnenen Werte von p und k .

5. Reglerdesign mit der Control System Toolbox

Die Control System Toolbox ermöglicht durch eine interaktive GUI mit Darstellungen sowohl im Zeit-, als auch im Frequenzbereich einen effizienten und anschaulichen Reglerentwurf. Ziel ist die stabile Regelung des Istwerts, in diesem Fall der Position des magnetischen Körpers, bei Vorgabe von Einheitssprung oder Impuls als Sollwert.

5.1. Initalisierung

5.1.1. Erzeugung der Streckenübertragungsfunktion

Übertragungsfunktionen der Form

$$G_S(s) = \frac{b_m \cdot s^m + \dots + b_1 \cdot s + b_0}{a_n \cdot s^n + \dots + a_1 \cdot s + a_0} \quad (5.1)$$

werden in MATLAB durch Zeilenvektoren mit den Koeffizienten a_i und b_j erzeugt. Üblicherweise verwendet man für das Zählerpolynom den Variablennamen `num` (numerator), für das Nennerpolynom `den` (denominator). Das erste Element der Vektoren ist jeweils der Koeffizient der höchsten Potenz von s . Danach folgen der Reihe nach, analog zu Gleichung 4.27, alle Koeffizienten bis einschließlich a_0 bzw. b_0 .

Die Umsetzung der Übertragungsfunktion aus Gleichung A.8 mit den im vorigen Kapitel ermittelten und gerundeten Werten für die Streckenparameter erfolgt mit der Funktion `tf`¹ durch die Statements

```
num = 1.414;
den = [1 0 -1357];
Gs = tf(num,den)
```

mit dem in der Kommandozeile ausgegebenen Ergebnis

```
    1.414
-----
s^2 - 1357.
```

Die Streckenübertragungsfunktion wird in dieser Form für den Start von `sisotool` benötigt.

¹transfer function: Übertragungsfunktion

5.1.2. Vorbereitung der Benutzeroberfläche

Nach dem Aufruf `sisotool(Gs)` öffnet sich der *Control and Estimation Tools Manager*. Im Reiter *Architecture* wählt man die in Abbildung C.1 dargestellte Regelkreisstruktur.

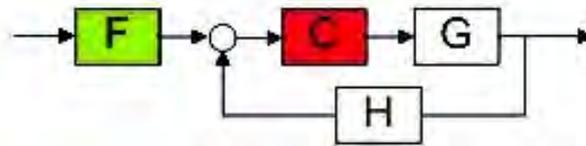


Abbildung 5.1: Regelkreisstruktur

Im Modell des Regelkreises sind die Elemente H und F nicht vorhanden. Ihre Übertragungsfunktion ist deshalb über den Button *System Data ...* gleich 1 zu setzen

Es ist zweckmäßig, die Fenster *Control and Estimation Tools Manager* mit dem *Compensator Editor*, *SISO Design* und *LTI Viewer* so auf dem Bildschirm zu arrangieren, dass alle drei gleichzeitig sichtbar sind. Diese Darstellungen sind untereinander verbunden und werden bei Änderungen im *Compensator Editor* oder im *SISO Design* automatisch aktualisiert. So kann die Systemreaktion im Zeitbereich bei Veränderungen der Werte von Polen, Nullstellen und Verstärkung in der Reglerübertragungsfunktion oder im Bode-Diagramm des Regelkreises beobachtet werden.

Den *LTI Viewer* öffnet man im Reiter *Analysis Plots*. Als *Plot Type* für *Plot 1* wird *Step* ausgewählt und der Haken für den Inhalt des Plots bei *Closed Loop r to y* gesetzt, was nach Drücken des Buttons *Show Analysis Plot* die Antwort des geschlossenen Regelkreises auf den Einheitssprung (Sollwert von 0 mm auf 1 mm) darstellt. Wichtig ist der Haken in der Checkbox *Real-Time Update*

Das *SISO Design* wird über den Reiter *Graphical Tuning* aktiviert. Die Wahl von *Root Locus* für *Plot 1* und *Open-Loop Bode* für *Plot 2*, sowie *Open Loop 1* für beide Darstellungen zeigt nach drücken des Buttons *Show Design Plot* die Wurzelortskurve und das Bode-Diagramm des Systems.

5.2. Entwurf

Die Reglerübertragungsfunktion C wird im *Compensator Editor* des *Control and Estimation Tools Manager* in Produktform standardmäßig mit den Zeitkonstanten $T_{Ni} = \frac{1}{\omega_{Ni}}$, $T_{Pi} = \frac{1}{\omega_{Pi}}$ und Verstärkung dargestellt. Allgemein:

$$C(s) = K \cdot \frac{(1 + T_{N1} \cdot s) \cdot (1 + T_{N2} \cdot s) \cdot \dots}{(1 + T_{P1} \cdot s) \cdot (1 + T_{P2} \cdot s) \cdot \dots} \quad (5.2)$$

Die Zeitkonstanten entsprechen dem Kehrwert der Kreisfrequenz, bei der die Pole und Nullstellen angesiedelt sind. Die Darstellung ließe sich in der Menüleiste des

Control and Estimation Tools Manager über *Edit* → *SISO Tool Preferences...* im Reiter *Options* in eine Form mit *Natural frequency* (ω) oder *Zero/pole/gain* ändern, die weiter unten folgenden Berechnungen zur Stabilität des Regelkreises bringen dann allerdings keine verwertbaren Ergebnisse. Auch die Anschaulichkeit würde nicht wesentlich gesteigert.

Die am Beginn der Abschnitte 5.2.1 und 5.2.2 angegebenen Funktionen C_{PD} und C_{PID} sind nach Rechtsklick im Bereich *Dynamics* über *Add Pole/Zero* generierbar. Markiert man die eingefügten Pole und Nullstellen, können deren Werte $s_{pi} = -\omega_{pi}$ und $s_{Ni} = -\omega_{Ni}$ im Bereich *Edit Selected Dynamics* eingegeben werden. Im Feld *Compensator* erscheint der jeweilige Wert als $T_i = \frac{1}{\omega_i}$. Die Verstärkung K ist ebenfalls in diesem Feld angebbbar. Während diesen Eingaben kann man beobachten, wie sich Darstellungen im *LTI Viewer* und im *SISO Design* anpassen. Die angegebenen Pole und Nullstellen, sowie auch der gesamte Graph können sowohl in der Wurzelortskurve wie auch im Bodediagramm verschoben werden. Die Werte im *Compensator Editor* folgen dabei den Veränderungen.

Um sinnvolle Startwerte für die Reglerparameter beim Entwurf zu erhalten, ist die Betrachtung des Kreises nach den Stabilitätskriterien von Hurwitz empfehlenswert. Die Übertragungsfunktion

$$G_W(s) = \frac{C(s) \cdot G_S(s)}{1 + C(s) \cdot G_S(s)} \quad (5.3)$$

des Systems mit Regler und Strecke aus Abbildung C.1 darf für einen stabilen Kreis ausschließlich Pole s_i in der linken s -Halbebene aufweisen ($Re\{s_i\} < 0$). Mit dem Hurwitz-Kriterium kann die Stabilität schon anhand der charakteristischen Gleichung aus dem Nennerpolynom von $G_W(s)$,

$$1 + C(s) \cdot G_S(s) = \frac{a_n \cdot s^n + \dots + a_1 \cdot s + a_0}{b_m \cdot s^m + \dots + b_1 \cdot s + b_0} = 0 \quad (5.4)$$

woraus folgt

$$a_n \cdot s^n + \dots + a_1 \cdot s + a_0 = 0, \quad (5.5)$$

ohne deren expliziter Lösung beurteilt werden. Alle Koeffizienten müssen für einen stabilen Regelkreis vorhanden (d.h. $a_i \neq 0$) und positiv sein [7].

5.2.1. PD-Regler

Wie leicht nachzuprüfen ist, ergibt sich mit der Streckenübertragungsfunktion aus Gleichung A.8 und der PD-Reglerübertragungsfunktion

$$C_{PD} = K_{PD} \cdot (1 + T_N \cdot s) \quad (5.6)$$

die charakteristische Gleichung

$$s^2 + k \cdot K_{PD} \cdot T_N \cdot s + (k \cdot K_{PD} - p) = 0. \quad (5.7)$$

Daraus entstehen die Stabilitätsbedingungen

$$T_N > 0 \quad \text{und} \quad K_{PD} > \frac{p}{k} \quad (5.8)$$

für die Reglerparameter. Die Werte aus den Gleichungen 4.27 und 4.28 eingesetzt bedeutet das konkret

$$K_{PD} > \frac{1356,8}{1,4138} = 959,68 \approx 1000. \quad (5.9)$$

Mit diesem Ausgangspunkt und unter Berücksichtigung von physikalischen Gegebenheiten wie beispielsweise der Trägheit des Körpers kommt man mit dem oben beschriebenen Verfahren recht schnell auf das in den Abbildungen 5.3 und 5.2 gezeigte Ergebnis.



Abbildung 5.2: Einheitssprungantwort des Regelkreises mit PD-Regler im *LTI Viewer*

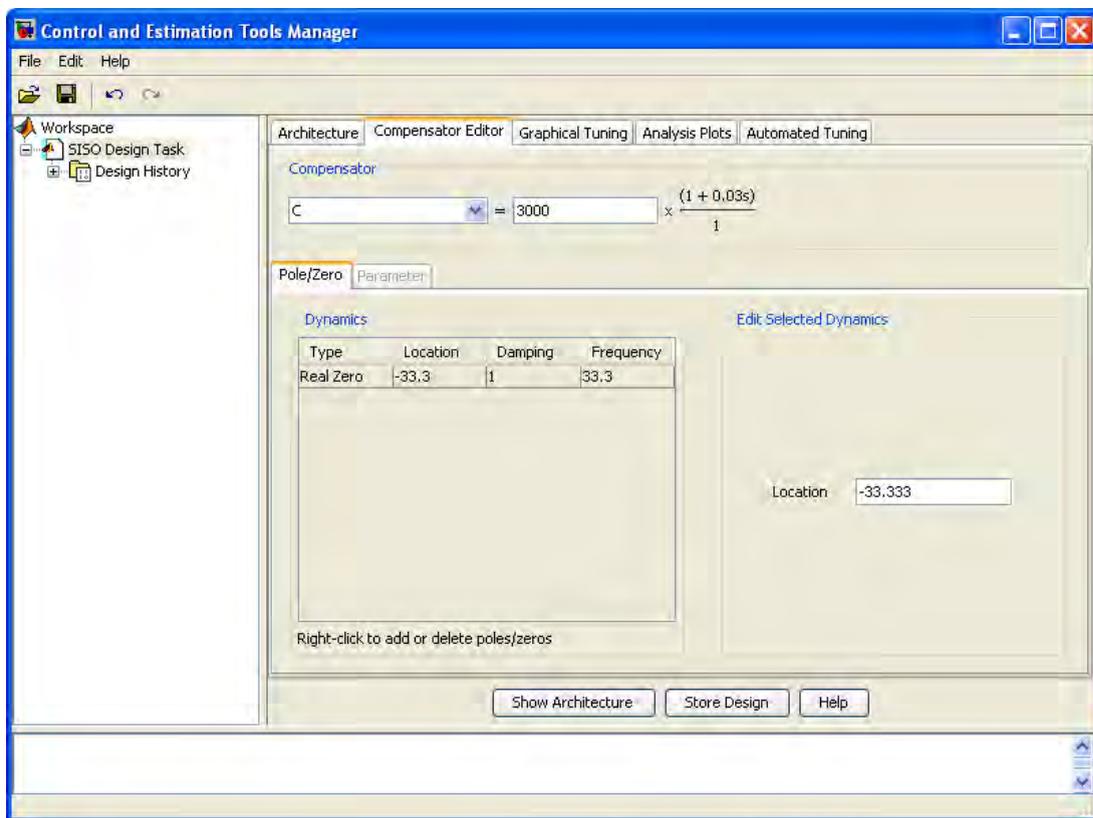


Abbildung 5.3: Übertragungsfunktion des PD-Reglers im *Compensator Editor*

Mit eingesetzten Werten lautet die ermittelte PD-Reglerübertragungsfunktion also

$$C_{PD} = 3000 \cdot (1 + 0,03 \cdot s) . \quad (5.10)$$

Die bleibende Regelabweichung des PD-Reglers stört bei der akkuraten Regelung des Sollwertes. Wird die Funktion C um einen integralen Anteil und damit zu der Übertragungsfunktion eines PID-Reglers erweitert, lässt sich diese Schwäche beheben.

5.2.2. PID-Regler

Analog zu Abschnitt 5.2.1 kann die charakteristische Gleichung mit der PID-Reglerübertragungsfunktion

$$C_{PID} = K_{PID} \cdot \frac{(1 + T_{N1} \cdot s) \cdot (1 + T_{N2} \cdot s)}{s} \quad (5.11)$$

als

$$s^3 + k \cdot K_{PID} \cdot T_{N1} \cdot T_{N2} \cdot s^2 + (k \cdot K_{PID} \cdot (T_{N1} + T_{N2}) - p) \cdot s + K_{PID} \cdot k = 0 \quad (5.12)$$

angegeben werden, woraus die Stabilitätsbedingungen

$$T_{N1} > 0, T_{N2} > 0 \quad \text{und} \quad K_{\text{PID}} \cdot (T_{N1} + T_{N2}) > \frac{p}{k} \approx 1000 \quad (5.13)$$

ableitbar sind. Wählt man z.B. $K_{\text{PID}} = 3000$, folgt daraus lediglich

$$T_{N1} + T_{N2} \gtrsim \frac{1}{3}. \quad (5.14)$$

Es ist ersichtlich, dass der analytische Ansatz aufgrund dreier unbekannter Variablen nur einen sehr groben Anhaltspunkt bietet. Mit Hilfe der Control System Toolbox, speziell der Möglichkeit des Verschiebens der Nullstellen im Bode-Diagramm des *SISO Designs*, kommt man jedoch trotzdem zügig zu dem Ergebnis in den Abbildungen 5.4 und 5.5 für die Parameter der Reglerübertragungsfunktion.

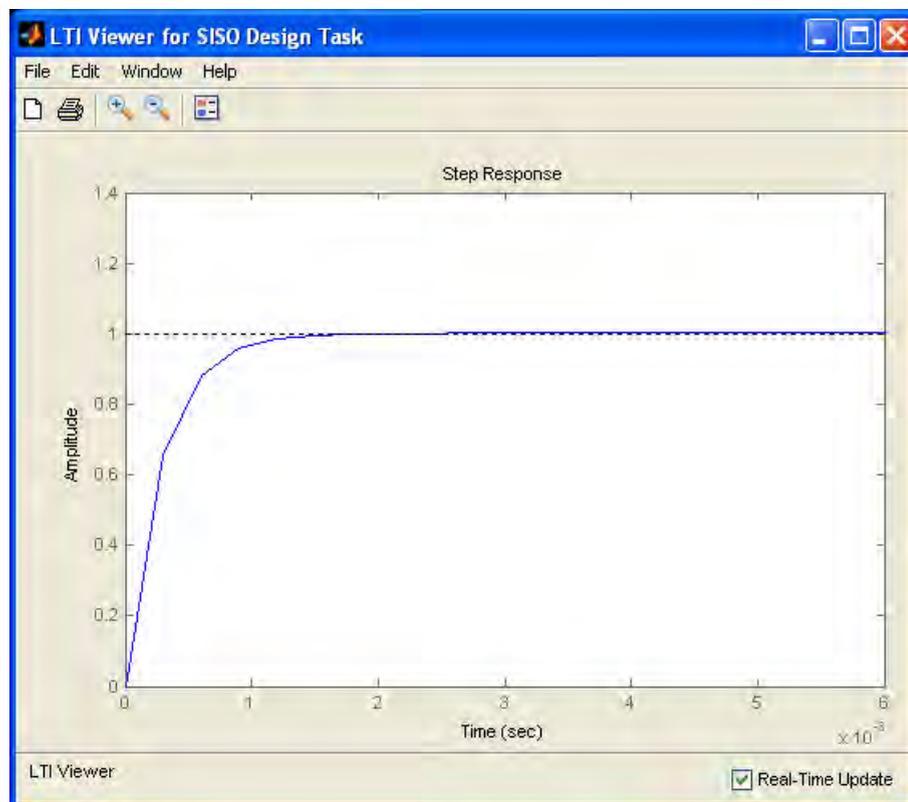


Abbildung 5.4: Einheitssprungantwort des Regelkreises mit PID-Regler im *LTI Viewer*

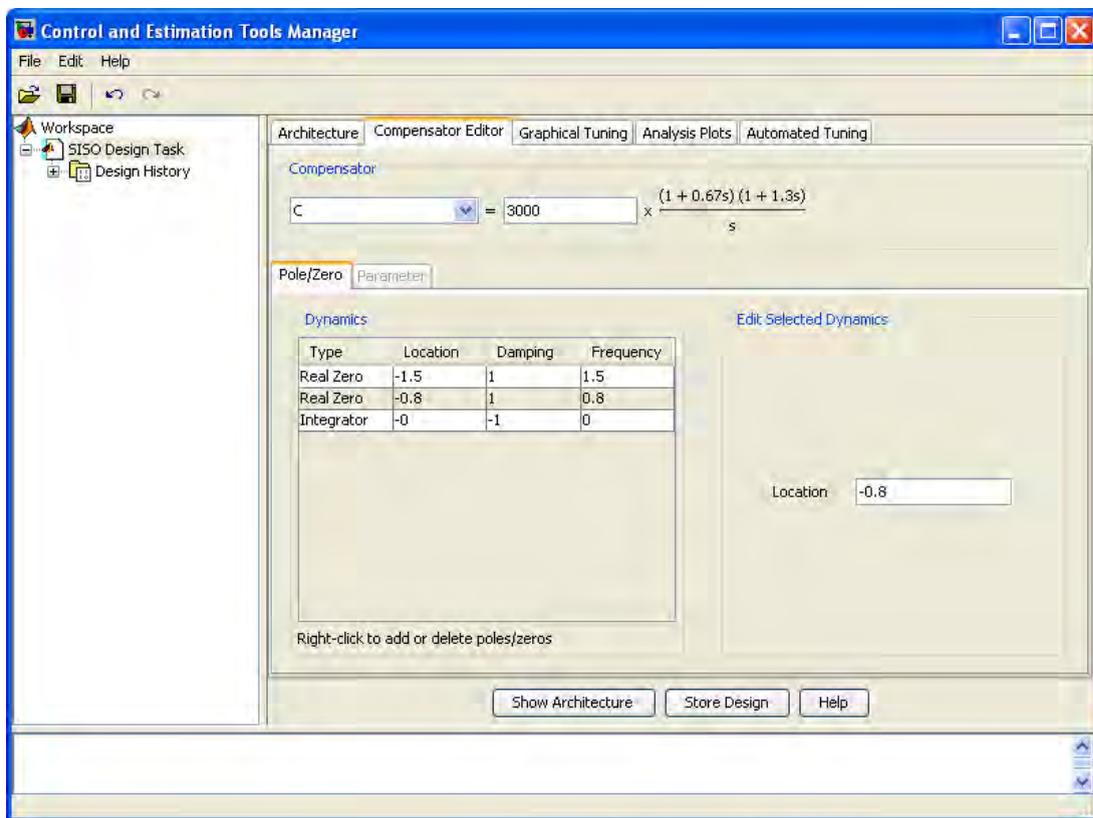


Abbildung 5.5: Übertragungsfunktion des PID-Reglers im *Compensator Editor*

Unter Beibehaltung des K -Wertes aus dem PD-Reglerentwurf ergibt sich so die PID-Reglerübertragungsfunktion

$$C_{\text{PID}} = 3000 \cdot \frac{(1 + 0,67 \cdot s) \cdot (1 + 1,3 \cdot s)}{s} . \quad (5.15)$$

6. Simulation des geschlossenen Regelkreises

Das unten abgebildete Simulink Modell `loop_sim` dient der Simulation des geschlossenen Regelkreises mit den vorher ermittelten Strecken- und Reglerparametern. Die Regelstrecke ist darin durch deren Übertragungsfunktion in einem *Transfer Fcn*-Block realisiert. Ziel der Simulationen ist einerseits die Verifikation der Ergebnisse aus dem Reglerentwurf sowie die Untersuchung und Erweiterung des regelbaren Bereichs, andererseits soll die Gefahr einer Beschädigung der Hardware reduziert werden. Voraussetzung für eine erfolgreiche Simulation ist die dem Zweck der jeweiligen Untersuchungen angepasste, ausreichend genaue Modellierung des realen Systemverhaltens.

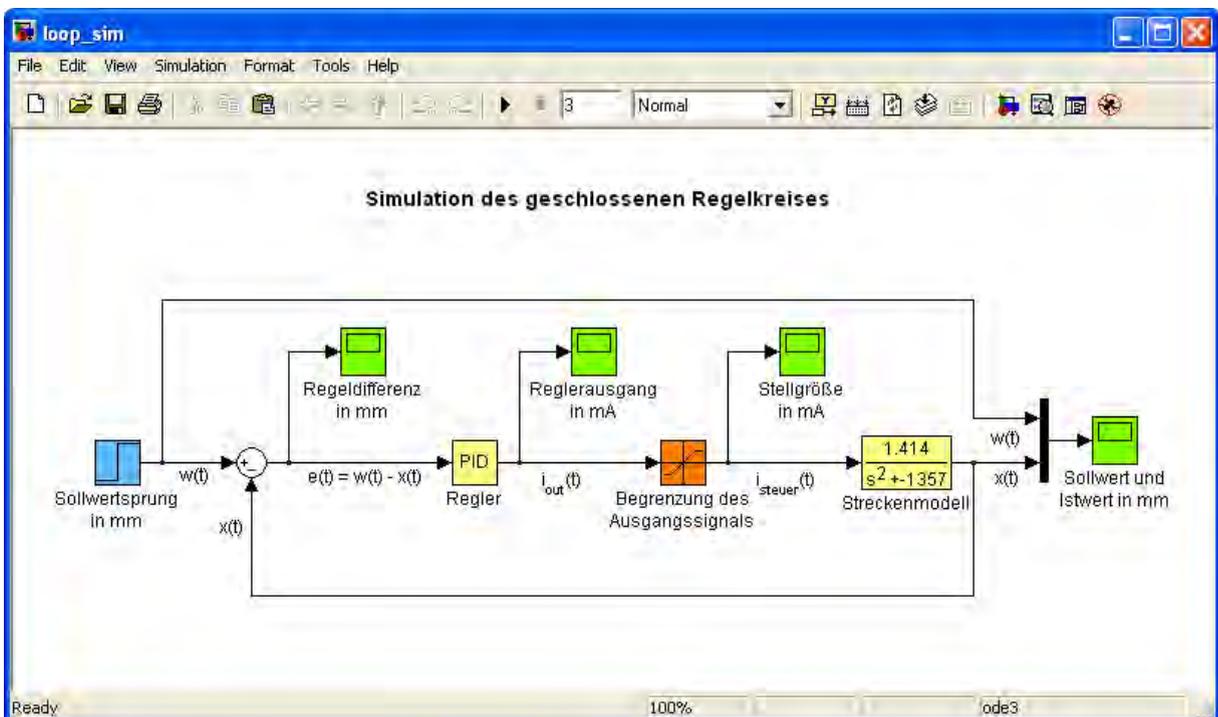


Abbildung 6.1: Blockschaltbild des Simulink-Modells `loop_sim`

6.1. Vorbereitende Überlegungen und Berechnungen

6.1.1. Realitätsnahe Modellierung

Die Grenzen des realen Streckensystems werden durch ein beschränktes Reglerausgangssignal nachgebildet. Diese Beschränkung des Stellsignals ist ein gravierender Unterschied zum idealen Systemverhalten. Sie kommt sowohl durch die Spezifikationen der analogen Ausgänge der Datenerfassungskarte als auch die damit zusammenhängende Auslegung des Leistungsverstärkers zustande. Dieser würde bei betragsmäßig größeren Eingangssignalen als 10 V im stark nichtlinearen Bereich seiner Kennlinie arbeiten, was eine Regelung der Strecke erheblich erschwert. Aus diesen Gründen limitiert der *Saturation*-Block im Modell für die Simulation des Regelkreises den Steuerspulenstrom auf den Bereich $-2800 \dots +2800 \text{ mA}$. Die Grenzwerte entsprechen der konvertierten und betragsmäßig auf der Hunderterstelle abgerundeten spezifizierten maximalen Ausgangsspannung der PCI-6014. Sie wurden etwas kleiner als die tatsächlich möglichen gewählt, weil die Qualität der Ergebnisse für das folgende Echtzeitexperiment dadurch etwas steigt.

Eine weitere Maßnahme für die wirklichkeitsnahe Simulation ist die im Abschnitt 6.4 beschriebene Ausführung des Modells unter Real-Time Workshop in Echtzeit.

6.1.2. Konvertierung der Reglerübertragungsfunktionen

Der PID-Reglerübertragungsfunktion hat in Simulink die additive Form

$$G_R(s) = P + \frac{I}{s} + D \cdot s, \quad (6.1)$$

wobei die Werte von P, I und D im Block eingestellt werden können. Bislang liegen die Übertragungsfunktionen aus dem Reglerdesign in Produktform vor. Um die Ergebnisse aus den Abschnitten 5.2.1 und 5.2.2 nutzen zu können, ist die Bereitstellung entsprechender Konvertierungsgleichungen erforderlich. Grundlage hierfür ist, nach Ausmultiplizieren der Übertragungsfunktionen C, ein Koeffizientenvergleich mit Gleichung 6.1. Im Fall der Funktion C_{PD} aus Gleichung C.9 gilt trivialerweise

$$P_{PD} = K_{PD}, \quad I_{PD} = 0 \quad \text{und} \quad D_{PD} = K_{PD} \cdot T_N, \quad (6.2)$$

C_{PID} aus Gleichung 5.11 wird durch

$$P_{PID} = K_{PID} \cdot (T_{N1} + T_{N2}), \quad I_{PID} = K_{PID} \quad \text{und} \quad D_{PID} = K_{PID} \cdot T_{N1} \cdot T_{N2} \quad (6.3)$$

in die additive Form überführt.

6.2. Verifizierung der Ergebnisse aus dem Reglerentwurf

Ausgangspunkt ist die Simulation mit den kovertierten Werten der Reglerübertragungsfunktionen aus den Gleichungen 5.10 und 5.15 im *PID*-Block. Der Regelkreis wird für beide Reglervarianten zuerst mit einer sprungartigen Sollwertänderung von 0 mm auf 1 mm (Einheitssprung) beaufschlagt.

6.2.1. PD-Regler

Mit den Konvertierungsgleichungen 6.2 ergeben sich die konkreten Werte

$$\begin{aligned} P_{PD} &= 3000 \\ \text{und } D_{PD} &= 3000 \cdot 0,03 = 90 \end{aligned} \quad (6.4)$$

für die Initialisierung des *PID*-Reglerblocks. Das in Abbildung 6.2 gezeigte Antwortverhalten entspricht in puncto stationärer Endwert den Erwartungen. Allerdings ist die Antwort des simulierten Kreises etwas langsamer als im Reglerentwurf (vgl. Abbildung 5.2). Das könnte an einem Reglerausgangssignal liegen, das nicht, wie im *sisotool* angesetzt, unendlich hohe Sprünge machen kann, sondern auf einen Maximalbetrag von 2800 mA begrenzt ist. Simulationen ohne Begrenzung bestätigen diese Annahme nicht.

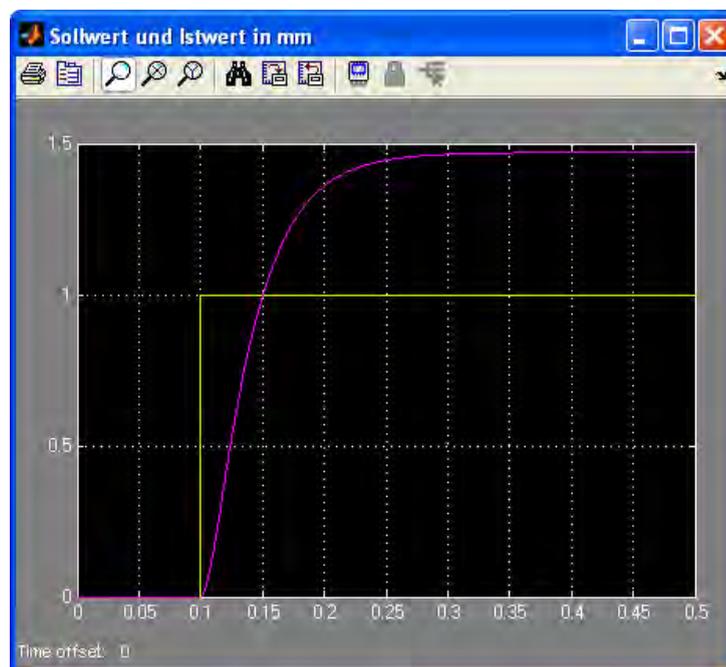


Abbildung 6.2: Simulierte Sprungantwort des Kreises mit PD-Regler

6.2.2. PID-Regler

Um die Sprungantwort des Kreises mit PID-Regler zu simulieren, werden die Ergebnisse der kovertierten Übertragungsfunktion

$$\begin{aligned} P_{\text{PID}} &= 3000 \cdot (0,67 + 1,3) = 5910 \approx 6000, \\ I_{\text{PID}} &= 3000 \\ \text{und } D_{\text{PID}} &= 3000 \cdot 0,67 \cdot 1,3 = 2613 \approx 2600 \end{aligned} \quad (6.5)$$

im Reglerblock eingetragen. Wie in Abbildung 6.3 zu erkennen ist, weicht das Simulationsergebnis von den Erwartungen ab. Das Systemverhalten ist zum einen langsamer als im *LTI Viewer* beim Reglerentwurf, zum anderen schwingt der Istwert über. Bei genauerer Betrachtung sieht man in Abbildung 5.2 ebenfalls ein sehr leichtes Überschwingen.



Abbildung 6.3: Simulierte Sprungantwort des Kreises mit PID-Regler

6.2.3. Resumee

Variationen des Simulationssetups über Menüleiste, *Simulation*→*Configuration Parameters*... liefern keine Erklärung für die beobachteten Abweichungen. Weder die Wahl der Schrittweite im *Fixed-step*-Modus des Solvers, noch ein *Variable-step* Ansatz, noch die Verwendung eines anderen Solvertyps bei der Lösung des dem Simulink-Modell zugrunde liegenden Differential- bzw. Differenzgleichungssystems bringt eine signifikante Änderung der Simulationsergebnisse. Sowohl der ursprünglich gewählte Solver, *ode¹3(Bogacki-Shampine)*, als auch die gesetzte Schrittweite für die Simulationen von 0,0005 Sekunden entsprechend der Sample-Rate wird deshalb beibehalten.

Auch der Einsatz eines *Transfer Fcn*-Blocks anstatt des PID-Blocks zur Umsetzung der Reglerübertragungsfunktion im Modell analog zu Abschnitt 5.1.1 bringt keine Änderung des in Abbildung 6.3 dargestellten Verhaltens. Erschwerend kommt hinzu, dass die im *Control and Estimation Tools Manager* in Kapitel 5 angegebenen Übertragungsfunktionen für PD- und PID-Regler ideal sind. Realisierbare Reglerübertragungsfunktionen dürfen keinen Zählergrad größer Nennergrad haben und werden von MATLAB/Simulink nicht akzeptiert. Der Nenner muss mit einem zusätzlichen Pol, der ausreichend weit von den Polen des Systems entfernt liegt (z.B. bei $\omega_p = 1000 \frac{1}{s}$ bzw. $T_p = 0,001 s$) erweitert werden. Der Praktikumsversuch würde dadurch an Übersichtlichkeit verlieren. Die Verwendung des PID-Reglerblocks ist anschaulich und relativ einfach, weshalb sie bei der Modellierung im Blockschaltbild letzten Endes bevorzugt wurde.

Die Vermutung liegt nahe, dass *sisotool* eine grundsätzlich andere Algorithmik verwendet als Simulink. In der Control System Toolbox ist beispielsweise ein zeitkontinuierliches Modell angesetzt. Es ist möglich, das Modell im Reiter *Architecture* des *Control and Estimation Tools Manager* über den Button *Sample Time Conversion*... in ein zeitdiskretes zu konvertieren. Dann tritt allerdings einerseits wieder die Schwierigkeit auf, dass für Zählergrad \leq Nennergrad eine weitere Polstelle in die Reglerübertragungsfunktion eingesetzt werden müsste, andererseits wäre bei den Studenten die z-Transformation vorauszusetzen. Insgesamt würde die Komplexität des Praktikumsversuchs u.a. durch zusätzliche Berechnungen und eine unübersichtliche Darstellung der Funktion *C* im *Compensator Editor* weiter steigen.

Der genaue Grund für die auftretenden Unterschiede konnte trotz etlichen Versuchen und ausgiebiger Recherche nicht fest gemacht werden. Bei den weiteren Simulationen und dem anschließenden Echtzeitexperiment sind diese im Prinzip maginalen Abweichungen nicht von großer Bedeutung. Die Reglerparameter werden vor der Regelung des realen Systems ohnehin noch optimiert. Der im Kapitel 5 beschriebene Reglerentwurf ist als ad-hoc-Verfahren für das schnelle Finden eines Ansatzes zur Simulation zu sehen.

Die von der Control System Toolbox gebotenen Möglichkeiten (z.B auch spezielle Methoden zur Optimierung der Reglerparameter) sind vielfältig. Die exakte Handhabung der einzelnen Werkzeuge erfordert teilweise eine längere Einarbeitungsphase. Um die verbleibenden Themengebiete nicht zu vernachlässigen, wird hier auf tiefer gehende Untersuchungen verzichtet.

¹ordinary differential equation

6.3. Bestimmung und Erweiterung des regelbaren Bereichs

Die Antwort des Regelkreises auf den Einheitssprung in der Simulation ist nach den o.g. Gesichtspunkten zufriedenstellend ausgefallen. Die Höhe der Sollwertsprünge wird nun in kleinen Schritten erhöht. Es zeigt sich, dass der Kreis mit PD-Regler ab einer Sprunghöhe von $\pm 2,0 \text{ mm}$, mit PID-Regler ab $\pm 2,4 \text{ mm}$ instabiles Verhalten aufweist. Die Stellgröße i_{steuer} geht in den Anschlag und der Istwert x gegen Unendlich.

Den stabilen Bereich der Sollwertsprünge kann man durch Experimente mit Variationen der Werte von P, I und D etwas erweitern. Im Falle des PD-Reglers ist für $P = 9000$ und $D = 300$ eine maximale Sprunghöhe von $\pm 2,6 \text{ mm}$ erreichbar. Der PID-Regler schafft diese Sprünge ebenfalls mit $P = 12000$, $D = 5000$ und $D = 1000$. Beim letzterem Reglertyp bewirkt der vergrößerte Integralanteil außerdem eine geringere Regelabweichung.

Weitere Erhöhungen der Parameterwerte bringen keine wesentliche Verbesserung mehr und sind auch in Bezug auf die Begrenzungen der realen Regelstrecke, wie sich im Echtzeitexperiment zeigt, nicht sinnvoll. Für die folgenden Simulationen wurden gute Voraussetzungen geschaffen.

6.4. Echtzeit-Simulation

Um dem Echtzeitexperiment mit realem System näher zu kommen, wird die Regelung des geschlossenen Kreises mit Real-Time Workshop in Echtzeit simuliert. Der generierte, kompilierte und gepackte C-Code läuft dadurch auf dem mittels Real-Time Windows Target bereitgestellten Echtzeitkernel, kann aber zur Laufzeit noch Daten mit dem zugrunde liegenden Simulink-Modell austauschen. Das ermöglicht sowohl die Beeinflussung von Variablen als auch die Visualisierung der Signalverläufe.

Die Verwendung einer Kombination aus *Constant*- und *Slider Gain*-Block im erweiterten Modell `loop_sim_rt` gestaltet die Sollwertvorgabe etwas komfortabler. Dadurch ist der Sollwert zur Laufzeit mit einem Schieberegler im Bereich $-5 \dots +5 \text{ mm}$ variierbar. Im *Main*-Tab von *Constant* wird $1/2000$ als *Sample time* eingegeben, was dem Sample-Intervall der realen Abtastung mit 2 kHz entspricht ($T = \frac{1}{f}$) und von den folgenden Blöcken im Modell übernommen wird. Das Simulationensende ist auf inf^2 gesetzt. So beendet erst das Drücken des Buttons *Stop real-time code* die Ausführung der Simulation. Der *Simulation mode External* bewirkt die Ausführung auf externer, in diesem Fall virtueller, Hardware mit Real-Time Workshop. Um jeweils 10 Sekunden der Signalgraphen anzeigen zu können, ist ein entsprechend großer Datenpuffer notwendig. Über die Menüleiste *Tools* \rightarrow *External Mode Control Panel...* \rightarrow *Signal & Triggering* muss dazu bei 2 kHz Abtastrate ein *Duration*-Wert von 20000 gewählt werden.

²infinity: positiv Unendlich

Exemplarisch sind in Abbildung 6.4 10 Sekunden eines Durchlaufs bei diesen Einstellungen mit PID-Regler und den Parametern aus Abschnitt 6.3 dargestellt.

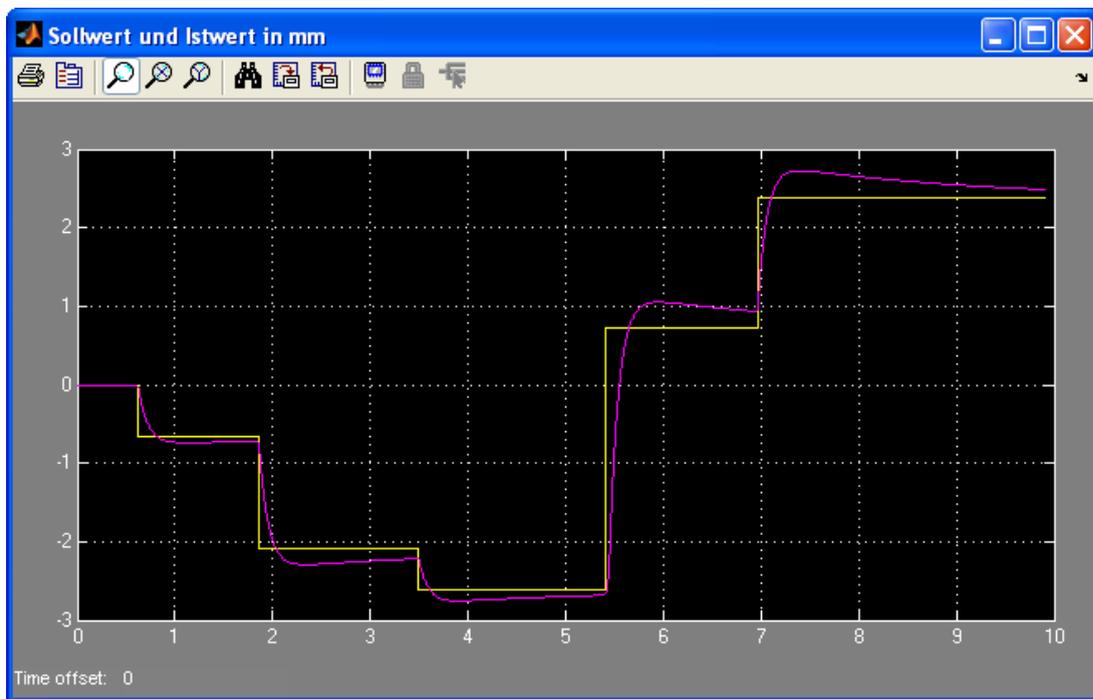


Abbildung 6.4: Echtzeitsimulation des Kreises mit PID-Regler

Man sieht, dass auch betragsmäßig größere Sprünge als 2,6 mm vom negativem in den positiven Bereich möglich sind. Bei Erhöhung in kleinen Schritten kann der Istwert sogar Sollwerten über 3 mm noch folgen, was weitere Versuchsreihen gezeigt haben. Die tatsächliche Qualität der Simulationen wird im folgenden Kapitel bei Untersuchungen des geschlossenen Regelkreises mit der realen Strecke analysiert.

7. Echtzeituntersuchung des geschlossenen Regelkreises mit realer Strecke

Die vorangegangene Simulation kommt dem Verhalten des Regelkreises mit realer Strecke sehr nahe. Allerdings wurden bei der Modellierung auch Streckeneigenschaften wie beispielsweise der geringe Rauschanteil im Positionssignal nicht berücksichtigt. Das unten abgebildeten Modell `loop_real` zeigt eine Erweiterung des Modells aus Abschnitt 6.4.

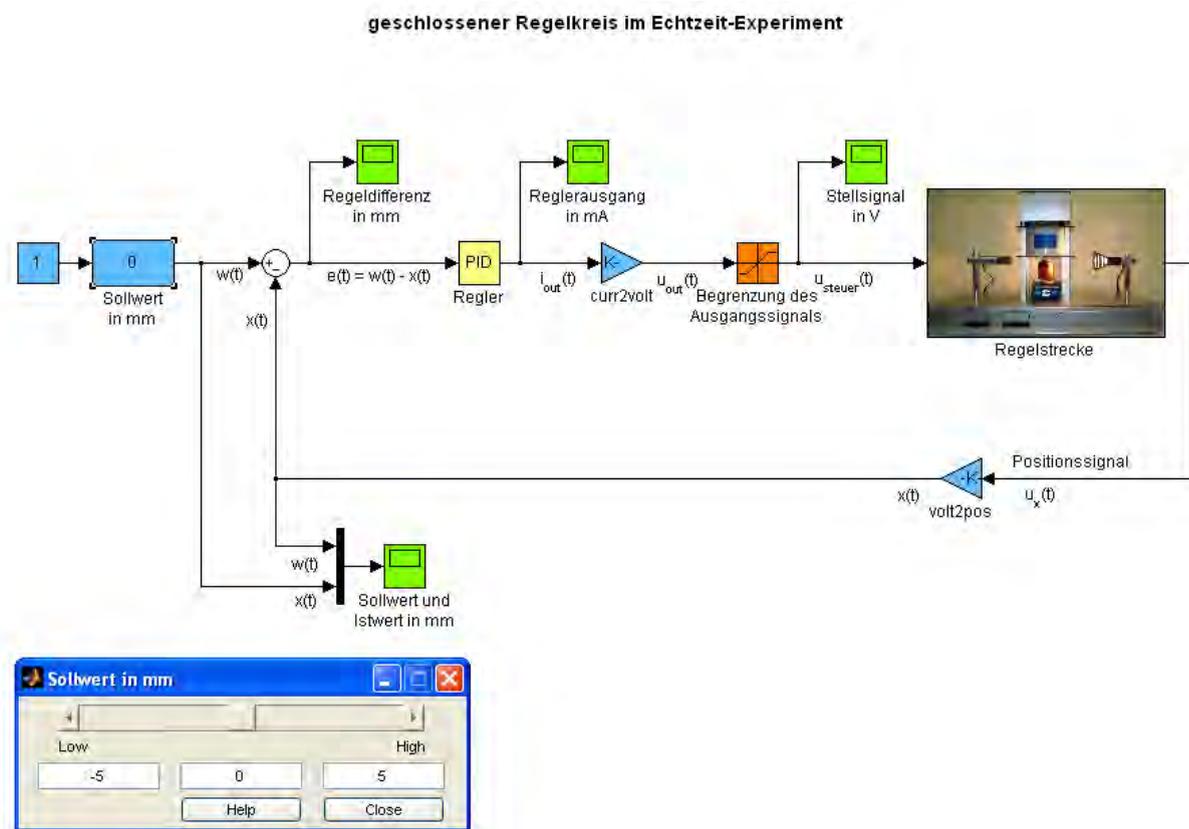


Abbildung 7.1: Blockschaltbild des Simulink-Modells `loop_real` mit geöffnetem *Slider Gain*

Die Sollwertvorgabe erfolgt ebenfalls wieder mit dem in der Darstellung geöffneten Schieberegler. Ein Subsystem mit analogem Ein- und Ausgangskanal zur Datenerfassungskarte ersetzt die vorher mittels Übertragungsfunktion modellierte Strecke. Auf dem Funktionsblock liegt aus Anschaulichkeitsgründen ein Foto des Experimentaufbaus.

Der PID-Reglerblock wurde außerdem so implementiert, dass die Reglerparameter über *Slider Gains* veränderbar sind. Abbildung 7.4 auf Seite 45 zeigt diese im Subsystem verwendeten Schieberegler in ungeöffnetem Zustand in blau. Der Aufbau des PID-Systems entspricht ansonsten exakt der ursprünglichen Funktionalität des Blocks aus der Simulink-Library. Um die Signalverläufe über einen Zeitraum von 20 Sekunden beobachten zu können, wird bei einer Abtastrate von 2 kHz ein *Duration*-Wert von 40000 im *External Mode Control Panel* für *Floating Scope* und *Signal & Triggering* eingestellt. In allen Scope Bausteinen zur Signalvisualisierung ist nach Drücken des Buttons *Parameters* im Reiter *General* die *Time Range* auf 20 gesetzt, was eine Ausgabe der Graphen von jeweils 20 Sekunden bewirkt.

Anmerkung: Bei allen Modellen die Real-Time Workshop verwenden, muss vor Start der Ausführung durch *Strg+B* zuerst der C-Code erstellt, kompiliert und gepackt und dann die Verbindung zum Real-Time Target mit *Strg+T* hergestellt sein. Die Standardkonfiguration für Real-Time Windows Target ist mit dem MATLAB-Kommandozeilenaufruf `rtwinconfigset('modelname')` bereits geladen worden.

7.1. Regelkreis mit PD-Regler

Bei Versuchen mit den unter Punkt 6.3 angegebenen Reglerparametern fällt schnell auf, dass diese Werte große Sprünge im Ausgangssignal $i_{\text{out}}(t)$ des Reglers und damit letzten Endes auch im begrenzten Stellsignal $u_{\text{steuer}}(t)$ über den gesamten Bereich von $-10 \dots +10\text{ V}$ verursachen. Verantwortlich dafür ist der Rauschanteil im Positionssignal $u_x(t)$. Die Rauschamplitude $\lesssim 10\text{ mV}$ wird vom D-Term erheblich verstärkt. Eine Regelung der Strecke ist zwar möglich, die erzeugte Rauschleistung und damit die ineffektive Energieumsetzung u.a. durch die hohen Wärmeverlustenergie der Regelstrecke ist jedoch unerwünscht. Erzielt soll eine Regelung werden, deren Energieverbrauch möglichst gering ist.

Bei Variationen der Parameter P und D stellt sich schließlich heraus, dass die unter Punkt 6.3 ermittelten Werte zu hoch angesetzt wurden. Nach iterativer Optimierung der Reglerparameter auf Energieeffizienz zeigt der Regelkreis mit

$$P_{\text{PD}} = 1000 \quad (7.1)$$

und $D_{\text{PD}} = 50$

das in den Abbildungen 7.2 und 7.3 gezeigte Verhalten. Man erkennt zum einen, dass das Stellsignal $u_{\text{steuer}}(t)$ nur eine Rauschamplitude $\lesssim 1\text{ V}$ (anstatt 10 V) aufweist. Zum anderen lässt sich die Strecke in einem weit größeren Bereich regeln, als die Simulationsergebnisse vermuten ließen. Sprünge von 10 mm bei stabilem Verhalten des Kreises sind problemlos möglich. Auch die bleibende Regelanweichung ist wesentlich geringer, als beim simulierten, in Abbildung 6.2 dargestellten Verhalten.

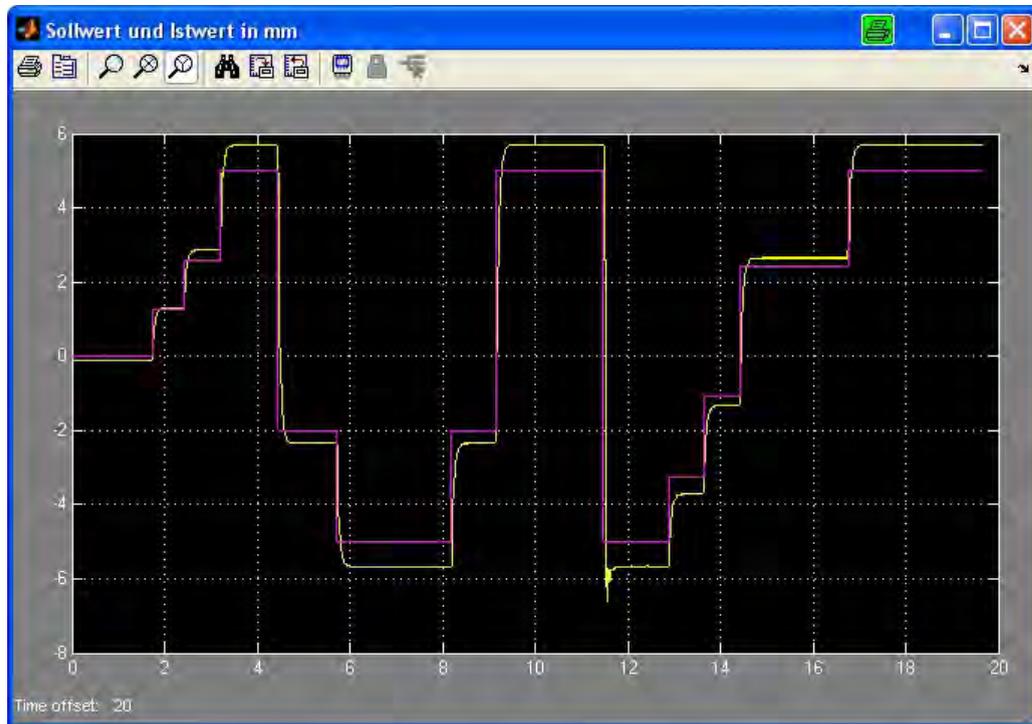


Abbildung 7.2: Sollwert und Position des Regelkreises mit PD-Regler



Abbildung 7.3: Stellsignal des Regelkreises mit PD-Regler

7.2. Regelkreis mit PID-Regler

Auch bei einem Regelkreis mit PID-Regler und den Parameterwerten aus Abschnitt 6.3 ergibt sich zuerst das selbe Resultat wie oben beim PD-Regler beschrieben. Abbildung 7.4 zeigt die ermittelten Reglerwerte nach der iterativen Optimierung im geöffneten Subsystem PID.

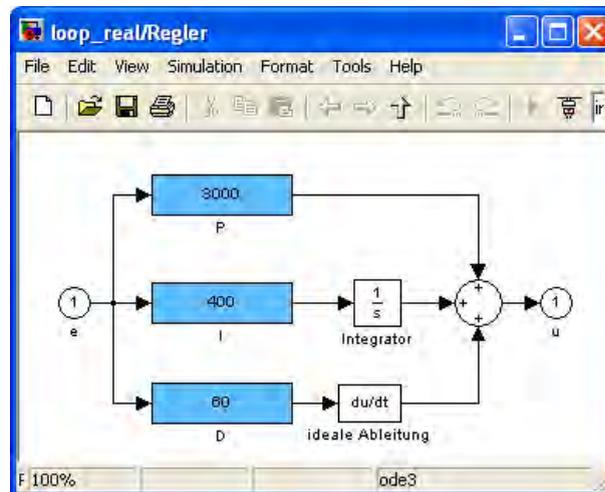


Abbildung 7.4: Subsystem PID mit optimierten Parameterwerten

Mit

$$\begin{aligned}
 P_{\text{PID}} &= 3000, \\
 I_{\text{PID}} &= 400 \\
 \text{und } D_{\text{PID}} &= 60
 \end{aligned}
 \tag{7.2}$$

kann man das in den Abbildungen 7.5 und 7.6 dargestellte Verhalten des Kreises beobachten. Auch hier sieht man, dass wesentlich größere Sprünge des Körpers möglich sind, als die Simulation vermuten lies. Der Rauschanteil im Stellsignal hat eine akzeptable Maximalamplitude $\lesssim 1\text{ V}$, womit der Energieverbrauch wieder verringert werden konnte.

Weitere Versuche zeigen sogar noch stabiles Verhalten bei Sollwertvorgaben von $-9 \dots +9\text{ mm}$.

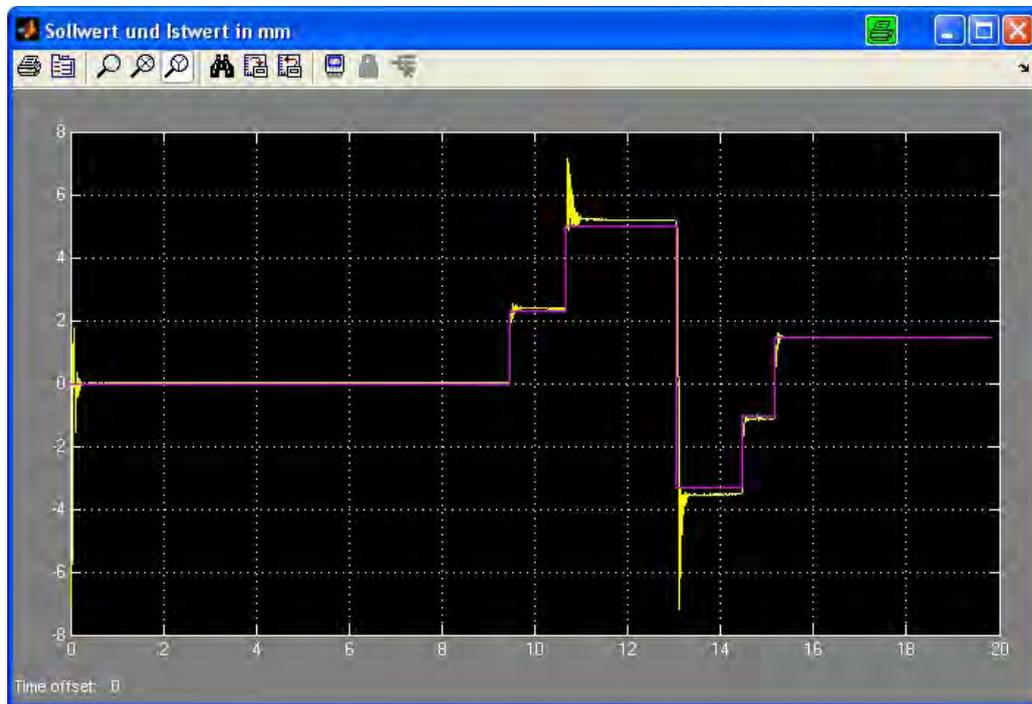


Abbildung 7.5: Sollwert und Position des Regelkreises mit PID-Regler

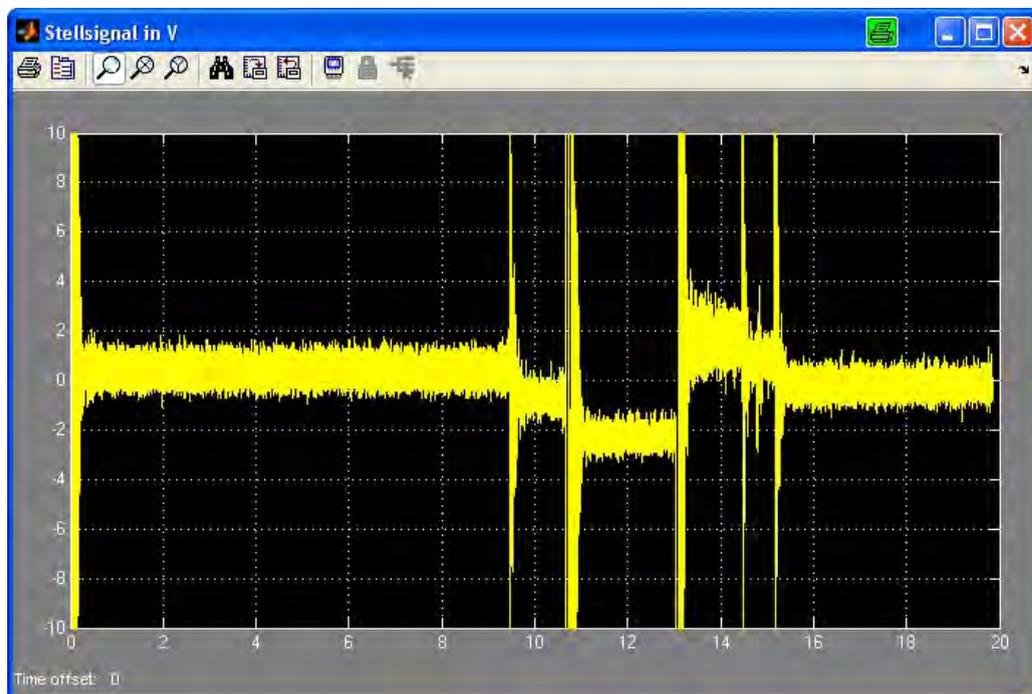


Abbildung 7.6: Stellsignal des Regelkreises mit PID-Regler

7.3. Resumee

Die Ergebnisse aus der Simulation des geschlossenen Kreises und dem Echtzeitexperiment weichen voneinander ab. Die letztendlich für das System mit realer Strecke ermittelten Reglerparameter liegen in der Größenordnung der beim Reglerentwurf in Kapitel 5 ermittelten Werte. Eine Erweiterung des regelbaren Bereichs durch Erhöhung von P, I und D, wie sie im Abschnitt 6.3 beschrieben ist, erweist sich als überflüssig. Dermaßen hohe Reglerparameter resultieren in einem erheblichen Energieverbrauch und sind somit in der Praxis ungeeignet. Macht man den Crosscheck und geht mit den in diesem Kapitel bestimmten Werten noch einmal in die Simulation, wird der Regelkreis wieder nach durchschnittlich 2,4 mm Sollwertsprunghöhe instabil. Die iterative Parameteroptimierung bei Echtzeituntersuchungen mit dem Experimentaufbau scheinen unvermeidlich.

Die Glättung des verrauschten Positionssignals $u_x(t)$ durch ein Tiefpassfilter mit einer Eckfrequenz bei der halben Abtastrate (wie in [3] angewendet) führte nicht zu einer Besserung des Verhaltens. Der Regelkreis zeigte dadurch eine sehr hohe Schwingneigung. Auf eine Tiefpassfilterung oder andere Methoden zur Glättung des Signals wurde verzichtet.

Abschließend ist festzustellen, dass die in dieser Diplomarbeit verwendeten Analyse-, Entwurfs- und Simulationsmethoden eine Regelung der Strecke "Magnetischer Schwebekörper" ermöglichen, die der ursprünglichen Praktikumsversuch in vielen Punkten überlegen ist (vgl. dazu Abbildungen A.14 bis A.17 im Anhang A.4.5 ab Seite 82). Der Energieverbrauch konnte, genauso wie das Überschwingen mit PID-Regelung, verringert werden. Beim PD-Regler ist die bleibende Regelabweichung sehr klein. Der Sollwert wird schnell erreicht und der regelbare Bereich konnte vergrößert werden.

Der Vollständigkeit halber seien hier noch die Ergebnisse angegeben, die aus einem Reglerentwurf, anschließender Simulation und dem Echtzeitexperiment mit den Streckenparametern aus der klassischen Streckenanalyse entstehen. Diese sind

$$\begin{aligned} P_{PD} &= 700 \\ \text{und } D_{PD} &= 30 \end{aligned} \tag{7.3}$$

für den Kreis mit PD-Regler,

$$\begin{aligned} P_{PID} &= 7750, \\ I_{PID} &= 700 \\ \text{und } D_{PID} &= 520 \end{aligned} \tag{7.4}$$

für PID-Regelung. Auch mit den hier angegebenen Werten lässt sich die Strecke gut regeln.

7.4. Sollwertvorgabe durch externe Hardware

Für den überarbeiteten Versuchsaufbau wurde eine zusätzliche Schnittstelle geschaffen, an der ein analoger Ausgang und ein analoger Eingang der Datenerfassungskarte über einen 4-poligen Gerätestecker zugänglich sind. So können Aktoren mit einem Ein- und einem Ausgangssignal im Bereich von jeweils $-10 \dots +10 \text{ V}$ verwendet werden, um den Sollwert der Position des magnetischen Körpers vorzugeben.

Für diese Diplomarbeit bzw. den dazugehörigen Praktikumsversuch findet dabei ein als Spannungsteiler geschaltetes Schiebepotentiometer Verwendung. Vorgesehen ist eine Versorgungsspannung von $+10 \text{ V}$, die mit dem Potentiometer auf $+10 \dots 0 \text{ V}$ heruntergeteilt werden kann. Eine Leuchtdiode signalisiert die anliegende Versorgungsspannung.

Im Simulink-Modell `loop_realHWcont` dient ein *Analog Output*-Baustein der Ausgabe von $+10 \text{ V}$ bei AO 1 der PCI-6014, was dem Ausgangskanal 2 im Block entspricht. Die geteilte Spannung wird mit *Analog Input* von AI 2 (Eingangskanal 3) eingelesen. 1 V entspricht dabei 1 mm Sollwert. Um einen Regelbereich von $-5 \dots +5 \text{ mm}$ zu erhalten, ist die Addition eines Offsets von -5 notwendig.

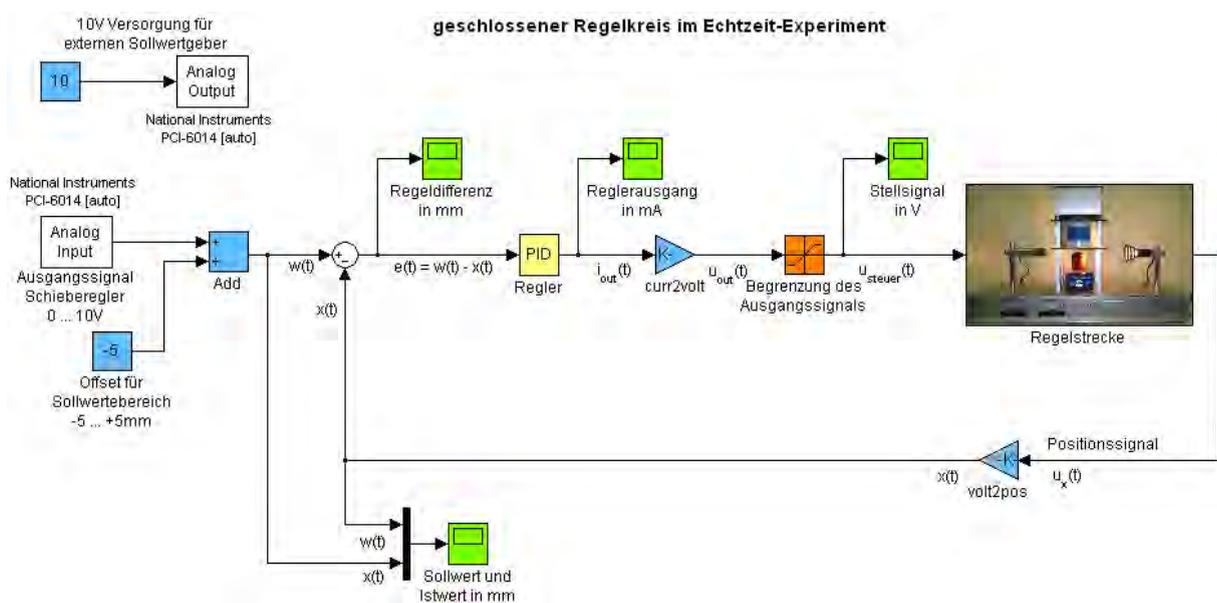


Abbildung 7.7: Simulink-Modell `loop_realHWcont` zur Einbindung des externen Sollwertgebers

Die Versuchsergebnisse unterscheiden sich im Prinzip nicht von den in den Abschnitten 2.1 und 7.2 abgebildeten. Vom externen Sollwertgeber wird allerdings ein Sollwertsignal mit 2 kHz eingelesen und verarbeitet, beim *Slider Gain* übernimmt Simulink die Werte erst nach Loslassen der Maustaste. So kann man mit dem Hardware-Schiebepotentiometer im Wertebereich etwas sanfter und kontinuierlicher regeln.

8. Zusammenfassung und Ausblick

Der Praktikumsversuch "Magnetischer Schwebekörper" konnte ohne größere Schwierigkeiten auf den aktuellen Stand der Technik gebracht und so in eine neue Dekade hinübergerettet werden. Der Versuchsaufbau macht Regelungstechnik "begreifbarer", da in die Regelstrecke tatsächlich eingegriffen werden kann und diese ein im Gegensatz zu beispielsweise rein elektronischen Strecken ein wirklich beobachtbares Verhalten zeigt.

Die Praktikumsanleitung deckt Lernziele wie den grundsätzlichen Umgang mit MATLAB/Simulink, mathematische Modellierung physikalischer Regelstrecken, softwaregestützte Streckenanalyse, Reglerentwurf und Simulation von Systemen, sowie Signalausgabe über Hardware und Verarbeitung eingelesener Signale ab. Die mathematischen Inhalte umfassen unter anderem die Laplace-Transformation, Zustandsraumdarstellung, Konvertierung zwischen Reglerübertragungsfunktionen in additiver Form und Produktform und Stabilitätsbetrachtungen im s-Raum anhand des charakteristischen Polynoms nach Hurwitz. Durch den modularen Aufbau der Anleitung kann diese sowohl auf die Lerninhalte der jeweiligen Studiengänge, als auch auf den gewünschten zeitlichen Umfang des Praktikumsversuchs angepasst werden.

Das Ergebnis bei der Streckenanalyse mittels System Identification Toolbox übertrifft die Resultate des ursprünglichen Versuchs. Auch der Reglerentwurf mit der Control System Toolbox ist wegen seiner Anschaulichkeit ein großer Gewinn für den Versuchsablauf. Die parallele und sich aktualisierende Darstellung des Regelkreisverhaltens im Zeit- und Frequenzbereich fördert das Verständnis des Zusammenhangs von Antwortverhalten des System und der Lage von Polen und Nullstellen in der Übertragungsfunktion. Die Beurteilung der Simulation und des Echtzeitexperiments wurden in den Kapiteln 6 und 7 bereits vorweggenommen.

Es stellt sich heraus, dass die genäherte und linearisierte mathematische Streckenbeschreibung für den Entwurf einer effektiven und energieverbrauchssarmen Regelung ausreichende Genauigkeit liefert. Eine detaillierte Analyse des Systemverhaltens, insbesondere das Hinzunehmen einer Beschreibung des dynamischen Verhaltens der Ströme und Magnetfelder mit den maxwellschen Gleichungen, stellt eine interessante Herausforderung dar.

Da im Moment nur ein Versuchsaufbau an der Hochschule Rosenheim existiert, der Praktikumsversuch aber von zwei Studiengängen durchgeführt wird, wäre die Anschaffung eines zweiten Aufbaus, auch wegen der Belastung der Hardware, sinnvoll. Die Technische Universität München verfügt noch über mindestens eine, nicht im Betrieb befindlichen Anordnung. Es besteht die Hoffnung, dass der Hochschule Rosenheim ein zusätzlicher Aufbau zur Verfügung gestellt wird.

Was die Modernisierung des Praktikumsversuchs betrifft, ist die Erweiterung um eine Fuzzy-Logic-Regelung denkbar. The Mathworks bietet auch hierzu eine passende Toolbox zur Erleichterung dieser Aufgabe an. Im Labor für Mess- und Regelungstechnik ist die Toolbox noch nicht vorhanden. Die Umsetzung des Themengebiets bleibt einer zukünftigen Arbeit vorbehalten.

Literaturverzeichnis

- [1] DÖRRSCHEIDT, F.; LATZEL, W.: *Grundlagen der Regelungstechnik*. 7. Auflage, Harry Deutsch, Frankfurt am Main 2007
- [2] GRUPP, F.; GRUPP, F.: *Simulink - Grundlagen und Beispiele*. Oldenbourg, München 2007
- [3] KOTTER, H.: *Identifikation einer unbekanntten Regelstrecke und Reglersynthese mittels Matlab*. Diplomarbeit, Hochschule Rosenheim 2008
- [4] LUTZ, H.; WENDT W.: *Taschenbuch der Regelungstechnik*. Diplomarbeit, Hochschule Rosenheim 2008
- [5] LJUNG, L.: *System Identification Toolbox™ 7 - Getting Started Guide*. The MathWorks, Inc., Third printing, March 2008
- [6] SCHMIDT, G.: *Lehr- und Experimentiermodell Magnetischer Schwebekörper (Handbuch)*. Lehrstuhl für Steuerungs- und Regelungstechnik der Technischen Universität München, Ausgabe 1/93
- [7] SCHITTENHELM, W.: *Unterlagen zur Vorlesung Regelungstechnik I für EIT5*. Hochschule Rosenheim 2007
- [8] SCHNEIDER, E.: *Praktikumsanleitung zum Fach Regelungstechnik, Versuch Magnetschwebekörper*. Hochschule Rosenheim 2007
- [9] THE MATHWORKS: *Control System Toolbox® 8 - Getting Started Guide*. The MathWorks, Inc., Third printing, March 2008
- [10] THE MATHWORKS: *Real-Time Workshop™ 7 - Getting Started Guide*. The MathWorks, Inc., Sixth printing, March 2008
- [11] <http://www.mathworks.com/access/helpdesk/help/toolbox/ident/>
(The MathWorks US/Canada, Dokumentation der System Identification Toolbox)
- [12] <http://www.dudkowski.de/stecker/seriel25.htm>
(Homepage von Eduard Dubkowsky, Sammlung von Dokumenten zur Steckerbelegungen)
- [13] <http://www.mathworks.de/products/control/>
(The MathWorks Deutschland, Produktbeschreibung der Control System Toolbox)

- [14] <http://www.mathworks.de/products/daq/>
(The MathWorks Deutschland, Produktbeschreibung der Data Acquisition Toolbox)
- [15] <http://www.mathworks.de/products/rtw/>
(The MathWorks Deutschland, Produktbeschreibung des Real-Time Workshop)
- [16] <http://www.mathworks.com/products/rtwt/>
(The MathWorks Deutschland, Produktbeschreibung des Real-Time Workshop)
- [17] <http://www.mathworks.de/products/sysid/>
(The MathWorks Deutschland, Produktbeschreibung der System Identification Toolbox)
- [18] <http://www.ni.com/pdf/manuals/370844b.pdf>
(National Instruments Homepage, Spezifikation der NI 6013/6014-Familie)
- [19] <http://www.wyu.edu.cn/stations/csjsweb/zlxz/ziliao/PCI-6014.pdf>
(NI PCI-6013/-6014 User Manual)
- [20] <http://www.rn-wissen.de/index.php/RS232>
(Roboternetz Website, RS232 Kontaktbelegungen)

Alle aufgeführten Webseiten wurden im Zeitraum von Oktober 2009 bis Januar 2010 aufgerufen.

A. Anhang

A.1. Abbildungsverzeichnis	54
A.2. Pläne und Schaltbilder	56
A.2.1. Leistungsverstärker	56
A.2.2. Messelektronik	58
A.2.3. Schnittstellen	60
A.2.4. Blockschaltplan des neuen Versuchsaufbaus	63
A.2.5. Verdrahtungsplan des Schiebepotentiometers	64
A.3. m-Files	65
A.3.1. Listing: load_conv_param.m	65
A.3.2. Listing: get_static_data.m	65
A.3.3. Listing: plot_static_data.m	69
A.3.4. Listing: prep_step_data.m	70
A.3.5. Listing: plot_step_lin.m	71
A.3.6. Listing: plot_step_log.m	72
A.3.7. Listing: get_state_param.m	73
A.3.8. Listing: comp_id_re.m	74
A.3.9. Listing: reset_PCI6014.m	77
A.4. Kennlinien	78
A.4.1. Statische Kennlinie der Regelstrecke	78
A.4.2. Aufbereitete Sprungantwort der Regelstrecke	79
A.4.3. Näherungsgerade zur Auswertung der halblogarithmisch aufgetragenen Sprungantwort	80
A.4.4. Vergleich von realer Sprungantwort mit idealen Verläufen aus klassischer Streckenanalyse und Grey-Box-Modellierung	81
A.4.5. Ausdrücke der Ergebnisse des ursprünglichen Praktikumsversuchs	82
Praktikumsanleitung	86

A.1. Abbildungsverzeichnis

Abb. 1.1. Experimentaufbau, Frontansicht	2
Abb. 1.2. PC-Experimentsteuerung, Menü	3
Abb. 3.1. Experimentaufbau, Funktionsplan	11
Abb. 4.1. Kräfte am magnetischen Körper	14
Abb. 4.2. graphische Darstellung der Messpunkte zur Ermittlung der statischen Kennlinie	18
Abb. 4.3. Blockschaltbild des Simulink-Modells <code>get_step_data</code>	20
Abb. 4.4. Rohdaten der Sprungantwort	21
Abb. 4.5. Halblogarithmische Auftragung der Sprungantwort	22
Abb. 4.6. Vergleich der Sprungantwort aus theoretischem Modell mit realen Daten bei klassischer Streckenanalyse	23
Abb. 4.7. Vergleich der Sprungantwort aus theoretischem Modell mit realen Daten bei Grey-Box-Modellierung	27
Abb. 5.1. Regelkreisstruktur	29
Abb. 5.2. Einheitssprungantwort des Regelkreises mit PD-Regler im <i>LTI Viewer</i>	31
Abb. 5.3. Übertragungsfunktion des PD-Reglers im <i>Compensator Editor</i>	32
Abb. 5.4. Einheitssprungantwort des Regelkreises mit PID-Regler im <i>LTI Viewer</i>	33
Abb. 5.5. Übertragungsfunktion des PID-Reglers im <i>Compensator Editor</i>	34
Abb. 6.1. Blockschaltbild des Simulink-Modells <code>loop_sim</code>	35
Abb. 6.2. Simulierte Sprungantwort des Kreises mit PD-Regler	37
Abb. 6.3. Simulierte Sprungantwort des Kreises mit PID-Regler	38
Abb. 6.4. Echtzeitsimulation des Kreises mit PID-Regler	41
Abb. 7.1. Blockschaltbild des Simulink-Modells <code>loop_real</code> mit geöffnetem <i>Slider Gain</i>	42
Abb. 7.2. Sollwert und Position des Regelkreises mit PD-Regler	44
Abb. 7.3. Stellsignal des Regelkreises mit PD-Regler	44
Abb. 7.4. Subsystem PID mit optimierten Parameterwerten	45
Abb. 7.5. Sollwert und Position des Regelkreises mit PID-Regler	46
Abb. 7.6. Stellsignal des Regelkreises mit PID-Regler	46
Abb. 7.7. Simulink-Modell <code>loop_realHWcont</code> zur Einbindung des externen Sollwertgebers	48
Abb. A.1. Leistungsverstärker, Prinzipschaltbild	56
Abb. A.2. Leistungsverstärker, elektrischer Aufbau	57
Abb. A.3. Messelektronik, Prinzipschaltbild	58
Abb. A.4. Messelektronik, elektrischer Aufbau	59
Abb. A.5. Experimentaufbau, Rückansicht	60
Abb. A.6. Ausgangspinbelegung NI PCI-6014	61
Abb. A.7. Verkabelung der Schnittstellen	62

Abb. A.8. Blockschaltplan des neuen Versuchsaufbaus	63
Abb. A.9. Verdrahtungsplan des Schiebepotentiometers	64
Abb. A.10. Statische Kennlinie der Regelstrecke	78
Abb. A.11. Aufbereitete Sprungantwort der Regelstrecke	79
Abb. A.12. Näherungsgerade zur Auswertung der halblogarithmisch aufgetragenen Sprungantwort	80
Abb. A.13. Vergleich der idealen Sprungantworten mit dem realen Verlauf . . .	81
Abb. A.14. Simulation des geschlossenen Regelkreises mit PD-Regler (ursprünglicher Versuch)	82
Abb. A.15. Simulation des geschlossenen Regelkreises mit PID-Regler (ursprünglicher Versuch)	83
Abb. A.16. Echtzeitexperiment mit PD-Regler (ursprünglicher Versuch)	84
Abb. A.17. Echtzeitexperiment mit PID-Regler (ursprünglicher Versuch)	85

A.2. Pläne und Schaltbilder

A.2.1. Leistungsverstärker

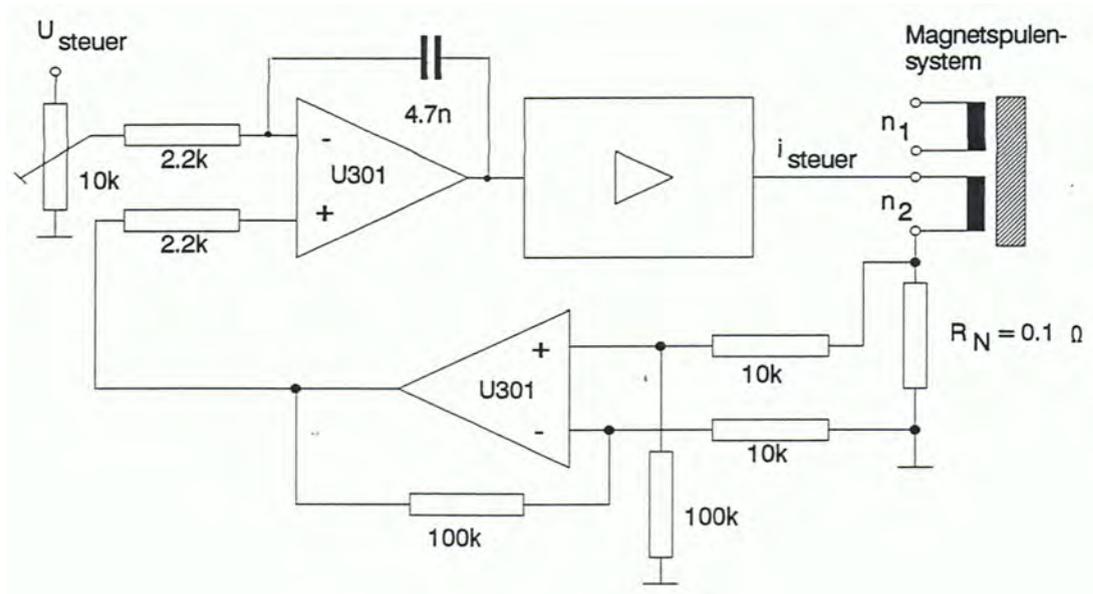


Abbildung A.1: Leistungsverstärker, Prinzipschaltbild (Quelle: [6])

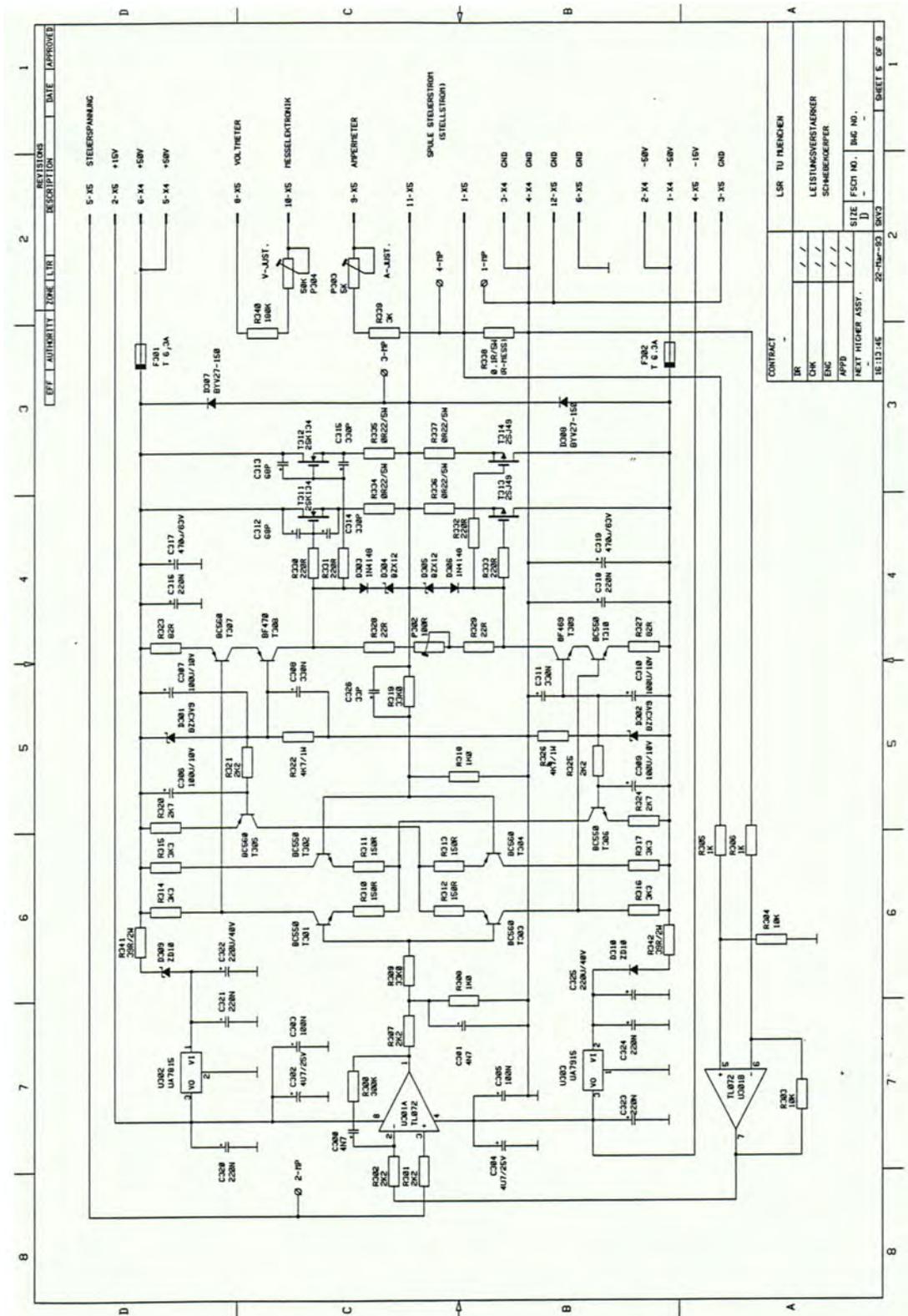


Abbildung A.2: Leistungsverstärker, elektrischer Aufbau (Quelle: [6])

A.2.2. Messelektronik

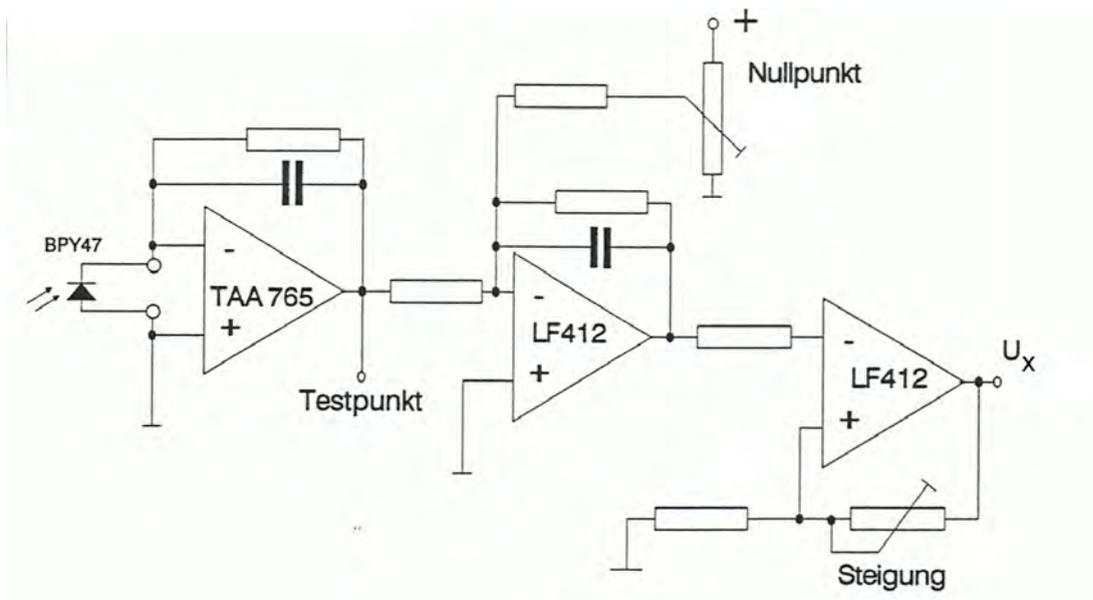
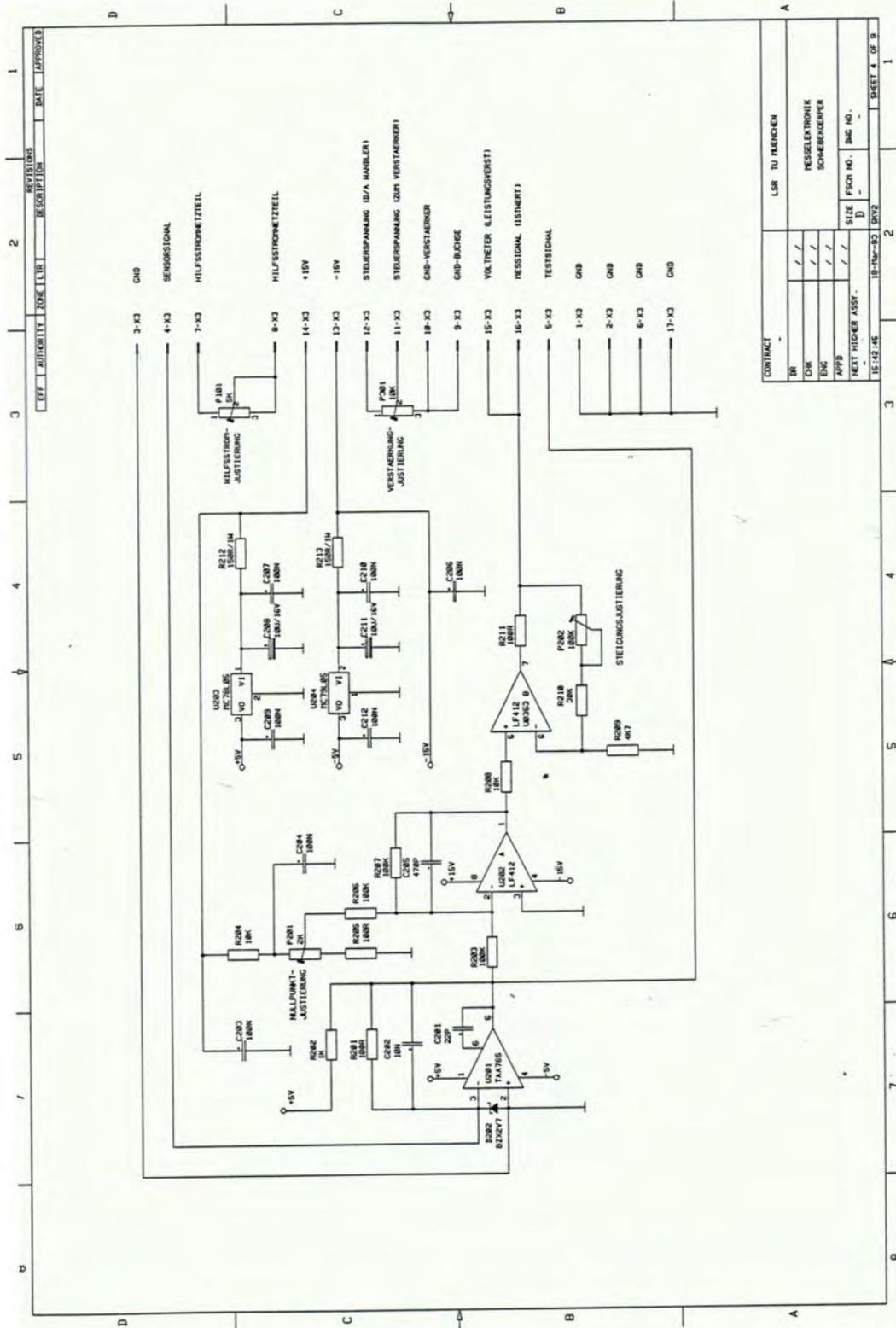


Abbildung A.3: Messelektronik, Prinzipschaltbild (Quelle: [6])



REV. NO.		DATE	APPROVED
EFF.	AUTHORITY	ZONE	LUR
DESCRIPTION			
LUR TU MÜNCHEN			
BR	/ /	MESSELEKTRONIK	
CHK	/ /	SCHREIBGERÄTE	
ENG	/ /	SCHREIBGERÄTE	
APP	/ /	SCHREIBGERÄTE	
NEXT HIGHER ASST.		SIZE	FOOT NO.
15-12-145		10-20-83	8062
SHEET 4 OF 9			

Abbildung A.4: Messelektronik, elektrischer Aufbau (Quelle: [6])

A.2.3. Schnittstellen

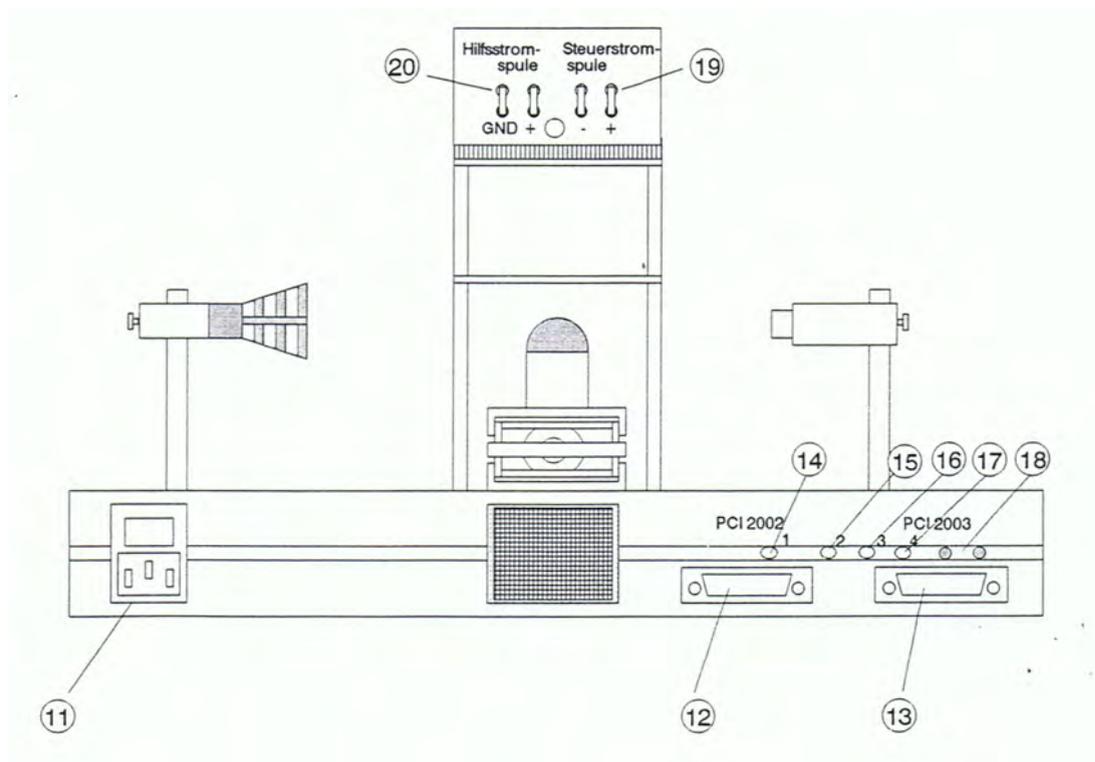


Abbildung A.5: Experimentaufbau, Rückansicht (Quelle: [6])

- ⑪ Netzanschluss mit Schalter
- ⑫ Anschluss an PC-Experimentsteuerung, Messsignal
- ⑬ Anschluss an PC-Experimentsteuerung, Stellsignal
- ⑭ Eingangsspannungsteiler des Leistungsverstärkers
- ⑮ Hilfsstromeinstellung
- ⑯ Nullpunktjustierung der Sensorelektronik
- ⑰ Steigungsjustierung der Sensorelektronik
- ⑱ Testpunktbuchsen für Sensorelektronik
- ⑲ Steckbrücken für Steuerstrom
- ⑳ Steckbrücken für Hilfsstrom

AI 8	34	68	AI 0
AI 1	33	67	AI GND
AI GND	32	66	AI 9
AI 10	31	65	AI 2
AI 3	30	64	AI GND
AI GND	29	63	AI 11
AI 4	28	62	AI SENSE
AI GND	27	61	AI 12
AI 13	26	60	AI 5
AI 6	25	59	AI GND
AI GND	24	58	AI 14
AI 15	23	57	AI 7
AO 0	22	56	AI GND
AO 1	21	55	AO GND
NC	20	54	AO GND
P0.4	19	53	D GND
D GND	18	52	P0.0
P0.1	17	51	P0.5
P0.6	16	50	D GND
D GND	15	49	P0.2
+5 V	14	48	P0.7
D GND	13	47	P0.3
D GND	12	46	AI HOLD COMP
PFI 0/AI START TRIG	11	45	EXT STROBE
PFI 1/AI REF TRIG	10	44	D GND
D GND	9	43	PFI 2/AI CONV CLK
+5 V	8	42	PFI 3/CTR 1 SRC
D GND	7	41	PFI 4/CTR 1 GATE
PFI 5/AO SAMP CLK	6	40	CTR 1 OUT
PFI 6/AO START TRIG	5	39	D GND
D GND	4	38	PFI 7/AI SAMP CLK
PFI 9/CTR 0 GATE	3	37	PFI 8/CTR 0 SRC
CTR 0 OUT	2	36	D GND
FREQ OUT	1	35	D GND

NC = No Connect

Abbildung A.6: Ausgangspinbelegung NI PCI-6014 (Quelle: [18])

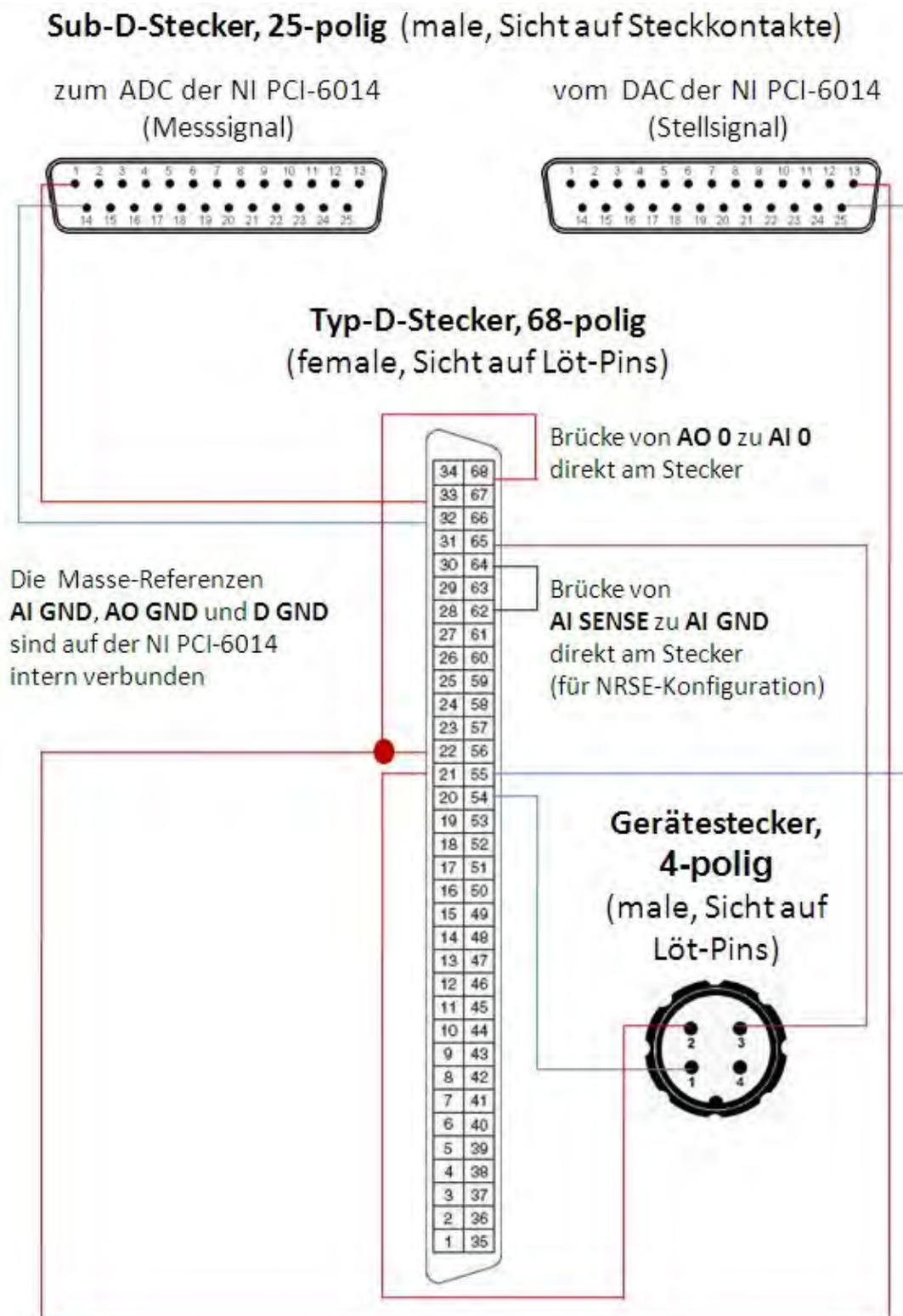


Abbildung A.7: Verkabelung der Schnittstellen (Quelle: eigene Darstellung mit [18] und [20])

A.2.4. Blockschaltplan des neuen Versuchsaufbaus

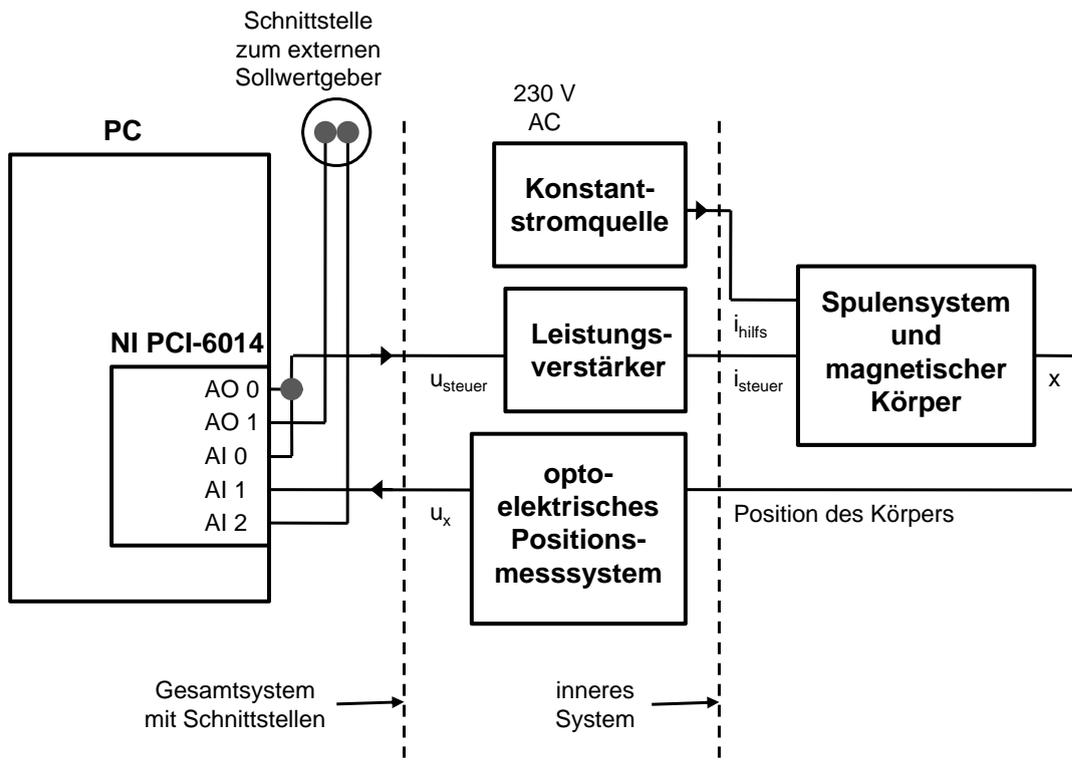


Abbildung A.8: Blockschaltplan des Versuchsaufbaus (Quelle: eigene Darstellung)

A.2.5. Verdrahtungsplan des Schiebepotentiometers

4-poliger Gerätestecker (female)

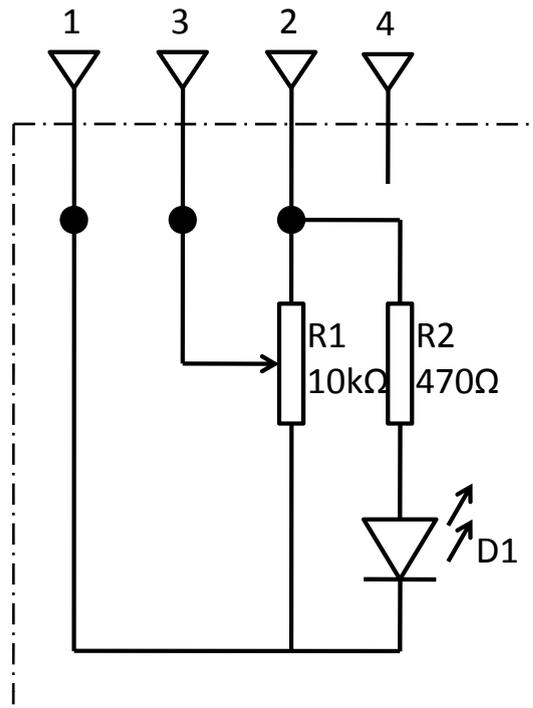


Abbildung A.9: Verdrahtungsplan des Schiebepotentiometers (Quelle: eigene Darstellung)

A.3. m-Files

A.3.1. Listing: load_conv_param.m

```

1 %LOAD_CONV_PARAM lädt die Parameter für die Konvertierung der
2 %       Spannungswerte an den analogen Ein- und Ausgängen der
3 %       Datenerfassungskarte in die Positions- bzw. Spulenstromwerte
4 %       des Systemkerns der Regelstrecke "Magnetischer Schwebekörper"
5 %       in den Workspace.
6 %   LOAD_CONV_PARAM
7 %
8 %   Faktoren für die Umrechnung
9 %   VOLT2CURR: Eingangsspannung auf Steuerspulenstrom [mA/V]
10 %   CURR2VOLT: Spulenstrom auf Ausgangsspannung [V/mA] (Kehrwert von VOLT2CURR)
11 %   VOLT2POS: Eingangsspannung auf vertikale Position [mm/V]
12 %
13 %   Die Werte wurden am Experimentaufbau eingestellt und gemessen am 10.11.2009.
14
15
16
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19
20 volt2curr = 283.0;
21 curr2volt = 1/volt2curr;
22 volt2pos = -1.0;

```

A.3.2. Listing: get_static_data.m

```

1 function stat_data = get_static_data(v2c,c2v,v2p)
2
3
4 %GET_STATIC_DATA dient dem sukzessiven, dialoggeführten Erfassen
5 %       von maximal 20 Messpunkten für die Darstellung der statischen
6 %       Kennlinie (beispielsweise mit PLOT_STATIC_DATA) der
7 %       Regelstrecke "Magnetischer Schwebekörper".
8 %       Für jeden zu erfassenden Messpunkt wird eine einer
9 %       Steuerstromrampe entsprechende Spannungsrampe mit
10 %       benutzerdefiniertem Startpunkt ausgegeben. Das Programm ermittelt
11 %       automatisch den Steuerspulenstromwert, bei dem der Körper abhebt
12 %       anhand eines Schwellenwerts des Positionssignals.
13 %       Dieses Wertepaar wird der Rückgabestruktur angefügt.
14 %   STAT_DATA = GET_STATIC_DATA(V2C,C2V,V2P)
15 %
16 %   STAT_DATA: Rückgabestruktur der Messwertpaare
17 %   STAT_DATA.Ist: Zeilenvektor mit Steuerstromwerten [mA]
18 %   STAT_DATA.x: Zeilenvektor mit Positionswerten des Schwebekörpers [mm]
19 %
20 %   V2C: Eingangsspannung auf Spulenstrom [mA/V]

```

```

21 %
22 %   C2V: Spulenstrom auf Ausgangsspannung [V/mA] (Kehrwert von V2C)
23 %
24 %   V2P: Eingangsspannung auf Position des magnetischen Körpers [mm/V]
25
26
27
28 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29 %% Variablendefinitionen und Initialisierungen
30 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
31
32 clc;                                % Command Window leeren
33
34 %% User-I/O
35 Ist = 0;                             % Steuerstromwert in mA
36 Ito = 0;                             % Strom, bei dem Körper abhebt in mA
37 user_entry = '';                    % Für Benutzereingaben
38 user_entry_check = true;            % Flag: Gültigkeit des Wertes von Ist
39 acquired_point = 1;                % Schleifenzähler Messpunktnummer
40 max_acq_points = 20;               % Maximale Messpunktanzahl
41 continue_loop = 'j';               % Aufnahme neuer Messwert?
42 pause on;                          % Wartefunktion aktivieren
43
44 %% Signal- bzw. Daten-I/O
45 rate = 2000;                       % Abtastrate
46 duration = 1;                      % Dauer in Sekunden
47 len = duration*rate;               % Länge in Samples
48 start_ramp = 0;                   % Startpunkt Rampe
49 end_ramp = start_ramp + 500*c2v;    % Enpunkt Rampe (Ist = 500mA)
50 ramp = linspace(start_ramp,end_ramp,len); % Rampe erzeugen
51
52 %% Konfiguration des analogen Ausgangs
53 AO = analogoutput('nidaq','Dev1'); % analoges Output-Objekt erzeugen
54 ao_chan = addchannel(AO, 0);       % dem Objekt einen Kanal zuweisen
55 putsample(AO, 0);                 % Ausgang auf 0V setzen
56
57 %% Konfiguration der analogen Eingänge
58 AI = analoginput('nidaq','Dev1'); % analoges Input-Objekt erzeugen
59 AI.InputType = 'NonReferencedSingleEnded'; % Typ des Bezugspotentials
60 ai_chans = addchannel(AI,0:1);     % Objekt zwei Kanäle (0,1) zuweisen
61 set(AI,'SampleRate', rate);        % Setzen der Abtastrate
62 set(AI,'SamplesPerTrigger', len);  % Sample-Anzahl
63
64
65 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
66 %% Maximal 20 Wertepaare für statische Kennlinie bestimmen
67 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
68
69 while acquired_point <= max_acq_points & continue_loop == 'j'
70     clc;                            % Command Window leeren
71
72     %% abfragen, ob neuer Messpunkt aufgenommen werden soll;
73     % erst nach erstem Schleifendurchlauf und wenn Ist-Wert korrekt
74     if acquired_point > 1 & user_entry_check == true

```

```
75     continue_loop = input(['\nWollen Sie einen weiteren Messpunkt '...
76                          'aufnehmen? j/n [j]\n'], 's');
77     % falls nur mit return bestätigt, gilt "ja"
78     if isempty(continue_loop)
79         continue_loop = 'j';
80     elseif continue_loop == 'j'
81         continue_loop = 'j';
82     % falls "n" eingegeben, Datenaufnahme beenden
83     elseif continue_loop == 'n'
84         break;
85     % bei unzulässiger Eingabe, Schleife neu durchlaufen
86     else
87         % Ausgabe einer Fehlermeldung in rot
88         fprintf([2, 'Bitte nur "j" für ja, "n" für nein eingeben'...
89                 'oder mit Return bestätigen (= "j")\n\n']);
90         pause(2);
91         continue_loop = 'j';
92         continue;
93     end
94 end
95
96 clc; % Command Window leeren
97
98 %% neuen Wert für Spulenstrom abfragen
99 fprintf(['\n\nBitte geben Sie einen Startwert für den Spulenstrom ein'...
100         '\nBereich: -1000 ... 1000 [mA]' ] );
101 % Wert, bei dem Körper abgehoben hat erst nach
102 % erstem Schleifendurchlauf ausgeben
103 if acquired_point > 1
104     fprintf('\n(Körper hat bei der letzten Messung bei %dmA abgehoben)', Ito);
105 end
106 user_entry = input('\n\n neuer Wert: ', 's');
107 user_entry = str2num(user_entry); % String in Integer umwandeln
108
109 %% neuen Spulenstromwert prüfen und aufnehmen, falls im zulässigen Bereich
110 if (-1000) ≤ user_entry & user_entry ≤ 1000
111     user_entry_check = true;
112     Ist = user_entry;
113     % neuen Spulenstrom über Spannung am analogen Ausgang einstellen
114     putsample(AO, Ist*c2v);
115     fprintf('\neingestellter Spulenstrom: %dmA\n', Ist);
116     fprintf(['\nPlatzieren Sie den Körper jetzt so, dass er gerade '...
117             'noch nicht abhebt.\nStarten Sie dann die Messung '...
118             'mit beliebiger Taste...\n' ]]);
119     pause; % auf Benutzereingabe warten
120
121 %% falls unzulässige Eingabe, Schleifendurchlauf erneut starten
122 else
123     % Ausgabe einer Fehlermeldung in rot
124     fprintf(2, 'Unzulässiger Wert!\n');
125     pause(1); % 1s warten
126     user_entry_check = false; % Flag setzen
127     % äußerste while-Schleife neu durchlaufen
128     continue;
```

```
129     end
130
131     %% Signalausgabe starten und analoge Eingänge einlesen
132     % Spannungsrampe berechnen und ausgeben
133     start_ramp = Ist*c2v; % neuer Startpunkt
134     end_ramp = start_ramp + 500*c2v; % neuer Endpunkt (Start + 500mA)
135     ramp = linspace(start_ramp,end_ramp,len)'; % Rampe erzeugen
136     start(AI);
137     pause(0.1); % 100 ms warten
138     putdata(AO, ramp); % Rampe in die Queue stellen
139     start(AO);
140     [data_in, time] = getdata(AI); % analoge Eingänge einlesen
141
142     %% Punkt im eingelesenen Array suchen, bei dem Eisenkörper abgehoben hat
143     index_data_in = 1; % Suchindex zurücksetzen
144     % maximale Rauschunschärfe des Positionssignals ist kleiner 0.02 Volt
145     while data_in(index_data_in, 2) > ((data_in(3, 2)) - 0.02) &...
146         index_data_in < len
147         index_data_in = index_data_in + 1;
148     end
149
150     %% Punkt Rückgabestruktur anhängen, falls Körper sich korrekt bewegt hat
151     % falls keine Bewegung des Körpers
152     if index_data_in == len
153         % Ausgabe einer Fehlermeldung in rot
154         fprintf(2, '\nHaben Sie nicht etwas vergessen?');
155         % über den analogen Ausgang wieder alten Spulenstromwert setzen
156         putdata(AO, Ist*c2v);
157         pause(3); % 3 Sekunden warten
158         fprintf('\nFortsetzen mit beliebiger Taste...');
159         pause;
160         % äußerste while-Schleife neu durchlaufen
161         continue;
162     else
163         % Punkt in mm und mA umrechnen und der Rückgabestruktur anhängen
164         stat_data.Ist(acquired_point) = data_in(index_data_in, 1);
165         stat_data.Ist(acquired_point) = stat_data.Ist(acquired_point) * v2c;
166         stat_data.x(acquired_point) = data_in(index_data_in, 2);
167         stat_data.x(acquired_point) = stat_data.x(acquired_point) * v2p;
168         % Spulenstromwert neu setzen (Bildschirmausgabe im nächsten Schleifendurchlauf)
169         Ito = int16(stat_data.Ist(acquired_point));
170     end
171
172     clc; % Command Window leeren
173
174     %% Messung beenden, falls maximale Messpunktanzahl erreicht ist
175     if acquired_point == max_acq_points
176         clc; % Command Window leeren
177         fprintf('\n\nMaximale Messpunktanzahl %d aufgenommen.', max_acq_points);
178         break;
179     end
180     % Messpunktnummer erhöhen
181     acquired_point = acquired_point + 1;
182 end
```

```

183
184 pause off; % Wartefunktion deaktivieren
185 fprintf('\nDatenaufnahme beendet.\n\n')
186 putsample(AO, 0); % Ausgang wieder auf 0V setzen
187 % analogen Output freigeben
188 delete(AO);
189 clear AO;
190 % analogen Input freigeben
191 delete(AI);
192 clear AI;

```

A.3.3. Listing: plot_static_data.m

```

1 function plot_static_data(Ist,x)
2
3
4 %PLOT_STATIC_DATA stellt die Punkte der statischen Kennlinie der Regelstrecke
5 %           "Magnetischer Schwebekörper" durch die mittels GET_STATIC_DATA
6 %           gewonnenen Messwertpaaren aus Rückgabestruktur grafisch dar.
7 % PLOT_STATIC_DATA(Ist,X)
8 %
9 % Ist: Zeilenvektor mit Steuerstromwerten [mA]
10 % X: Zeilenvektor mit Positionswerten des Schwebekörpers [mm]
11
12
13
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15 %% Plotten der Punkte als 'x' mit passenden Achsenbeschriftungen und Gitternetz
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18 % neues Anzeigefenster öffnen, mit Programmnamen, ohne Fensternummer
19 figure('Name','plot_static_data','NumberTitle','off');
20 % Verlauf plotten
21 plot(Ist, x, 'x');
22 % Überschrift
23 title(['\fontsize{12}\bf Statische Kennlinie der Regelstrecke ',...
24        '\it magnetischer Schwebekörper'];
25        ['\newline\fontsize{12}\rm Position = f(Steuerspulenstrom)
26        bei ',...
27        'F_{mag} = F_{g}']);
28        % Datum und Uhrzeit ausgeben
29        ['\fontsize{10} aufgenommen am ' datestr(now,'dd.mm.yyyy, HH:MM','local')]);
29 % Achsenbeschriftungen
30 xlabel('Steuerspulenstrom [mA]');
31 ylabel('Position des Körpers [mm]');
32 % Gitternetz darstellen
33 grid on;

```

A.3.4. Listing: prep_step_data.m

```

1 function prep_data = prep_step_data(data,v2c)
2
3
4 %PREP_STEP_DATA extrahiert die für die weitere Verarbeitung
5 %       relevanten Werte aus einem mittels SIMULINK-Modell
6 %       GET_STEP_DATA erzeugten Datensatz der Sprungantwort
7 %       der Regelstrecke "Magnetischer Schwebekörper"
8 %       und bereitet diese auf.
9 % PREP_DATA = PREP_STEP_DATA(DATA,V2C)
10 %
11 % PREP_DATA: Ausgabestruktur mit aufbereiteten Daten
12 % PREP_DATA.t: Spaltenvektor mit Zeitwerten [s]
13 % PREP_DATA.Ist: Spaltenvektor Steuerstromwerten [mA]
14 % PREP_DATA.x: Spaltenvektor mit Positionswerten des Schwebekörpers [mm]
15 %
16 % DATA: Mit SIMULINK-Modell GET_STEP_DATA erzeugter Datensatz
17 %       (Format: Structure with time)
18 %
19 % V2C: Faktor zur Umrechnung von Volt in mA für die
20 %       Darstellung der Steuerstromwerte [mA/V]
21
22
23
24 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26
27 %% Positionsmaximum suchen und dessen Index für Ausgabevektoren setzen,
28 %% um Werte nach dem oberem Anschlag des Körpers abzuschneiden
29 [valuemax, indexmax] = max(data.signals.values(:,2));
30 indexmax = indexmax - 10;
31
32 %% Zeitvektor und Eingangssignal der Ausgabevariablen zuweisen und anpassen
33 % Zeitvektor bis indexmax zuweisen
34 prep_data.t = data.time(1:indexmax);
35 % Steuerstromwerte bis indexmax zuweisen, Volt in mA umrechnen
36 prep_data.Ist = data.signals.values(1:indexmax,1)*v2c;
37 % Minimum der Steuerstromwerte suchen
38 [valuemin, indexmin] = min(prepare_data.Ist);
39 % Steuerstromwerte entsprechend valuemin Richtung 0 verschieben
40 prep_data.Ist = prep_data.Ist - valuemin;
41
42
43 %% Positionswerte der Ausgabevariablen zuweisen und anpassen
44 % Positionswerte bis indexmax zuweisen
45 prep_data.x = data.signals.values(1:indexmax,2)
46 % Minimum der Positionswerte ermitteln
47 [valuemin, indexmin] = min(prepare_data.x);
48 % Positionswerte entsprechend valuemin Richtung 0 verschieben
49 prep_data.x = prep_data.x - valuemin;

```

A.3.5. Listing: plot_step_lin.m

```
1 function plot_step_lin(time,x,stepsize)
2
3
4 %PLOT_STEP_LIN stellt die Sprungantwort der Regelstrecke
5 %      "Magnetischer Schwebekörper" linear graphisch dar
6 % PLOT_STEP_LIN(TIME,X,STEP_SIZE)
7 %
8 % TIME: Spaltenvektor mit Zeitwerten [s]
9 % X: Spaltenvektor mit Positionswerten des Schwebekörpers [mm]
10 % STEP_SIZE: Angabe der Sprunghöhe [mA]
11
12
13
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15 %% Plotten der Sprungantwort mit passenden Achsenbeschriftungen und Gitternetz
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18 % Zeitvektor in Millisekunden Umrechnen
19 time = time * 1000;
20 % neues Anzeigefenster öffnen, mit Programmnamen, ohne Fensternummer
21 figure('Name','plot_step_lin','NumberTitle','off');
22 % Verlauf plotten
23 plot(time, x);
24 % Überschrift
25 title(['\fontsize{12}\bf Sprungantwort der Regelstrecke ',...
26       '\it magnetischer Schwebekörper']);
27 % Sprunghöhe ausgeben
28 ['\newline\fontsize{12}\rm lineare Auftragung, ',...
29  'Sprunghöhe: ', num2str(stepsize) , 'mA'];
30 % Datum und Uhrzeit ausgeben
31 ['\fontsize{10} aufgenommen am ' datestr(now,'dd.mm.yyyy, HH:MM','local')]);
32 % Achsenbeschriftungen
33 xlabel('Zeit [ms]');
34 ylabel('Position des Körpers [mm]');
35 % Gitternetz darstellen
36 grid on;
```

A.3.6. Listing: plot_step_log.m

```

1 function plot_step_log(time,x,stepsize)
2
3
4 %PLOT_STEP_LOG stellt die Sprungantwort der Regelstrecke
5 %           "Magnetischer Schwebekörper" halblogarithmisch graphisch dar
6 % PLOT_STEP_LOG(TIME,X,STEP_SIZE)
7 %
8 % TIME: Spaltenvektor mit Zeitwerten [s]
9 % X: Spaltenvektor mit Positionswerten des Schwebekörpers [mm]
10 % STEP_SIZE: Angabe der Sprunghöhe [mA]
11
12
13
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15 %% Aufbereitung der Daten
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18 % Positionswerte nach oben rücken -> positive Logarithmuswerte
19 x = x + 1.1;
20 % Logarithmieren
21 x = log(x);
22
23
24 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25 %% Plotten der logarithmierten Sprungantwort
26 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
27
28 % Zeitvektor in Millisekunden Umrechnen
29 time = time * 1000;
30 % neues Anzeigefenster öffnen, mit Programmnamen, ohne Fensternummer
31 figure('Name','plot_step_log','NumberTitle','off');
32 % Verlauf plotten
33 plot(time, x);
34 % Überschrift
35 title(['\fontsize{12}\bf Sprungantwort der Regelstrecke ',...
36        '\it magnetischer Schwebekörper'];
37        % Sprunghöhe ausgeben
38        ['\newline\fontsize{12}\rm halblogarithmische Auftragung, ',...
39        'Sprunghöhe: ', num2str(stepsize) , 'mA'];
40        % Datum und Uhrzeit ausgeben
41        ['\fontsize{10} aufgenommen am ' datestr(now,'dd.mm.yyyy, HH:MM','local')]);
42 % Achsenbeschriftungen
43 xlabel('Zeit [ms]');
44 ylabel('ln(Position + Sprunghöhe*k/p) [mm]');
45 % Gitternetz darstellen
46 grid on;

```

A.3.7. Listing: get_state_param.m

```

1 function [A,B,C,D,K,x0] = get_state_param(par,T,aux)
2
3
4 %GET_STATE_PARAM erzeugt die Matrixvariablen der Zustandsraumdarstellung
5 %   der Regelstrecke "Magnetischer Schwebekörper".
6 %   Diese Funktion wird für die Ermittlung eines Grey-Box-Modells
7 %   mit der MATLAB-Funktion IDGREY benötigt (vgl. MATLAB-Hilfe der
8 %   System Identification Toolbox bzw. Hilfe zu IDGREY).
9 %   [A,B,C,D,K,x0] = GET_STATE_PARAM(PAR,T,AUX)
10 %
11 %   [A,B,C,D,K,x0]: Koeffizientenmatrizen der Zustandsraumdarstellung der Form
12 %
13 %           d/dt x(t) = A x(t) + B u(t) + K e(t)
14 %           y(t) = C x(t) + D u(t) + e(t)
15 %           x(0) = x0
16 %
17 %   PAR: Zeilenvektor mit den zu schätzenden Parametern
18 %   PAR(1) = P [mm^2/s^2]
19 %   PAR(2) = K [mm^3/(mA*s^2)]
20 %   Es können entweder die mittels grafischer Auswertung der statischen
21 %   Kennlinie der Regelstrecke "Magnetischer Schwebekörper" ermittelten
22 %   Parameter eingesetzt werden, oder auch beliebige Werte.
23 %   Die Werte dienen als Startpunkt für den Schätzalgorithmus PEM
24 %   (vgl. MATLAB-Hilfe der System Identification Toolbox bzw. Hilfe zu PEM)
25 %
26 %   T: Sample-Intervall, ist für die zeitkontinuierliche Darstellung auf 0 zu
27 %   setzen
28 %   AUX: Hilfsvariablen, werden für die Ermittlung der Zustandsraumvariablen
29 %   der Regelstrecke "Magnetischer Schwebekörper" nicht benötigt, also
30 %   beliebiger Wert
31
32
33
34 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
35 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36
37 A = [0 1;par(1) 0];
38 B = [0;par(2)];
39 C = [1,0];
40 D = 0;
41 K = [0;0];
42 x0 = [0;0];

```

A.3.8. Listing: comp_id_re.m

```

1 function comp_id_re(t,x,p,k,stepsize)
2
3
4 %COMP_ID_RE stellt die experimentell ermittelte Sprungantwort der
5 % Regelstrecke "Magnetischer Schwebekörper" dem theoretischen
6 % Kurvenverlauf
7 %  $x(t) = \text{Sprunghöhe} \cdot (k/p) \cdot 0.5 \cdot (\exp(\sqrt{p}) \cdot t) + \exp(-\sqrt{p}) \cdot t - 2)$ 
8 % gegenüber.
9 % COMP_ID_RE(T,X,P,K,STEP SIZE)
10 %
11 % T: Spaltenvektor mit Zeitwerten Werte [s]
12 % X: Spaltenvektor mit Positionswerten des Schwebekörpers [mm]
13 % P: Zeilenvektor mit experimentell ermittelten Werten für den Parameter p
14 % der Übertragungsfunktion der Regelstrecke [mm^2/s^2]
15 % (P(1) für klassische Ermittlung, P(2) für Grey-Box-Modellierung)
16 % K: Zeilenvektor mit experimentell ermittelten Werten für den Parameter k
17 % der Übertragungsfunktion der Regelstrecke [mm^3/(mA*s^2)]
18 % (K(1) für klassische Ermittlung, K(2) für Grey-Box-Modellierung)
19 % STEPSIZE: Sprunghöhe bei der Durchführung des Experiments [mA]
20
21
22
23 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25
26 %% Untersuchen, ob Parametervektoren gleiche Länge haben
27 if length(p)  $\neq$  length(k)
28 % Ausgabe einer Fehlermeldung in rot
29 fprintf(2,'Parametervektoren müssen die gleiche Länge haben!\n\n');
30 end
31
32 %% Maximum der experimentellen Impulsantwort suchen
33 [xmax, xindexmax] = max(x);
34
35 %% theoretischen Verlauf der Impulsantwort berechnen
36 % falls ein Parametersatz
37 if length(p) == 1
38 step_id.x = stepsize*(k/p) * (0.5*exp(sqrt(p)*t) + 0.5*exp(-sqrt(p)*t) - 1);
39 % falls zwei Parametersätze
40 elseif length(p) == 2
41 step_id1.x=stepsize*(k(1)/p(1)) * (0.5*exp(sqrt(p(1))*t)+0.5*exp(-sqrt(p(1))*t)-1);
42 step_id2.x=stepsize*(k(2)/p(2)) * (0.5*exp(sqrt(p(2))*t)+0.5*exp(-sqrt(p(2))*t)-1);
43 end
44
45 %% Suchen, wann theoretischer Verlauf das Maximum der experimentellen
46 %% Impulsantwort erreicht
47 % Suchindizes initialisieren
48 search_index = 1;
49 search_index1 = 1;
50 search_index2 = 1;

```

```
51 % falls ein Parametersatz
52 if length(p) == 1
53     % suchen, bis Werte gleich
54     while step_id.x(search_index,1) ≤ xmax & search_index < xindexmax
55         search_index = search_index + 1;
56     end
57 % falls zwei Parametersätze
58 elseif length(p) == 2
59     % suchen, bis Werte gleich
60     while step_id1.x(search_index1,1) ≤ xmax & search_index1 < xindexmax
61         search_index1 = search_index1 + 1;
62     end
63     while step_id2.x(search_index2,1) ≤ xmax & search_index2 < xindexmax
64         search_index2 = search_index2 + 1;
65     end
66 end
67
68
69 % falls Graphen zu stark auseinanderdriften, Fehlermeldung ausgeben
70 if (search_index | search_index1 | search_index2) == xindexmax
71     % Ausgabe einer Fehlermeldung in rot
72     fprintf(2, 'Graphen driften zu stark auseinander. Kein Vergleich möglich!\n\n');
73 end
74
75 %% theoretischen Verlauf entsprechend ausschneiden
76 % falls ein Parametersatz
77 if length(p) == 1
78     step_id.x = step_id.x(1:search_index,1);
79     step_id.t = t(1:search_index,1);
80 % falls zwei Parametersätze
81 elseif length(p) == 2
82     step_id1.x = step_id1.x(1:search_index1,1);
83     step_id1.t = t(1:search_index1,1);
84     step_id2.x = step_id2.x(1:search_index2,1);
85     step_id2.t = t(1:search_index2,1);
86 end
87
88
89 %% beide Kurvenverläufe darstellen
90 % Zeitvektoren in Millisekunden Umrechnen
91 t = t * 1000;
92 % falls ein Parametersatz
93 if length(p) == 1
94     step_id.t = step_id.t * 1000;
95 % falls zwei Parametersätze
96 elseif length(p) == 2
97     step_id1.t = step_id1.t * 1000;
98     step_id2.t = step_id2.t * 1000;
99 end
100 % neues Anzeigefenster öffnen, mit Programmnamen, ohne Fensternummer
101 figure('Name', 'comp_id_re', 'NumberTitle', 'off');
102 % Anzeige offen halten, bis beide Graphen geplotted sind
103 hold on
104 % experimentellen Verlauf plotten
```

```
105 plot(t,x);
106 %% falls ein Parametersatz
107 if length(p) == 1
108 % theoretischen Verlauf als gepunktete Linie darstellen
109 plot(step_id.t,step_id.x,':');
110 % Überschrift
111 title({'\fontsize{12}\bf Simulation des unregulierten Systems ',...
112       '\it magnetischer Schwebekörper'});
113 % Anzeigen der Parameter p, k und Sprunghöhe in mA
114 ['\newline\fontsize{12}\rm Sprung von 0mA auf ', num2str(stepsize), 'mA',...
115   ' ', 'Simulation mit p = ', num2str(p), ' und k = ', num2str(k)];
116 % Datum und Uhrzeit ausgeben
117 ['\fontsize{10} aufgenommen am ',...
118   datestr(now, 'dd.mm.yyyy, HH:MM', 'local')]);
119 % Achsenbeschriftungen
120 xlabel('Zeit [ms]');
121 ylabel('Position des Körpers [mm]');
122 % Legende anzeigen
123 legend('real','ideal')
124 %% falls zwei Parametersätze
125 elseif length(p) == 2
126   plot(step_id1.t,step_id1.x,'--');
127   plot(step_id2.t,step_id2.x,'-.');
128   % Überschrift
129   title({'\fontsize{12}\bf Simulation des unregulierten Systems ',...
130         '\it magnetischer Schwebekörper'});
131   % Anzeigen der Parameter p, k und Sprunghöhe in mA
132   ['\newline\fontsize{12}\rm Sprung von 0mA auf ', num2str(stepsize),...
133     'mA      Simulation mit:'];
134   ['p = ', num2str(p(1)), ', k = ', num2str(k(1)), ' (klassisch) und ',...
135     'p = ', num2str(p(2)), ', k = ', num2str(k(2)), ' (Grey-Box)'];
136   % Datum und Uhrzeit ausgeben
137   ['\fontsize{10} aufgenommen am ',...
138     datestr(now, 'dd.mm.yyyy, HH:MM', 'local')]);
139   % Achsenbeschriftungen
140   xlabel('Zeit [ms]');
141   ylabel('Position des Körpers [mm]');
142   % Legende anzeigen
143   legend('real','ideal (klassisch)','ideal (Grey-Box)')
144 end
145 hold off
```

A.3.9. Listing: reset_PCI6014.m

```
1 %RESET_PCI6014 dient dem Rücksetzen des analogen Ausgangs AO0 der
2 %           Datenerfassungskarte NI PCI-6014 im Fehlerfall.
3 % RESET_PCI6014
4
5
6
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9
10 % analoges Ausgabeobjekt und zugehörige Kanäle erzeugen
11 AO = analogoutput('nidaq', 'Dev1');
12 ao_chan = addchannel(AO,0);
13 % 0V ausgeben
14 putsample(AO,0);
15 % Kanäle löschen
16 clear ao_chan;
17 % analoges Ausgabeobjekt löschen und aus Workspace entfernen
18 delete(AO);
19 clear AO;
```

A.4. Kennlinien

A.4.1. Statische Kennlinie der Regelstrecke

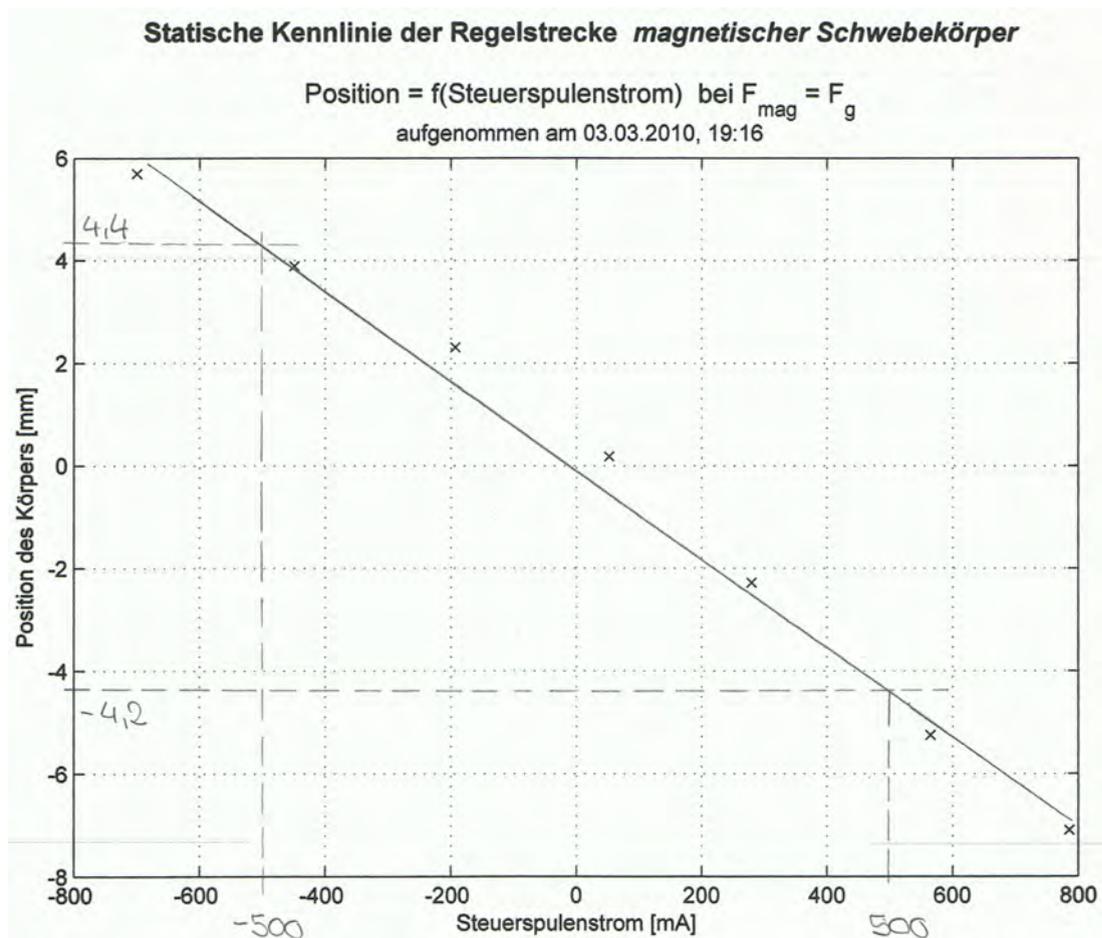


Abbildung A.10: Statische Kennlinie der Regelstrecke

A.4.2. Aufbereitete Sprungantwort der Regelstrecke

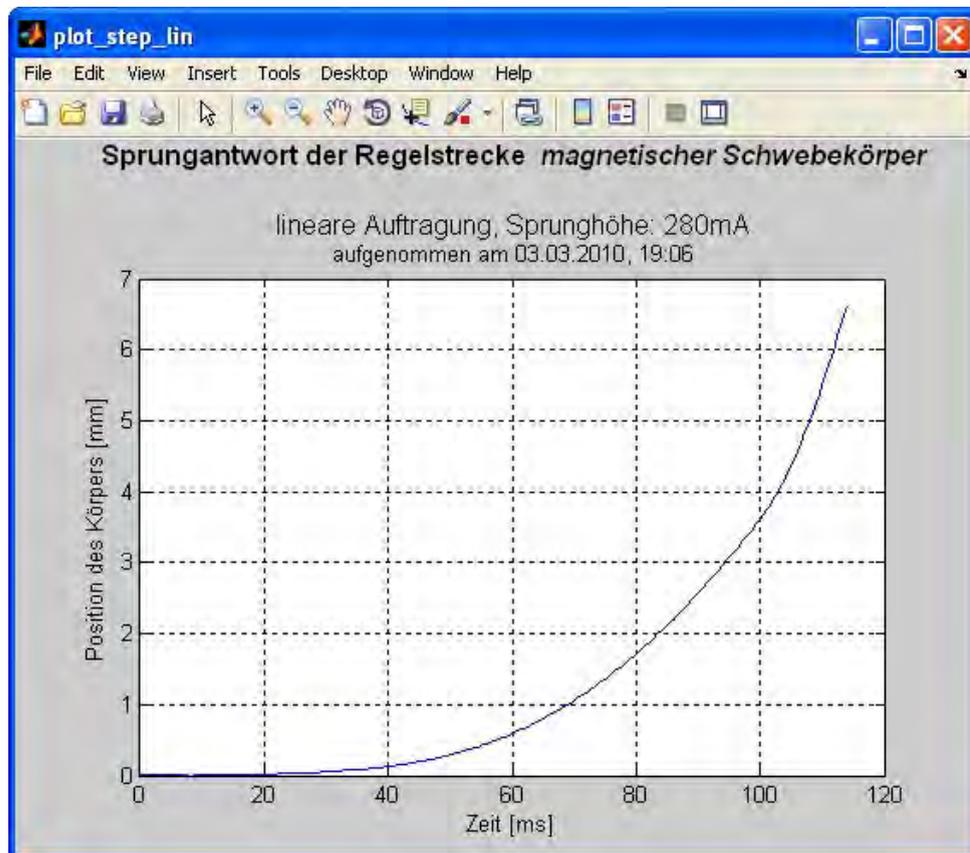


Abbildung A.11: Aufbereitete Sprungantwort der Regelstrecke

A.4.3. Näherungsgerade zur Auswertung der halblogarithmisch aufgetragenen Sprungantwort

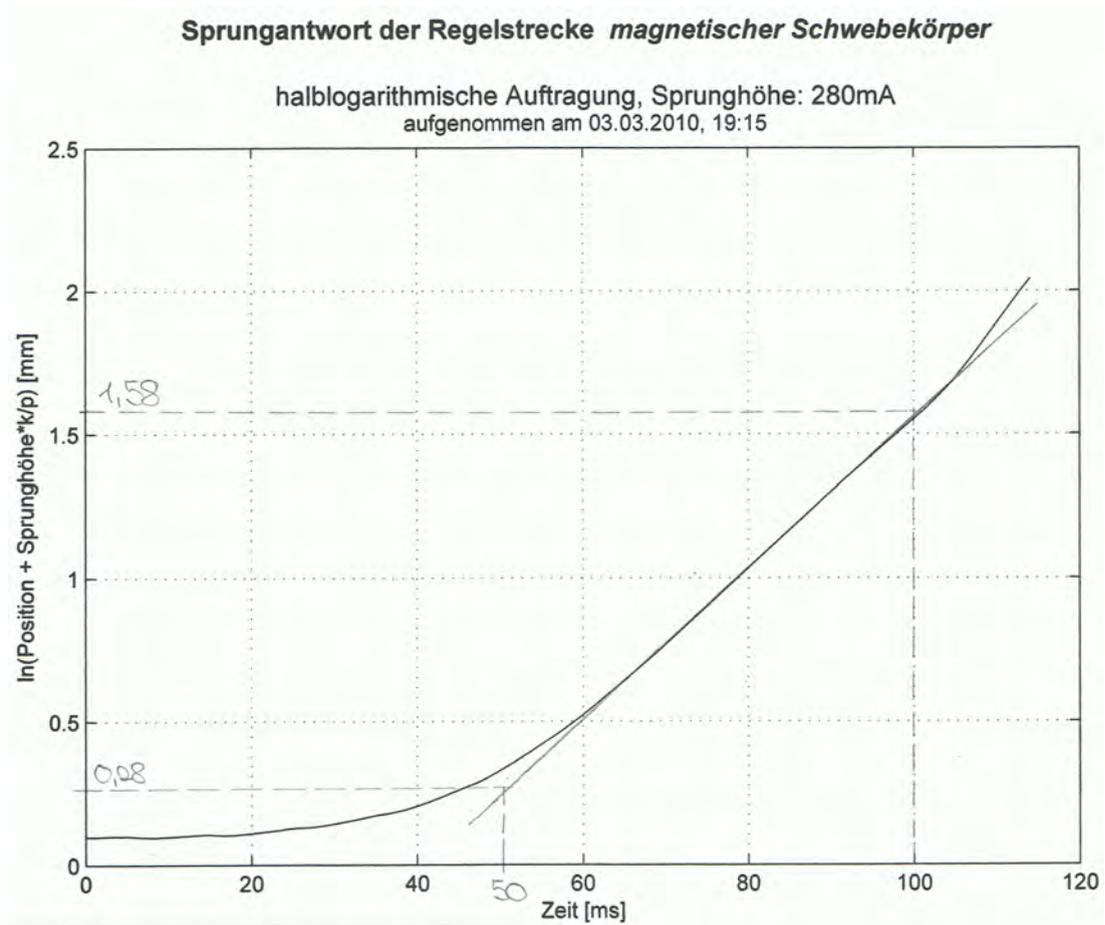


Abbildung A.12: Näherungsgerade zur Auswertung der halblogarithmisch aufgetragenen Sprungantwort

A.4.4. Vergleich von realer Sprungantwort mit idealen Verläufen aus klassischer Streckenanalyse und Grey-Box-Modellierung

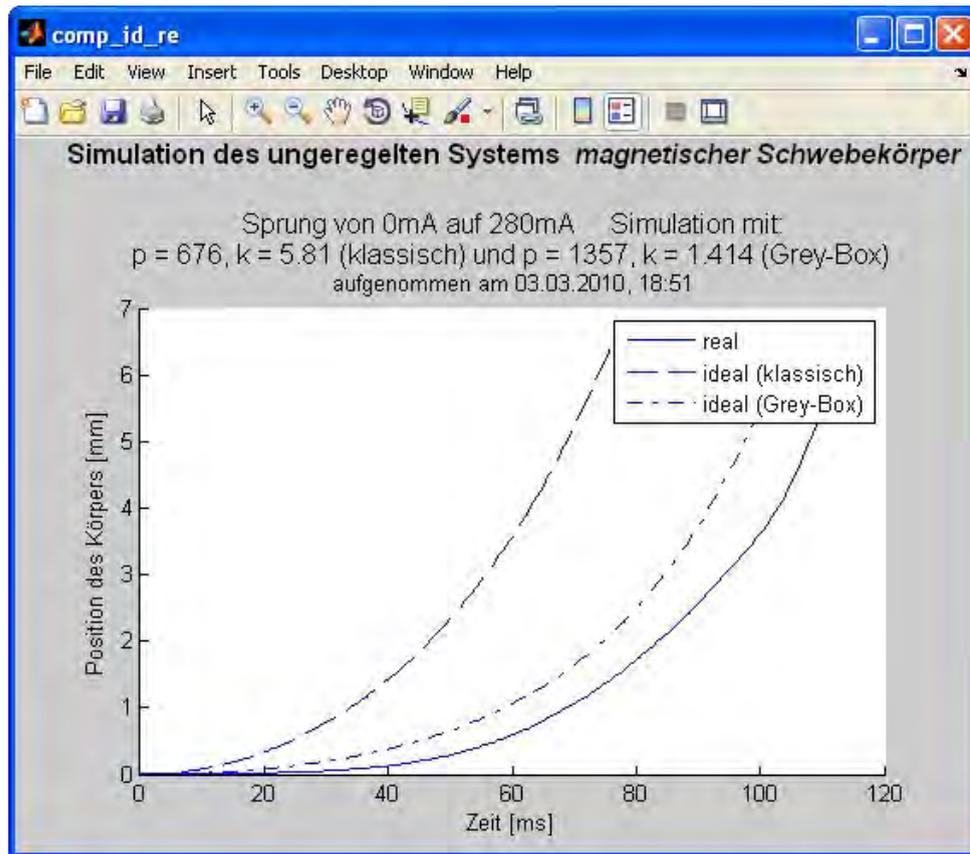


Abbildung A.13: Vergleich der idealen Sprungantworten mit dem realen Verlauf

A.4.5. Ausdrucke der Ergebnisse des ursprünglichen Praktikumsversuchs

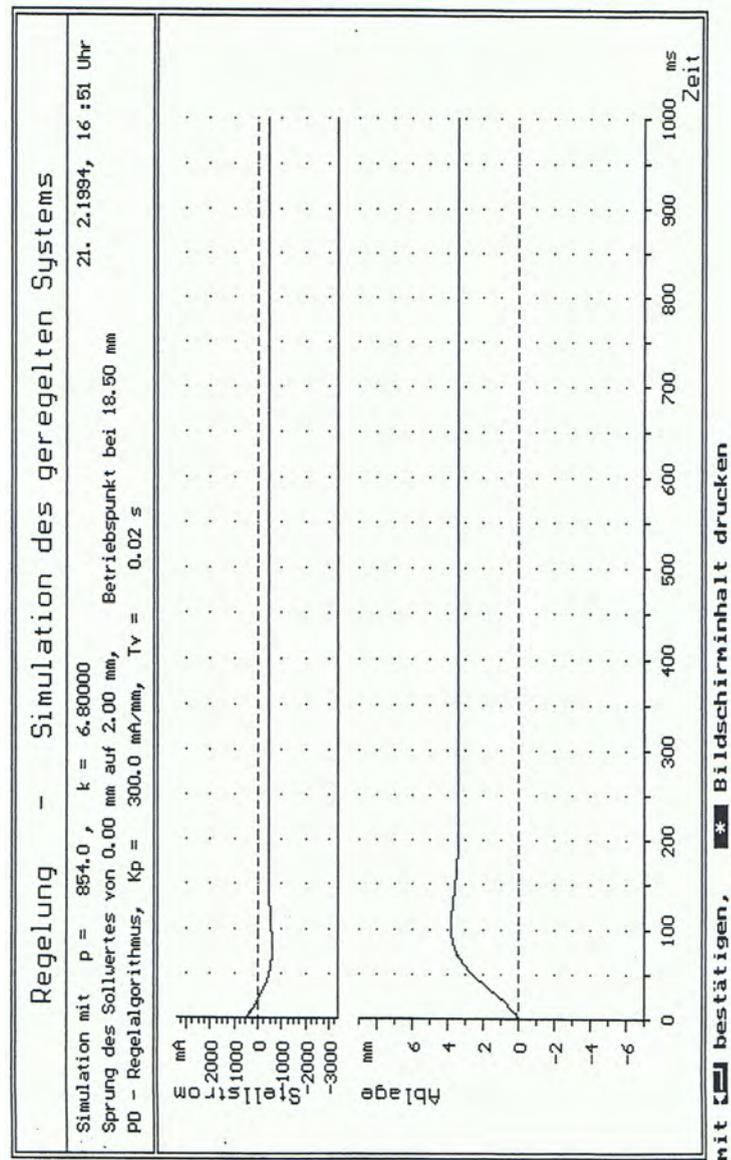


Abbildung A.14: Simulation des geschlossenen Regelkreises mit PD-Regler (ursprünglicher Versuch)

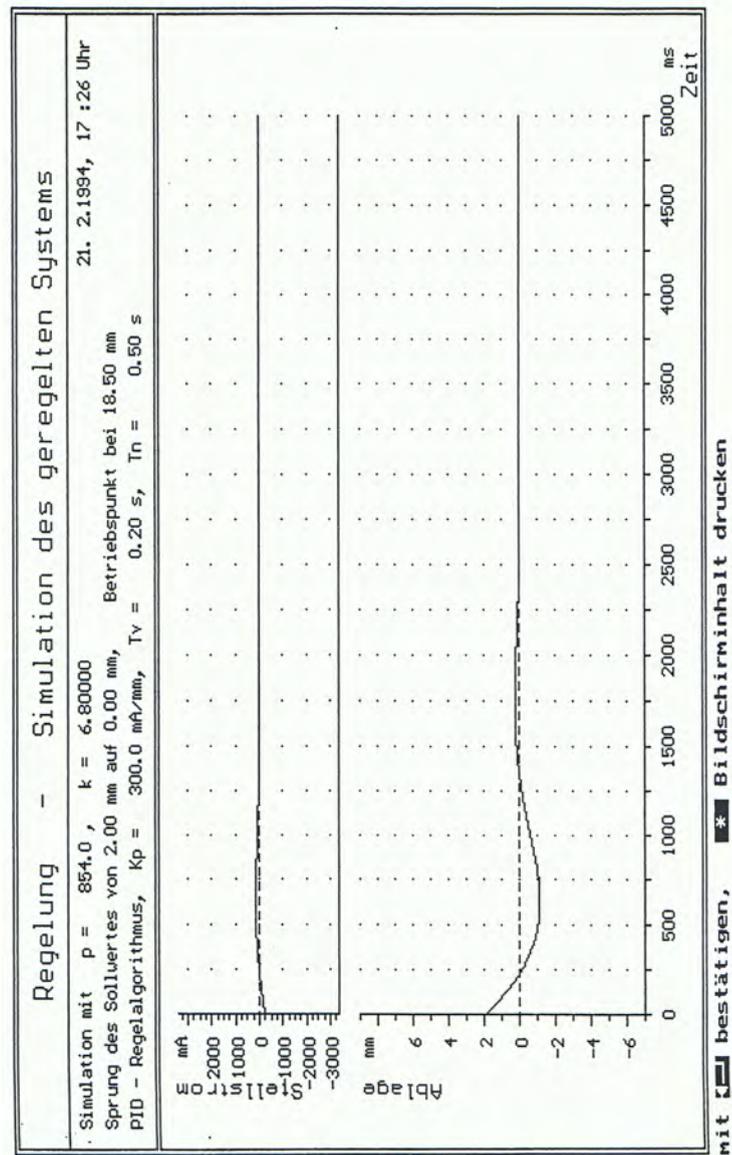


Abbildung A.15: Simulation des geschlossenen Regelkreises mit PID-Regler (ursprünglicher Versuch)

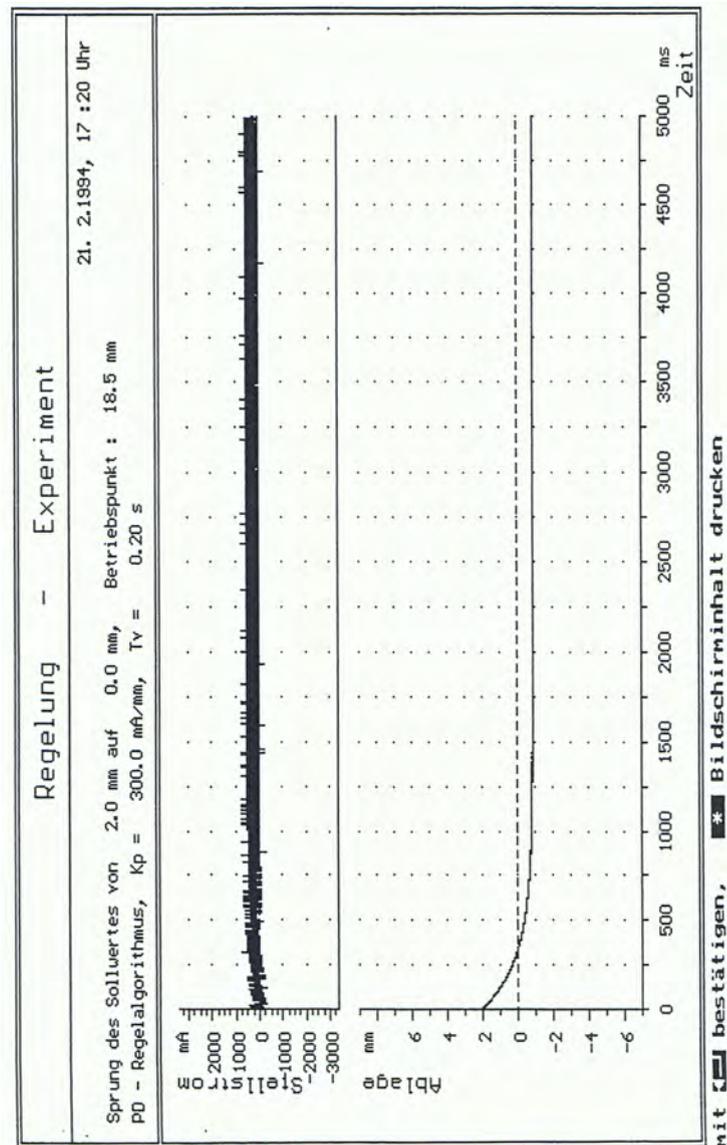


Abbildung A.16: Echtzeitexperiment mit PD-Regler (ursprünglicher Versuch)

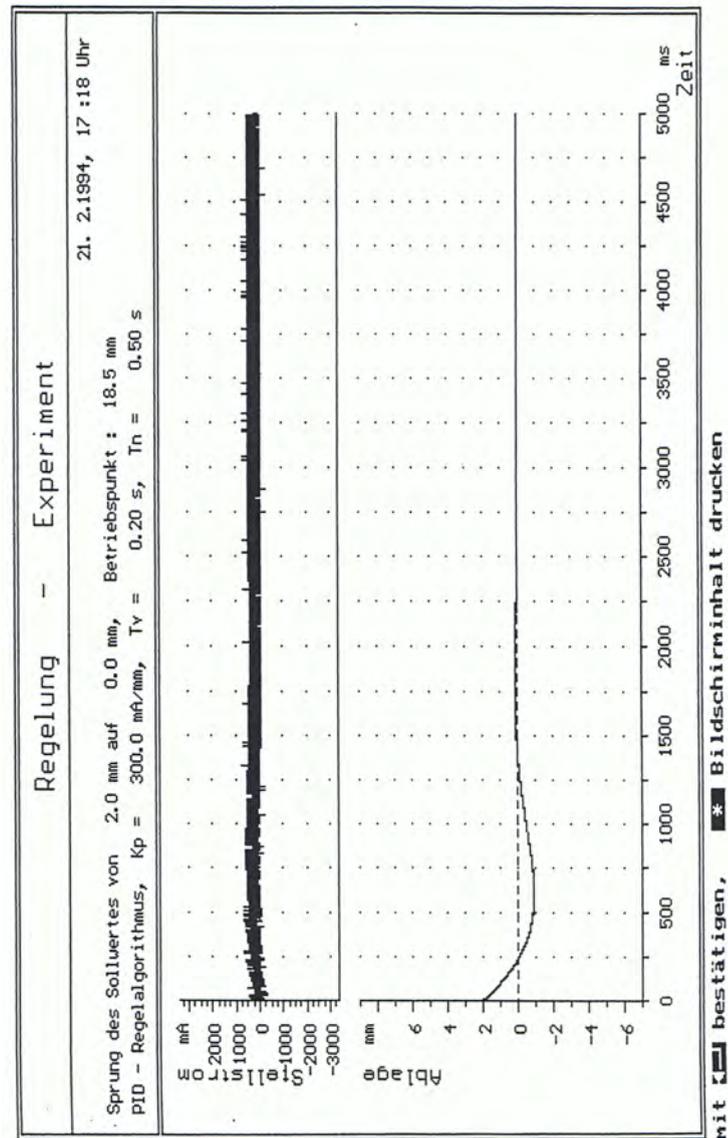
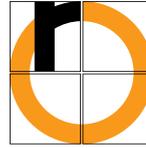


Abbildung A.17: Echtzeitexperiment mit PID-Regler (ursprünglicher Versuch)



Praktikumsanleitung

“Magnetischer Schwebekörper” – Streckenanalyse und Reglersynthese mit MATLAB/Simulink

Vorlesungsbegleitendes Praktikum
zur
Steuerungs- und Regelungstechnik

Studiengänge:
Elektro- und Informationstechnik,
Produktionstechnik

Teil A: Grundlagen

1. Versuchsaufbau

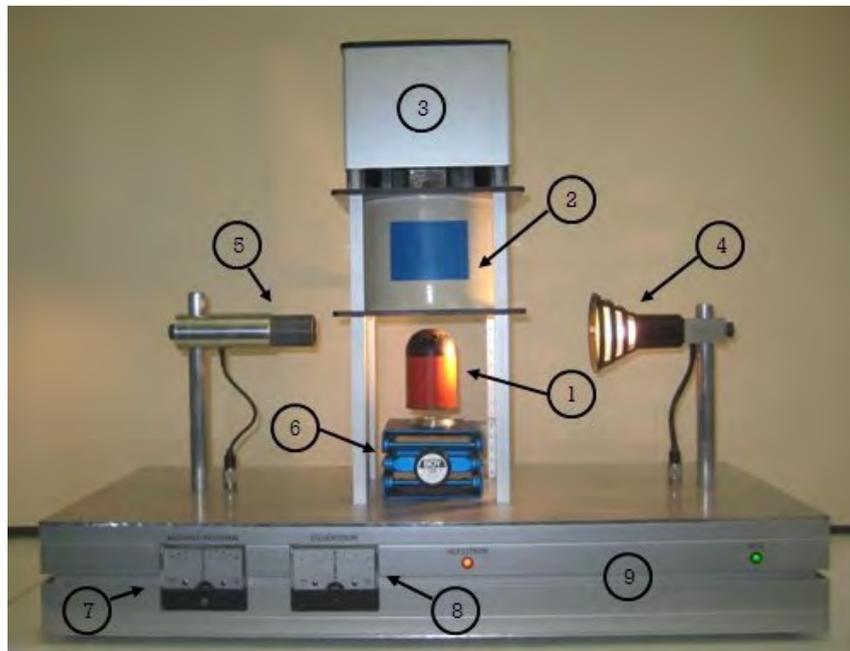


Abbildung A.1: Versuchsaufbau

- ① ferromagnetischer Körper
- ② Spulensystem
- ③ Lüfter
- ④ Lichtquelle
- ⑤ Sensor (Fotodiode)
- ⑥ höhenverstellbarer Scherentisch
- ⑦ analoge Anzeige des Positionssignals
- ⑧ analoge Anzeige des Steuerspulenstroms
- ⑨ Konstantstromquelle, Leistungsverstärker und Messelektronik

2. Versuchsbeschreibung

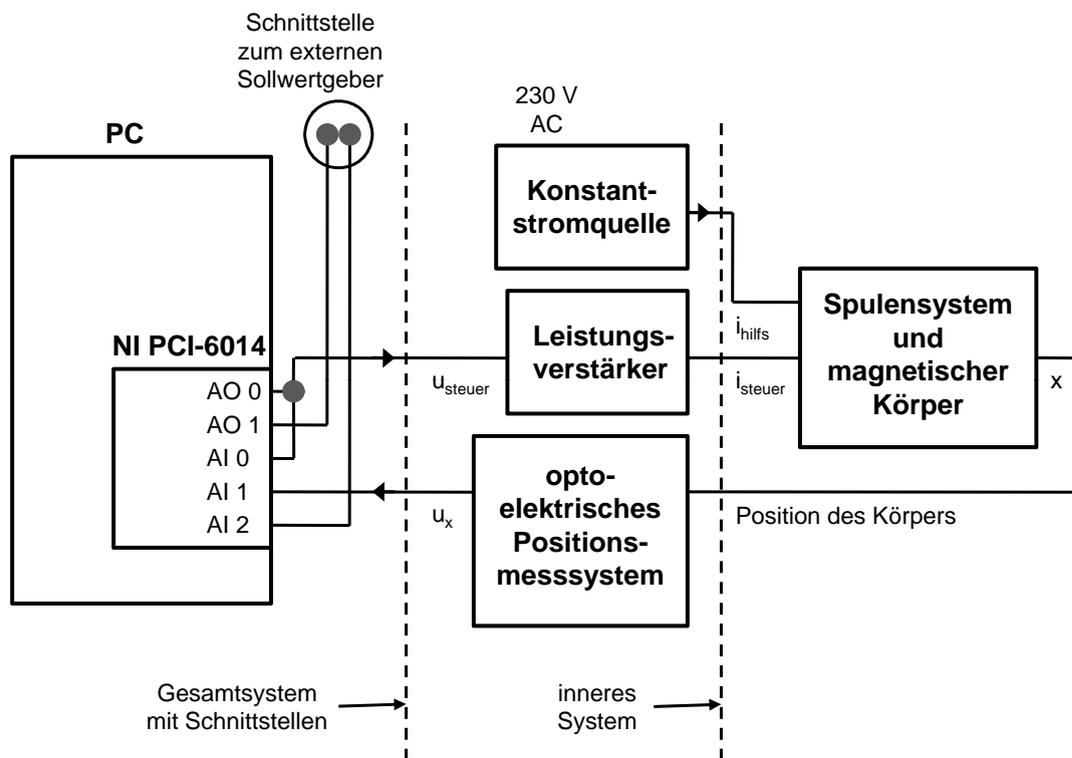


Abbildung A.2: Bockschaltbild des Versuchsaufbaus

Das Stellsignal u_{steuer} (in V) wandelt der Leistungsverstärker in den Strom für die Steuerspule i_{steuer} (in mA) um. Die Konstantstromquelle speist das Feld der Hilfsspule, welches im Betriebspunkt bei $x_0 = 0 \text{ mm}$ und $i_{\text{steuer}0} = 0 \text{ mA}$ für ein Gleichgewicht von Gewichtskraft und magnetischer Kraft am Körper sorgt (vgl. Abschnitt 4). Diesem Hilfsspulenfeld kann additiv das Steuerspulenfeld mit positiven und negativen Stromwerten überlagert werden.

Das optoelektrische Messsystem bestimmt die Position des Körpers. Es besteht unter anderem aus einer Lichtquelle (Niedervolt-Reflektorlampe) und der Fotodiode BPY47 als Sensor. Der zwischen Lampe und Sensor befindliche Schwebekörper schirmt abhängig von seiner Position (Höhe) das Licht unterschiedlich stark ab. Der Ausgangswert des Messsystems u_x ist proportional zu dieser Position.

Praktikumsziel: Ihre Aufgabe ist es, im Laufe dieses Praktikums eine Regelung für die nichtlineare, instabile Regelstrecke zu entwickeln, welche den magnetischen Körper mit möglichst geringem Energieverbrauch bei vorgebar Höhe in der Schwebelage hält.

Vorbereitung: Die so gekennzeichneten Aufgaben in allen Teilen dieser Praktikumsanleitung sind vor Praktikumsbeginn zu bearbeiten!

3. Verwendete Hard- und Software

3.1. Datenerfassungskarte

Dem Einlesen und Ausgeben der Schnittstellensignale u_x und u_{steuer} am PC dient die Multifunktions-Datenerfassungskarte NI PCI-6014 von National Instruments. Von den 16 verfügbaren analogen Eingängen sind nur drei ($AI 0$ bis $AI 2$, vgl. Abb.A.2) für diesen Versuch im Einsatz, von den analogen Ausgängen die beiden auf der Karte vorhandenen ($AO 0$ und $AO 1$). Alle Signale werden mit einer Rate von 2 kHz ausgegeben bzw. eingelesen. Der spezifizierte Ein- und Ausgangsspannungsbereich von $-10 \dots +10\text{ V}$ wird voll ausgenutzt.

3.2. Software

3.2.1. MATLAB

Ursprünglich als Programm für Matrizenrechnung geschrieben, ist aus MATLAB¹ mittlerweile eine mächtige Programmierumgebung für das Lösen verschiedenster technischer Aufgaben geworden. Wissenschaftler und Ingenieure auf der ganzen Welt schätzen die vielfältigen Möglichkeiten zur Umsetzung und Visualisierung ihrer Ideen. Durch diverse, auf spezifische Themengebiete zugeschnittene Erweiterungspakete, sogenannte Toolboxen, bietet MATLAB zusätzlich zeitsparende Unterstützung in vielen Bereichen der Forschung und Entwicklung.

Hinweise zum grundlegenden Umgang mit MATLAB:

- » Sie sollten das *Command Window* (Kommandozeile zur Eingabe von Befehlen), den *Workspace* (hier erscheinen die Variablen der aktuellen Sitzung) und die *Command History* im MATLAB-Fenster geöffnet halten. Das können Sie über den Button *Desktop* in der Menüleiste einstellen.
- » MATLAB verwendet den Punkt (.) als Dezimaltrennzeichen.

¹MATrix LABoratory

- » In Matrizen sind Zeilen durch Semicolon (;), Spalten durch Komma oder Leerzeichen getrennt.
A=[1 2 3] erzeugt einen Zeilenvektor,
B=[1;2;3] einen Spaltenvektor.
Vektoren bzw. Arrays sind eindimensionale Matrizen.
C(4,5) ist das Element in der 4. Zeile und der 5. Spalte der Matrix C,
C(1:3,5) das 1. bis 3. Element der 5. Spalte und
C(:,5) ist die ganze 5. Spalte von C.
Bei Zeilen- und Spaltenvektoren als Variable genügt eine Dimension zur Adressierung der Elemente.
- » Logisch zusammengehörende Daten werden oft in sogenannten Strukturvariablen baumförmig organisiert.
D.E.G.K(6) spricht das 6. Element des Vektors K in der dritten Unterebene der Struktur D an. E und G sind Strukturelemente der ersten und zweiten Ebene.
- » Nach Doppelklick auf eine Variable im Workspace erscheint deren Inhalt im *Variable Editor*. Hier können auch die Werte der Variablen manipuliert werden.
- » Mit den Tasten ↑ und ↓ erscheinen in der Kommandozeile schon einmal eingegebene Befehle.
- » Das Semicolon am Ende eines Statements unterdrückt die Ausgabe der Ergebnisse von Berechnungen in der Kommandozeile, was gerade bei Manipulationen von Arrays mit einer größeren Anzahl von Elementen empfehlenswert ist.
- » Zu allen Funktionen und -Skriptdateien bekommen Sie durch Eingabe von `help funktionname` (z.B. `sin`, `log`, `tf`²) Informationen über deren Zweck, Aufruf und ggf. Ein- und Ausgabevariablen im Command Window.
`doc funktionname` öffnet bei MATLAB-eigenen Funktionen die MATLAB-Hilfe. Diese Dokumentation kann auch über die Menüleiste mit *Help*→*Product Help* geöffnet werden.
- » Den Inhalt des Command Windows leeren Sie mit
`clc`, mit
`clear L` löschen Sie die Variable L aus dem Workspace.

3.2.2. Simulink

Simulink bietet dem Benutzer ein graphisches Interface, über das dynamische Systeme mittels Drag-And-Drop-Verfahren schnell modellier- und simulierbar sind. Mit entsprechenden Toolboxen und geeigneter Hardware ist ausserdem das Einlesen und Ausgeben von Signalen und damit eine Verwirklichung von Steuerungen und Regelungen realer Systeme, auch in Echtzeit, möglich.

²transfer function: Übertragungsfunktion

Hinweise zum grundlegenden Umgang mit Simulink:

- » Den Simulink Library Browser öffnen Sie mit dem Button  oder über *Start* (links unten im MATLAB-Fenster) → *Simulink* → *Library Browser*.
- » Sie können ein Modell in der Regel mit dem Kommandozeilenstatement `modelName` öffnen.
- » Alle Variablen aus dem Workspace sind in Simulink verfügbar und Simulink kann Variablen auch im Workspace ablegen, was diesen zu einer Schnittstelle zwischen MATLAB und Simulink macht.
- » Am schnellsten finden Sie einen Simulink-Block, wenn Sie im Suchfenster des *Library Browsers* dessen Namen eingeben, beispielsweise *Add* und *Sum* zum Addieren von Signalen, *PID* für einen PID-Regler, oder *Gain* zum Multiplizieren mit einer Konstante.
- » Blöcke können mit *Strg+R* (*rotate*) gedreht werden.
- » Sie verbinden Blöcke am schnellsten, wenn Sie die *Strg*-Taste gedrückt halten und nacheinander die zu verbindenden Blöcke anklicken. Sie können diese Pfeile aber auch mit der Maus ziehen.
- » Ein Fenster mit den Einstellung eines Blocks öffnet sich nach Doppelklick auf den Block.
- » Im Blockdiagramm können Sie mit *Copy* (*Strg+C*) und *Paste* (*Strg+V*) markierte Teile des Modells vervielfältigen.

3.2.3. Toolboxen

Die **Data Acquisition Toolbox** enthält eine Vielzahl von Werkzeugen, die das analoge und digitale Einlesen und Ausgeben von Signalen über PC-kompatible Geräte wesentlich erleichtern.

In MATLAB/Simulink erzeugte Daten können dabei als Signale auf einfachem Wege über entsprechende Hardware ausgegeben werden. Daten eingelesener Signale stehen zur Weiterverarbeitung in MATLAB/Simulink zur Verfügung.

Die **System Identification Toolbox** erstellt unter MATLAB/Simulink lineare und nicht-lineare mathematische Modelle von dynamischen Systemen auf Basis gemessener Ein- und Ausgangsdaten.

Es können einerseits Koeffizienten von gewöhnlichen Differentialgleichungen und Differenzgleichungen, welche mittels physikalischer Grundgesetze modelliert wurden, berechnet werden. Der datengetriebene Ansatz macht aber auch Systeme beschreibbar, welche sich nicht auf diese Art repräsentieren lassen, wie beispielsweise thermische

Prozesse eines Fluidums und elektromechanische Systeme.

Das resultierende Modell kann unter anderem bei der Simulation eines Systemausgangs bei einem bestimmten Eingangssignal Verwendung finden, oder beim Design einer Regelung helfen.

Die **Control System Toolbox** bietet Werkzeuge für die systematische Analyse, den Entwurf und letztlich die Optimierung steuerungs- und regelungstechnischer Systeme. Zunächst wird ein System in Form eines linearen Modells als Übertragungsfunktion, Pol-Nullstellen-Diagramm oder in der Zustandsraumdarstellung definiert.

Nachdem die Regelung modelliert ist, kann das Antwortverhalten sowohl im Zeit- als auch im Frequenzbereich untersucht und manipuliert werden.

Die automatisierten, interaktiven Techniken erleichtern typische Entwicklungsaufgaben der Steuerungs- und Regelungstechnik, wie zum Beispiel das Optimieren von Reglerparametern.

Mit dem **Real-Time Workshop** lässt sich aus MATLAB-Code oder Simulink-Modellen automatisch eigenständig lauffähiger C-Quellcode für Echtzeit- und Nicht-Echtzeit-Anwendungen generieren, packen und kompilieren.

Nachdem der generierte Code auf geeignete Hardware-Zielplattformen übertragen wurde und dort ausgeführt wird, ist dessen Beeinflussung aus der MATLAB/Simulink-Umgebung immer noch möglich.

Real-Time Windows Target wird benötigt, da die verwendete Datenerfassungskarte über keinen eigenen Prozessor verfügt, auf dem der mittels Simulink und Real-Time Workshop erzeugte Code eigenständig laufen könnte.

Durch Real-Time Windows Target kann ein und derselbe PC gleichzeitig als Host und als Target genutzt werden. Dabei wird ein Teil der CPU-Zeit des PCs als Echtzeit-Kernel mit maximaler Prozesspriorität genutzt, wobei der Systemtakt als primäre Zeitreferenz dient.

Die mitgelieferten I/O-Gerätetreiber erleichtern die Einbindung von Aktoren und Sensoren in eine Echtzeitregelung. Die Verbindung des Echtzeit-Kernels mit dem zugrunde liegenden Simulink-Modell ermöglicht ausserdem eine sich direkt auf den laufenden Code auswirkende Beeinflussung der Modellparameter. Da die PC-Aktivitäten dabei auf ein Minimum reduziert werden, sind Abtastraten bis zu 5 kHz erreichbar.

3.2.4. MATLAB-Hilfsprogramme und vorbereitete Simulink-Modelle

Zur Unterstützung des Praktikumsablaufs wurden für Sie einige MATLAB-Funktionen und -Skript-Dateien (m-Files), sowie Simulink-Modelle vorbereitet.

4. Mathematisches Modell der Regelstrecke

Die Natur der Regelstrecke "Magnetischer Schwebekörper" ist nichtlinear und instabil. Um diese unangenehmen Eigenschaften in den Griff zu bekommen, muss eine vereinfachte aber für Streckenanalyse und Reglerentwurf ausreichend genaue mathematische Beschreibung der Strecke entwickelt werden. Ausgangspunkt dabei sind die folgenden physikalischen Grundüberlegungen.

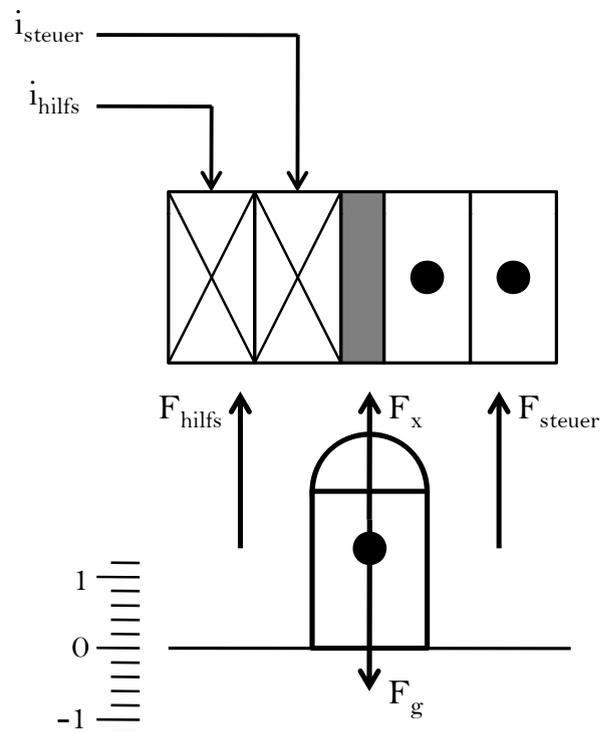


Abbildung A.3: Kräfte am magnetischen Körper

Ist die Position x in vertikaler Richtung, wie in Abbildung A.3 angedeutet, nach oben positiv definiert, ergibt sich folgende Kräftebilanz am Körper:

$$F_x = F_{\text{steuer}} + F_{\text{hilfs}} - F_g \quad (\text{A.1})$$

Diese wird mit $m \cdot \ddot{x} = F_x$ und $F_{\text{mag}} = F_{\text{steuer}} + F_{\text{hilfs}}$ zu

$$m \cdot \ddot{x} = F_{\text{mag}} - F_g = f(i_{\text{steuer}}, x) - F_g. \quad (\text{A.2})$$

Die resultierende Kraft auf den Körper in x -Richtung ergibt sich aus der Summe der durch die Funktion f beschriebene Magnetkraft F_{mag} und der entgegengesetzt dazu wirkenden Gewichtskraft F_g . f ist nichtlinear und liegt nur als experimentell gewinnbares Kennlinienfeld vor.

Die weitere Behandlung wird wesentlich vereinfacht, in dem man sich auf die Betrachtung des Systems in der Umgebung des Betriebspunktes bei $x = x_0 = 0$ beschränkt. An diesem Punkt wirkt nur die durch die Hilfsspule bereit gestellte Grundmagnetisierung ($i_{\text{steuer}} = i_{\text{steuer } 0} = 0$). Magnetische Kraft und Gewichtskraft heben sich gerade auf, wodurch letztere aus Gleichung A.2 entfällt, die an diesem Punkt zu

$$m \cdot \ddot{x} = F_{\text{mag}} \quad (\text{A.3})$$

wird. Bei einer linearisierenden Näherung ist somit nur noch F_{mag} zu behandeln:

$$\Delta F_{\text{mag}} = \left. \frac{\delta f}{\delta i_{\text{steuer}}} \right|_{i_{\text{steuer } 0}, x_0} \cdot \Delta i_{\text{steuer}} + \left. \frac{\delta f}{\delta x} \right|_{i_{\text{steuer } 0}, x_0} \cdot \Delta x \quad (\text{A.4})$$

Mit den Abkürzungen

$$f_i = \left. \frac{\delta f}{\delta i_{\text{steuer}}} \right|_{i_{\text{steuer } 0}, x_0}, \quad f_x = \left. \frac{\delta f}{\delta x} \right|_{i_{\text{steuer } 0}, x_0} \quad (\text{A.5})$$

und dem Weglassen des die lineare Beziehung anzeigenden Δ wird diese Gleichung zu

$$m \cdot \ddot{x} = f_i \cdot i_{\text{steuer}} + f_x \cdot x. \quad (\text{A.6})$$

Teilt man weiter noch durch die Masse m , stellt um und kürzt $p = \frac{f_x}{m}$ und $k = \frac{f_i}{m}$ ab, erhält man die Differentialgleichung 2. Ordnung

$$\boxed{\ddot{x} - p \cdot x = k \cdot i_{\text{steuer}}} \quad (\text{A.7})$$

zur Beschreibung des ungeredelten Systems. Die Konstanten p (in $\frac{mm^2}{s^2}$) und k (in $\frac{mm^3}{mA \cdot s^2}$) sind als Streckenparameter vor dem Entwurf einer Regelung zu bestimmen.

Da die inneren Systemgrößen nur über Schnittstellensignale zugänglich sind (vgl. Abbildung 1), muss für deren Darstellung im PC entsprechend konvertiert werden. Die Werte der Konvertierungsparameter sind

$283,0 \frac{mA}{V}$ für die Umrechnung von u_{steuer} auf i_{steuer} ,

$\frac{1}{283,0} \frac{V}{mA}$ für die Umrechnung von i_{steuer} auf u_{steuer} und

$-1 \frac{mm}{V}$ zur Bestimmung der Position x des Körpers aus dem Positionssignal u_x .

Vorbereitung: Ermitteln Sie durch Laplace-Transformation von Gleichung A.7 die Übertragungsfunktion des Systems.

$$G_S(s) = \quad (\text{A.8})$$

Vorbereitung, falls Teil C in Ihrer Praktikumsanleitung nicht (!) enthalten ist: Die Übertragungsfunktion des PID-Reglerblocks hat in Simulink die additive Form

$$G_{\text{PID}}(s) = P + \frac{I}{s} + D \cdot s, \quad (\text{A.9})$$

wobei die Werte von P, I und D im Block eingestellt werden können.

Prüfen Sie die Stabilität des geschlossenen Regelkreises mit den Reglertypen P ($I = D = 0$), PD ($I = 0$) und PID anhand des jeweils charakteristischen Polynoms. Formulieren Sie die Stabilitätsgrenzen der Reglerparameter (in Abhängigkeit von den Streckenparametern) allgemein. Begründen Sie ggf., warum gewisse Regler keine Stabilität in die Strecke bringen können.

Teil B1:

Aufnahme der Sprungantwort der Regelstrecke

1. Aufnahme der Sprungantwort

Legen Sie die Variablen `volt2curr`, `curr2volt` und `volt2pos` für Konvertierungsoperationen von inneren Systemgrößen und Schnittstellensignalen (vgl. Teil A, Abschnitt 4), sowie eine Variable mit dem Namen `stepsize` für die Vorgabe der Sprunghöhe in *mA* (Wert 280) im Workspace an. Öffnen Sie durch Eingabe von `get_step_data_todo` in der Kommandozeile das Simulink-Modell für die Aufnahme der Sprungantwort und speichern es unter `get_step_data` im MATLAB-Directory.

Untersuchen Sie, wie die Variable `stepsize` im den beiden *Step*-Blöcken Verwendung findet. Öffnen Sie über den *View*-Button in der Menüleiste den Library Browser und fügen einen *Gain*-Block für die Umrechnung des Stromsprungs auf Spannungswerte in das Modell ein. Verwenden Sie als Verstärkung die von Ihnen dazu im Workspace angelegte Variable. Benennen Sie den Block in `curr2volt` um und verbinden ihn mit seiner Umgebung. Benutzen Sie die von Ihnen für die Positionsermittlung erzeugte Variable im *Gain*-Block `volt2pos`.

Die aufgenommenen Daten sollen auch im Workspace gespeichert werden. Dazu müssen im *Scope*-Baustein (rechts unten im Blockschaltbild des Modells) entsprechende Einstellungen gemacht werden. Über den Button *Parameters* in der Symbolleiste des Scopes und den Reiter *Data history* finden Sie diese Einstellungen. Wählen Sie als Format *Structure with time* und vergeben Sie einen aussagekräftigen Variablennamen (z.B. `step280_01`). Speichern Sie das Modell.

Der C-Code des Modells wird generiert, indem man entweder aus dem Menü *Tools*→*Real-Time Workshop*→*Build Model* wählt oder *Strg+B* drückt. Nachdem die in der MATLAB-Kommandozeile verfolgbare Generierung und Kompilierung des Codes abgeschlossen ist, wird die Datenaufnahme mit dem Button *Connect To Target* in der Symbolleiste (links neben der Anzeige der *Simulation stop time*) initialisiert und mit dem Button *Start real-time code*, ebenfalls in der Symbolleiste, gestartet. Der nun auf dem Target laufende Code gibt einen der Sprunghöhe des Steuerspulenstroms entsprechenden Spannungssprung über *AO 0* aus und liest diesen Spannungsverlauf und das Positionssignal auf *AI 0* und *AI 1* ein.

Starten Sie die Messung erst, wenn Sie den magnetischen Körper durch Verstellen des Scherentischs so in den Arbeitspunkt bei 0 mm gebracht haben, dass er gerade noch nicht abhebt!

2. Sichtung und Aufbereitung der Daten

Sie können die Rohdaten der Sprungantwort mit bereits konvertiertem Positionssignal jetzt im *Scope* sichten. Nutzen Sie dabei die *Autoscale*-Funktion (Fernglas). Sie sehen, dass auch das Anschlagen des Körpers am unteren Ende des Spulensystems aufgezeichnet worden ist. Diese Werte stören bei den weiteren Analysen. Auch Anfangswerte der Ein- und Ausgangssignale des Systemkerns ungleich 0 sind nicht wünschenswert.

Prüfen Sie nach jedem der folgenden Schritte das Ergebnis im Variable Editor durch Doppelklick auf die entsprechende Variable im Workspace.

Vor der Aufbereitung ist eine Extraktion der relevanten Daten aus der von Simulink im Workspace angelegten Strukturvariablen sinnvoll.

Die Positionswerte befinden sich in der zweiten Spalte der Variablen `values` in der zweiten Unterebene dieser Struktur. Ermitteln Sie mit `max` den Index des Positionsmaximums. Weisen Sie den Positonsvektor bis etwa zum Index 10 vor dem Index dieses Maximums einer neuen Strukturvariable mit aussagekräftigem Namen in der ersten Unterebene zu, beispielsweise `prep_data.x`.

Extrahieren Sie Zeitvektor sowie Eingangssignal (1. Spalte von `values`) auch bis zu diesem Index, ebenfalls in die erste Unterebene der neuen Struktur, z.B. als `prep_data.t` und `prep_data.Ist`. Rechnen Sie dann die Eingangsspannungswerte u_{steuer} in Steuerpulvenstromwerte i_{steuer} des Systemkerns um. Den richtigen Faktor für die Konvertierung finden Sie u.a. über die Hilfe von `load_conv_param`.

Bestimmen Sie mit der Funktion `min` den Minimalwert des neuen Positionsvektors. Subtrahieren Sie diesen Wert, um die Anfangswerte auf 0 zu ziehen. Verfahren Sie mit dem Stromvektor der neuen Struktur genauso. Visualisieren Sie Ihr Ergebnis mit `plot_step_lin`. Das Ergebnis sollte in etwa wie in Abbildung B1.1 aussehen.

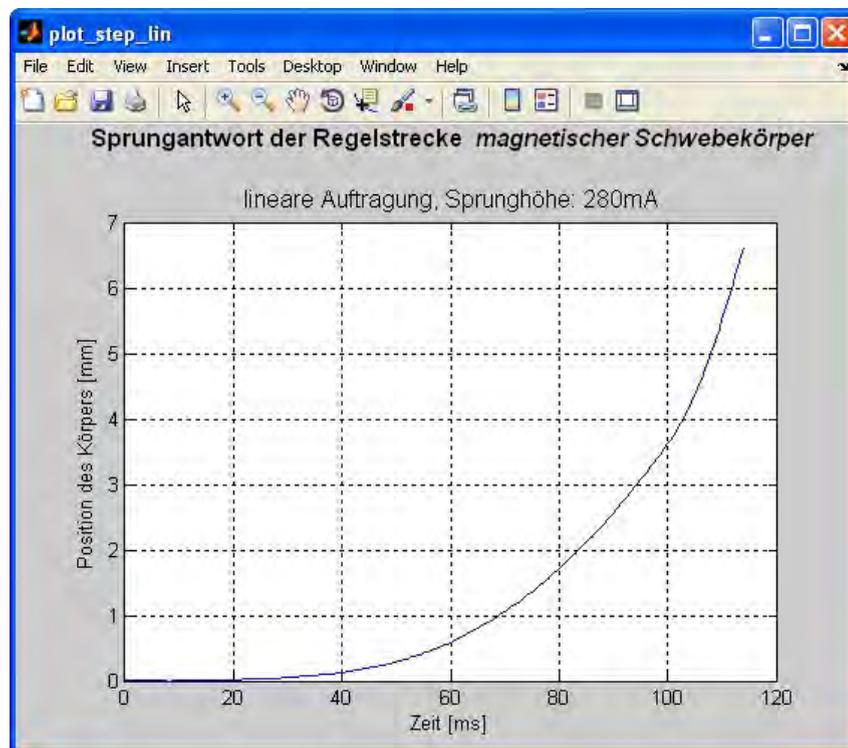


Abbildung B1.1: Aufbereitete Daten der Sprungantwort

Lagern Sie nun die gesamte Sequenz in eine Funktion aus. Öffnen Sie dafür das m-File `prep_step_data_todo` (z.B. Namen in der Kommandozeile markieren → Rechtsklick → *Open Selection*) und speichern es unter `prep_step_data` im MATLAB-Verzeichnis. Ergänzen Sie den Code an den gekennzeichneten Stellen und speichern die Funktion erneut. Testen Sie Ihre Funktion.

Teil B2:

Klassische Streckenanalyse

Die Streckenparameter können durch Ausnutzen von Gleichung A.7 aus der Aufnahme von statischer Kennlinie und Sprungantwort gewonnen werden. Die beiden entstehenden Graphen entsprechen zwei Gleichungen eines bestimmten Gleichungssystems, welche zur eindeutigen Berechnung der zwei unbekanntes p und k aus dem mathematischen Modell mindestens nötig sind.

1. Sprungantwort

Löst man Gleichung A.7 mit den Gesetzen der Differentialrechnung oder über die Sätze der Laplace-Transformation, erhält man die Sprungantwort der Regelstrecke als Ansatz

$$x(t) = i_{\text{steuer}} \cdot \frac{k}{p} \cdot 0,5 \cdot [e^{\sqrt{p} \cdot t} + e^{-\sqrt{p} \cdot t} - 2] \quad (\text{B2.1})$$

für die Bestimmung der Konstanten p , mit i_{steuer} als Sprunghöhe. Die anklingende Exponentialfunktion überwiegt bei größeren Werten von $\sqrt{p} \cdot t$ und $e^{-\sqrt{p} \cdot t}$ kann vernachlässigt werden. Nach Umstellen ergibt sich so

$$x(t) + i_{\text{steuer}} \cdot \frac{k}{p} = i_{\text{steuer}} \cdot \frac{k}{p} \cdot 0,5 \cdot e^{\sqrt{p} \cdot t} \quad (\text{B2.2})$$

und nach Bildung des Logarithmus

$$\ln [x(t) + i_{\text{steuer}} \cdot \frac{k}{p}] = \sqrt{p} \cdot t + \ln [i_{\text{steuer}} \cdot \frac{k}{p} \cdot 0,5]. \quad (\text{B2.3})$$

Im halblogarithmischen Maßstab entspricht das einer Gleichung der Form $y = ax + b$, wobei $a = \sqrt{p}$ die Steigung der Funktion nach dem Einschwingen ist. Mit dem Ergebnis aus Gleichung B2.1 sind die gesuchten Größen p und k nun zahlenmäßig berechenbar.

1.1. Bestimmung von p

Die graphische Ermittlung von \sqrt{p} erfordert nach Gleichung B2.3 eine halb-logarithmische Auftragung der Sprungantwort. Benutzen Sie dafür die Funktion `plot_step_log`. Drucken Sie den Graphen aus und nähern den Kurvenverlauf im linearen Bereich durch Einzeichnen einer Geraden. Bestimmen Sie deren Steigung und damit den Wert von p (Quadrieren nicht vergessen!).

$$p = \frac{\text{mm}^2}{\text{s}^2}$$

2. Statische Kennlinie

Untersucht man Positionen der Ruhelagen ($\ddot{x} = 0$) des Körpers bei verschiedenen Steuerpulenströmen, an denen sich magnetische Kraft und Gewichtskraft genau aufheben, erhält man die Punkte einer statischen Kennlinie der Regelstrecke. Im mathematischen Modell reduziert das Gleichung A.7 nach Umstellen auf

$$x(i_{\text{steuer}}) = -\frac{k}{p} \cdot i_{\text{steuer}} \quad (\text{B2.4})$$

Aus der Steigung dieser statischen Kennlinie ist also das Verhältnis $\frac{k}{p}$ bestimmbar.

2.1. Aufnahme der statischen Kennlinie

Informieren Sie sich über den Gebrauch der Funktion `get_static_data`. Rufen Sie die Funktion auf die in der Hilfe beschriebene Weise auf. Benutzen Sie für die Rückgabestruktur den Namen `static_data`. Lassen Sie sich von dem Dialog in der Kommandozeile durch die Messpunktaufnahme führen.

Stellen Sie die aufgenommenen Daten mit Hilfe der Funktion `plot_static_data` grafisch dar und drucken diese Darstellung aus.

2.2. Bestimmung der Steigung und des Werts von k

Ermitteln Sie nach Gleichung B2.4 durch Einzeichnen einer Näherungsgeraden für die Messpunkte die Steigung der statischen Kennlinie.

$$\frac{k}{p} = \frac{\text{mm}}{\text{mA}}$$

Berechnen Sie k mit dem Ergebnis aus Abschnitt 1.1.

$$k = \frac{\text{mm}^3}{\text{mA} \cdot \text{s}^2}$$

3. Vergleich der realen Sprungantwort mit dem theoretischem Verlauf

Durch `comp_id_re` kann der Verlauf der Sprungantwort aus Gleichung B2.1 mit den experimentell ermittelten Daten verglichen werden. Plotten Sie die Kurvenverläufe mit dieser Funktion und erklären Sie ggf. Abweichungen. Drucken Sie die Darstellung aus.

Teil B3: Parameterschätzung mittels Grey-Box-Modellierung

Bei Verwendung der System Identification Toolbox ist ein Grey-Box-Modellierung immer dann empfehlenswert, wenn die physikalischen Gegebenheiten des zu untersuchenden Systems, so wie bei diesem Praktikumsversuch, einsehbar sind. Es wird MATLAB dabei die Struktur des mathematischen Streckenmodells vorgegeben, was den Rechenaufwand bei der Suche nach einer Näherungslösung für den realen Verlauf der Sprungantwort reduziert.

Die im folgenden beschriebene Methode zur Ermittlung der Streckenparameter erfordert ein *iddata*-Objekt mit den aufgenommenen und aufbereiteten Daten der Sprungantwort der Regelstrecke. Des Weiteren ist die Übertragung von Gleichung A.7 in den Zustandsraum, sowie ein Zeilenvektor mit den Startwerten der Parametern p und k für den Schätzalgorithmus nötig, um damit ein *idgrey*-Modell der Strecke zu erzeugen. Mit *iddata*-Objekt und *idgrey*-Modell kann dann die Parameterschätzung initialisiert werden.

Eine Zustandsraumdarstellung ermöglicht die Überführung einer Differentialgleichung n -ter Ordnung in ein Gleichungssystem mit n Differentialgleichungen 1. Ordnung. Das vereinfacht zum einen grundsätzlich viele Problemstellungen, zum anderen kann speziell das matrixorientierte MATLAB mit den dabei entstehenden Vektoren und Matrizen effizient rechnen.

Um Gleichung A.7 in den Zustandsraum übertragen zu können, definiert man

$$\dot{x}_1 = x_2 \quad \text{und} \quad \dot{x}_2 = p \cdot x_1 + k \cdot i_{\text{steuer}} . \quad (\text{B3.1})$$

Es ergibt sich so die Vektordifferentialgleichung 1. Ordnung

$$\dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{b} \cdot i_{\text{steuer}} \quad (\text{B3.2})$$

mit Zustandsmatrix $\underline{A} = \begin{pmatrix} 0 & 1 \\ p & 0 \end{pmatrix}$, Steuervektor $\underline{b} = \begin{pmatrix} 0 \\ k \end{pmatrix}$ und Zustandsvektor

$\underline{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ in Matrixschreibweise

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ p & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ k \end{pmatrix} \cdot i_{\text{steuer}} . \quad (\text{B3.3})$$

Üblicherweise bezeichnet man im Regelkreis die Eingangsgröße mit u , die Ausgangsgröße mit y . Es gilt $y = x$, und

$$y = \underline{c}^T \cdot \underline{x} \quad (\text{B3.4})$$

wird mit dem Ausgangsvektor $\underline{c}^T = (1 \ 0)$ in Matrixschreibweise zu

$$y = x_1 = (1 \ 0) \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \quad (\text{B3.5})$$

Es sei darauf hingewiesen, dass die Position x des magnetischen Körpers nicht mit dem Zustandsvektor \underline{x} oder dessen Elementen zu verwechseln ist!

Im allgemeinen, auch für MIMO¹-Systeme geltenden Fall schreibt man die Gleichungen B3.2 und B3.4 mit der üblichen Notation als

$$\dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{B} \cdot u \quad (\text{B3.6})$$

und

$$y = \underline{C} \cdot \underline{x}. \quad (\text{B3.7})$$

Diese Darstellung des Streckenmodells mit den Matrizen \underline{A} , \underline{B} und \underline{C} ist in der Funktion `get_state_param` implementiert.

1. Sprungantwort

Nehmen Sie die Sprungantwort erneut auf und verwenden für die Aufbereitung der Daten die von Ihnen vervollständigte Funktion `prep_step_data`.

2. Erzeugung eines `iddata`-Objekts der Sprungantwort

Benutzen Sie die Funktion `iddata` mit den aufbereiteten Daten der Sprungantwort unter Verwendung eines aussagekräftigen Namens für die Ausgabevariable. Für Abtastperiode können Sie `1/Abtaste` angeben.

¹Multiple Input Multiple Output

3. Erzeugung eines `idgrey`-Modells der Regelstrecke

Legen Sie einen Zeilenvektor `par` mit Startwerten für den Schätzalgorithmus an. Das erste Element repräsentiert den Streckenparameter p , das zweite k . Verwenden Sie 1000 für `par(1)` und 10 für `par(2)`. Die Werte dienen als Startpunkt für den iterativen Schätzalgorithmus (vgl. 4).

Untersuchen Sie, wo in der Funktion `get_state_param` Zustandsmatrix, Steuervektor und Ausgangsvektor aus den Gleichungen B3.3 und B3.5 verwendet werden.

Erzeugen Sie mit `idgrey` ein zeitkontinuierliches lineares Grey-Box-Modell der Regelstrecke mit dem Namen `grey_mod`. Das Sample Intervall ist dabei auf 0 zu setzen. `FileArgument` kann beliebig gewählt werden, da es von `get_state_param` nicht verwendet wird. Der Dateiname (`MfileName`) `get_state_param` ist zwischen Hochkommas (') anzugeben.

4. Schätzung der Streckenparameter

Da nun sowohl ein `iddata`-Objekt der Sprungantwort als auch ein `idgrey`-Modell der Strecke im Workspace vorhanden sind, kann die Schätzfunktion `pem` (prediction-error minimization) mit diesen beiden Parametern aufgerufen werden. Dabei wird der Graph der experimentellen Sprungantwort in einem iterativen Verfahren durch das mathematische Modell mit Variation von dessen Parametern angenähert. Verwenden Sie den Namen `estimated_mod` für die Ausgabevariable und vergleichen dieses geschätzte Modell mit Gleichung B3.3. Notieren Sie sich die Näherungswerte der Streckenparameter.

$$p = \frac{\text{mm}^2}{\text{s}^2}$$

$$k = \frac{\text{mm}^3}{\text{mA} \cdot \text{s}^2}$$

5. Vergleich der realen Sprungantwort mit dem theoretischen Verlauf

Übergeben Sie der Funktion `comp_id_re` die notwendigen Parameter für einen Vergleich von Sprungantwort aus der Lösung von Differentialgleichung A.7 ($x(t) = i_{\text{steuer}} \cdot \frac{k}{p} \cdot 0,5 \cdot [e^{\sqrt{p} \cdot t} + e^{-\sqrt{p} \cdot t} - 2]$) und den experimentell ermittelten Daten. Drucken Sie die Dartsellung aus und beurteilen Sie das Ergebnis.

Teil C:

Reglerentwurf mit der Control System Toolbox

Die Control System Toolbox ermöglicht durch eine interaktive GUI¹ mit Darstellungen sowohl im Zeit-, als auch im Frequenzbereich einen effizienten und anschaulichen Reglerentwurf. Ziel ist die stabile Regelung des Istwerts, in diesem Fall der Position des magnetischen Körpers, bei Vorgabe von Einheitssprung oder Impuls als Sollwert.

Das `sisotool` wird mit der Streckenübertragungsfunktion aus Teil B gestartet. Falls Sie Teil B1 und B2 durchgeführt haben, verwenden Sie im weiteren die Streckenparameter, welche beim Vergleich der realen Sprungantwort mit dem idealen Verlauf das bessere Ergebnis geliefert haben.

1. Erzeugung der Streckenübertragungsfunktion G_S

Übertragungsfunktionen der Form

$$G_S(s) = \frac{b_m \cdot s^m + \dots + b_1 \cdot s + b_0}{a_n \cdot s^n + \dots + a_1 \cdot s + a_0} \quad (\text{C.1})$$

werden in MATLAB durch Zeilenvektoren mit den Koeffizienten a_i und b_j erzeugt. Üblicherweise verwendet man für das Zählerpolynom den Variablennamen `num`², für das Nennerpolynom `den`³. Das erste Element der Vektoren ist jeweils der Koeffizient der höchsten Potenz von s . Danach folgen der Reihe nach, analog zu Gleichung C.1, alle Koeffizienten bis einschließlich a_0 bzw. b_0 .

Legen Sie in einer Variablen namens `Gs` mit Hilfe der Funktion `tf` und den ermittelten Parameterwerten aus Teil B die Übertragungsfunktion der Regelstrecke an. Notieren Sie das Ergebnis und zeigen es dem Betreuer.

Gs =

¹Graphical User Interface: grafische Benutzeroberfläche

²numerator: Zähler

³denominator: Nenner

Vorbereitung: Geben Sie die Statements für eine Umsetzung der Polynome aus Gleichung A.8 mit den Bezeichnungen p und k für die Streckenparameter allgemein an.

num =

den =

Die Reglerübertragungsfunktion C^4 wird im *Compensator Editor* des *Control and Estimation Tools Manager* in Produktform standardmäßig mit den Zeitkonstanten $T_{Ni} = \frac{1}{\omega_{Ni}}$, $T_{Pi} = \frac{1}{\omega_{Pi}}$ und Verstärkung dargestellt.

Allgemein:

$$C(s) = K \cdot \frac{(1 + T_{N1} \cdot s) \cdot (1 + T_{N2} \cdot s) \cdot \dots}{(1 + T_{P1} \cdot s) \cdot (1 + T_{P2} \cdot s) \cdot \dots} \quad (\text{C.2})$$

Die Zeitkonstanten entsprechen dem Kehrwert der Kreisfrequenz, bei der die Pole und Nullstellen angesiedelt sind.

Geben Sie durch Bestimmung des Zähler- und Nennergrades der Ihnen bekannten Übertragungsfunktionen von (idealem) PD- und PID-Regler in additiver Form die Reglerübertragungsfunktionen analog zu Gleichung 4 allgemein an.

Hinweis: Eine Nullstelle bzw. ein Pol bei $\omega = 0$ lässt den Ausdruck $(1 + T \cdot s)$ zu s werden (Differenzierer- bzw. Integriererterm in der Übertragungsfunktion).

$$C_{PD} = K_{PD} \cdot \quad (\text{C.3})$$

$$C_{PID} = K_{PID} \cdot \quad (\text{C.4})$$

Prüfen Sie die Stabilität des geschlossenen Regelkreises mit den Reglertypen P, PD und PID in oben angegebener Produktform anhand des jeweils charakteristischen Polynoms. Begründen Sie ggf., warum gewisse Regler keine Stabilität in die Strecke bringen können. Formulieren Sie die Stabilitätsgrenzen der Reglerparameter (in Abhängigkeit von den Streckenparametern) für PD-Regler sowie PID-Regler allgemein.

⁴compensator: Regler

2. Initialisierung von `sisotool`

Starten Sie über die Kommandozeile das `sisotool` mit der Streckenübertragungsfunktion G_S . Wählen Sie im Reiter *Architecture* des *Control and Estimation Tools Manager* die in Abbildung C.1 dargestellte Regelkreisstruktur.

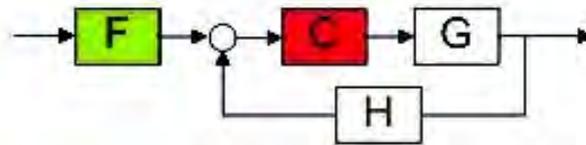


Abbildung C.1: Regelkreisstruktur

Im Modell des Regelkreises für diesen Praktikumsversuch sind die Elemente H und F nicht vorhanden. Ihre Übertragungsfunktion ist deshalb gleich 1 zu setzen.

Es ist zweckmäßig, die Fenster *Control and Estimation Tools Manager* mit dem *Compensator Editor*, *SISO Design* und *LTI Viewer* so auf dem Bildschirm zu arrangieren, dass alle drei gleichzeitig sichtbar sind. Diese Darstellungen sind untereinander verbunden und werden bei Änderungen im *Compensator Editor* oder im *SISO Design* automatisch aktualisiert. So kann die Systemreaktion im Zeitbereich bei Veränderungen der Werte von Polen, Nullstellen und Verstärkung in der Übertragungsfunktion C oder im Bode-Diagramm des Regelkreises beobachtet werden.

Den *LTI Viewer* öffnen Sie im Reiter *Analysis Plots*. Wählen Sie *Step* als *Plot Type* für *Plot 1* und setzen den Haken für den Inhalt des Plots bei *Closed Loop r to y*. Damit wird die Antwort des geschlossenen Regelkreises auf den Einheitssprung (Sollwert von 0 mm auf 1 mm) dargestellt. Drücken Sie den Button *Show Analysis Plot*. Setzen Sie im neu geöffneten Fenster den Haken in der Checkbox *Real-Time Update*.

Das *SISO Design* wird über den Reiter *Graphical Tuning* aktiviert. Wählen Sie *Root Locus* für *Plot 1* und für *Plot 2 Open-Loop Bode*. Stellen Sie für beide Plots *Open Loop 1* ein und drücken dann den Button *Show Design Plot*.

3. Reglerdesign

Nach Rechtsklick im Bereich *Dynamics* des *Compensator Editor* werden im Pop-Up-Menü über *Add Pole/Zero* der Reglerübertragungsfunktion C (in diesem Fall reelle!) Pole und Nullstellen zugefügt. Markieren Sie diese dann, können Sie im Bereich *Edit Selected Dynamics* die Werte $s_{p_i} = -\omega_{p_i}$ und $s_{n_i} = -\omega_{n_i}$ verändern. Im Feld *Compensator* erscheint der jeweilige Wert als $T_i = \frac{1}{\omega_i}$. $T = 0.001$ wäre beispielsweise durch die Eingabe $-1/0.001$ zu setzen. Die Verstärkung K beeinflussen Sie ebenfalls im Feld *Compensator*.

Jetzt ist auch das Arbeiten mit dem *SISO Design* möglich. Sowohl die Pole und Nullstellen, als auch die komplette Kurve im Bode-Diagramm können Sie mit der Maus verschieben.

Anmerkung: Die in diesem Praktikum entwickelten Reglerübertragungsfunktionen sind ideal. Für technisch realisierbare Systeme muss Zählergrad \leq Nennergrad gelten, was bei den hier verwendeten Reglern eine zusätzliche Polstelle erfordert, die ausreichend weit von den Polen des Systems entfernt liegt (etwa Faktor 1000).

Beobachten Sie bei allen Variationen das Antwortverhalten des geschlossenen Regelkreises im Zeitbereich. Achten Sie insbesondere auf die Achsenskalierungen.

3.1. PD-Regler

Entwerfen Sie mit dem oben beschriebenen Verfahren einen PD-Regler (vgl. Gleichung C.3). Orientieren Sie sich bezüglich Initialwerten für Verstärkung und Nullstelle an Ihren Berechnungen in der entsprechenden Vorbereitungsaufgabe zu diesem Praktikumsteil. Variieren Sie die Parameter so lange, bis Sie ein realistisches Antwortverhalten des geschlossenen Regelkreises ähnlich Abbildung C.2 erhalten. Die gestrichelte Linie im *LTI Viewer* symbolisiert den stationären Endwert.

Untersuchen Sie auch das Verhalten des Systems in der Nähe der Stabilitätsgrenze.

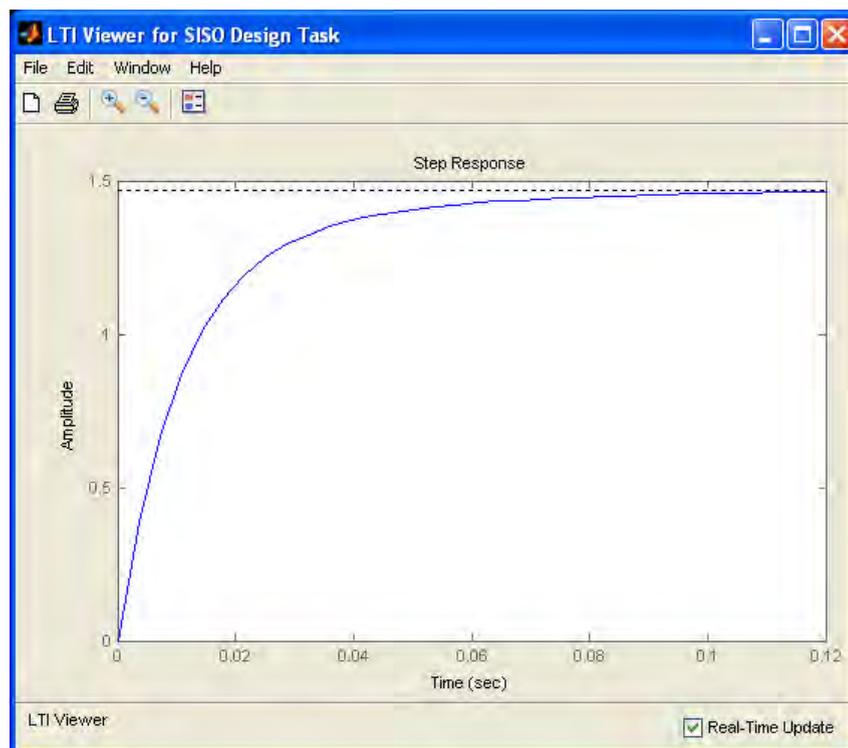


Abbildung C.2: Sprungantwort des Regelkreises mit PD-Regler

Notieren Sie sich die Reglerübertragungsfunktion C sobald Sie mit der Systemantwort zufrieden sind.

$$C_{PD} =$$

3.2. PID-Regler

Um die bleibende Regelabweichung des PD-Reglers zu kompensieren, wird nun um einen integralen Anteil erweitert.

Verändern Sie die Übertragungsfunktion C so, dass sie der eines PID-Reglers entspricht (vgl. Gleichung C.4). Verwenden Sie wieder Startwerte für die Parameter der Reglerübertragungsfunktion nach Ihrer Vorbereitung. Das System sollte, möglichst ohne Überschwingen (aperiodischer Grenzfall), nach annehmbarer Zeit den Sollwert erreichen.

Experimentieren Sie so lange, bis Sie mit der Systemantwort zufrieden sind und notieren sich dann wieder das Ergebnis.

$$C_{\text{PID}} =$$

Vorbereitung: Die Übertragungsfunktion des PID-Reglerblocks hat in Simulink die Form

$$G_{\text{PID}}(s) = P + \frac{I}{s} + D \cdot s, \quad (\text{C.5})$$

wobei die Werte von P , I und D im Block eingestellt werden können. Geben Sie allgemein die Konvertierungsgleichungen dieser Parameter mit den Darstellungen C.3 und C.4 für PD- und PID-Regler an.

$$P_{\text{PD}} = \quad (\text{C.6})$$

$$I_{\text{PD}} = \quad (\text{C.7})$$

$$D_{\text{PD}} = \quad (\text{C.8})$$

$$P_{\text{PID}} = \quad (\text{C.9})$$

$$I_{\text{PID}} = \quad (\text{C.10})$$

$$D_{\text{PID}} = \quad (\text{C.11})$$

Teil D:

Simulation des geschlossenen Regelkreises

Gängige Praxis ist es, Systeme vor Ihrer tatsächlichen Inbetriebnahme zu simulieren. Während der Simulationsphase können oft Schwachstellen entdeckt und behoben werden, was fehlerhaften Hardwareimplementierungen und dem damit verbundenen Aufwand entgegenwirkt, oder die Gefahr der Beschädigung teurer Hardware reduziert. Voraussetzung für erfolgreiche Simulation ist eine dem Zweck der Untersuchungen angepasste, ausreichend genaue Modellierung des realen Systemverhaltens.

Vorbereitung: Berechnen Sie mit Hilfe der Konvertierungsparameter die Grenzwerte für den Steuerspulenstrom, welche durch die Spannung u_{steuer} über den analogen Ausgang der Datenerfassungskarte eingestellt werden können.

$$i_{\text{steuer max}} = \quad \text{mA}$$

$$i_{\text{steuer min}} = \quad \text{mA}$$

1. Fertigstellung des Modells

Öffnen Sie das Simulink-Modell `loop_sim_todo` für die Simulation des geschlossenen Regelkreises und speichern es unter `loop_sim` im MATLAB-Arbeitsverzeichnis. Verwenden Sie für das Streckenmodell die von Ihnen berechnete Streckenübertragungsfunktion aus Gleichung A.8 mit den in Teil B ermittelten Parameterwerten.

Um auch die Grenzen des realen Systems nachzubilden, muss das Reglerausgangssignal beschränkt werden. Diese Beschränkung des Stellsignals ist ein gravierender Unterschied zum idealen Systemverhalten! Sie kommt sowohl durch die Spezifikationen der analogen Ausgänge der Datenerfassungskarte als auch die damit zusammenhängende Auslegung des Leistungsverstärkers zustande. Dieser würde bei betragsmäßig größeren Eingangssignalen als 10 V im stark nichtlinearen Bereich seiner Kennlinie arbeiten, was eine Regelung der Strecke erheblich erschwert. Als Grenzwerte im *Saturation*¹-Block sind daher die Werte aus obiger Vorbereitungsaufgabe einzusetzen. Speichern Sie das Modell.

¹saturation: Sättigung

Falls Teil C nicht (!) in Ihrer Praktikumsanleitung enthalten ist, verwenden Sie als Ausgangspunkt für die Simulationen in den folgenden Abschnitten Ihre Berechnungen aus der Vorbereitungsaufgabe am Ende von Teil A.

2. PD-Regler

2.1. Antwort des Regelkreises auf den Einheitssprung

Speichern Sie Ihr Modell unter dem Namen `loop_simPD`. Verwenden Sie die Ergebnisse des Reglerentwurfs aus Teil C für die Werte P, I und D des entsprechenden Blocks im Modell. Setzen Sie im *Step-Block* *Final value* auf 1, um den Sollwertsprung von 0 mm auf 1 mm zu simulieren. Speichern Sie erneut und starten die Simulation. Passen Sie die Simulationsdauer ausgehend von der Darstellung des Istwerteverlaufs ggf. an.

Analysieren Sie nun die verschiedenen Signale im Modell durch Doppelklick auf die hellgrün gekennzeichneten Blöcke. Benutzen Sie die *Autoscale*-Funktion (Fernglas), um den Anzeigebereich auszunutzen. Besonderes Augenmerk sollten Sie auf den Verlauf der Positionswerte und der Stellgröße i_{steuer} (begrenzttes Reglerausgangssignal) richten. Mit dem Mauszeiger kann ein Rechteck über einem Abschnitt des Kurvenverlaufs aufgezogen und so in diesen Bereich gezoomt werden.

2.2. Optimierung der Reglerparameter

Ziel ist es, dass der Schwebekörper seine durch den Sollwert vorgegebene Position in einem großen Bereich schnell, ohne Überschwingen (aperiodischer Grenzfall) und mit möglichst geringer Regelabweichung erreicht. Der Steuerspulenstrom sollte dabei einen wenig verrauschten Kurvenverlauf zeigen.

Experimentieren Sie sinnvoll durch Variation der Parameter des Reglers. Beaufschlagen Sie das System mit höheren, auch negativen Endwerten für den Sollwertsprung, um die Grenzen für die Sprunghöhe sowie des regelbaren Bereichs auszuloten. Versuchen Sie diesen Bereich durch kleine Änderungen der Reglerparameter etwas auszudehnen. Behalten Sie dabei auch immer das begrenzte Reglerausgangssignal im Auge! Lernen Sie, welcher Parameter für welches Verhalten verantwortlich ist. Wenn Sie mit dem Ergebnis zufrieden sind, speichern Sie das Modell und notieren sich die ermittelten Parameterwerte.

$P_{PD} =$

$D_{PD} =$

3. PID-Regler

Speichern Sie Ihr Modell unter dem Namen `loop_simPID`. Erweitern Sie den PID-Reglerblock um einen integralen Anteil, passen den proportionalen und differentiellen Anteil entsprechend ihren Ergebnissen aus Teil C an und gehen dann wie in Abschnitt 2 vor. Notieren Sie sich die Werte der Reglerparameter, sobald Sie mit dem Systemverhalten zufrieden sind. Speichern Sie das Modell.

$$P_{PID} =$$

$$I_{PID} =$$

$$D_{PID} =$$

4. Simulation in Echtzeit

Um dem Echtzeitexperiment mit realem System so nahe wie möglich zu kommen, soll die Regelung des geschlossenen Kreises mit Real-Time Workshop in Echtzeit simuliert werden. Der generierte, kompilierte und gepackte C-Code läuft dann auf dem mittels Real-Time Windows Target bereitgestellten Echtzeitkernel, kann jedoch zur Laufzeit noch Daten mit dem zugrunde liegenden Simulink-Modell austauschen. Das ermöglicht sowohl die Beeinflussung von Variablen als auch die Visualisierung der Signalverläufe in Echtzeit. Die Sollwertvorgabe wird deshalb etwas komfortabler gestaltet. Insgesamt sind dazu einige Änderungen im Modell vorzunehmen.

Speichern Sie Ihr Modell unter `loop_sim_rt`. Setzen Sie das Ende der Simulationszeit auf inf^2 . Die Simulation läuft dann so lange, bis Sie den Button *Stop real-time code* in der Symbolleiste drücken. Der *Simulation mode* ist von *Normal* auf *External* umzustellen. Markieren Sie das gesamte Modell mit *Strg+A* und schieben es mit der \rightarrow -Taste etwas nach rechts.

Ersetzen Sie nun den *Step*-Block zur Vorgabe des Sollwerts durch eine Kombination aus *Constant*- und *Slider Gain*-Block. Verwenden Sie für die Konstante den Wert 1. In deren *Main*-Tab wird die *Sample time* auf $1/2000$ gesetzt, was dem Sample-Intervall der realen Abtastung mit 2kHz (bzw. 2000 Samples/Sekunde) entspricht und von den folgenden Blöcken in diesem Modell übernommen wird. Der Schieberegler sollte etwa von -5 bis 5 reichen und auf 0 initialisiert werden. Nach diesen Anpassungen müsste `loop_sim_rt` ungefähr wie in Abbildung D.1 aussehen. Die Streckenparameter in der Darstellung sind nicht korrekt.

²infinity: positiv Unendlich

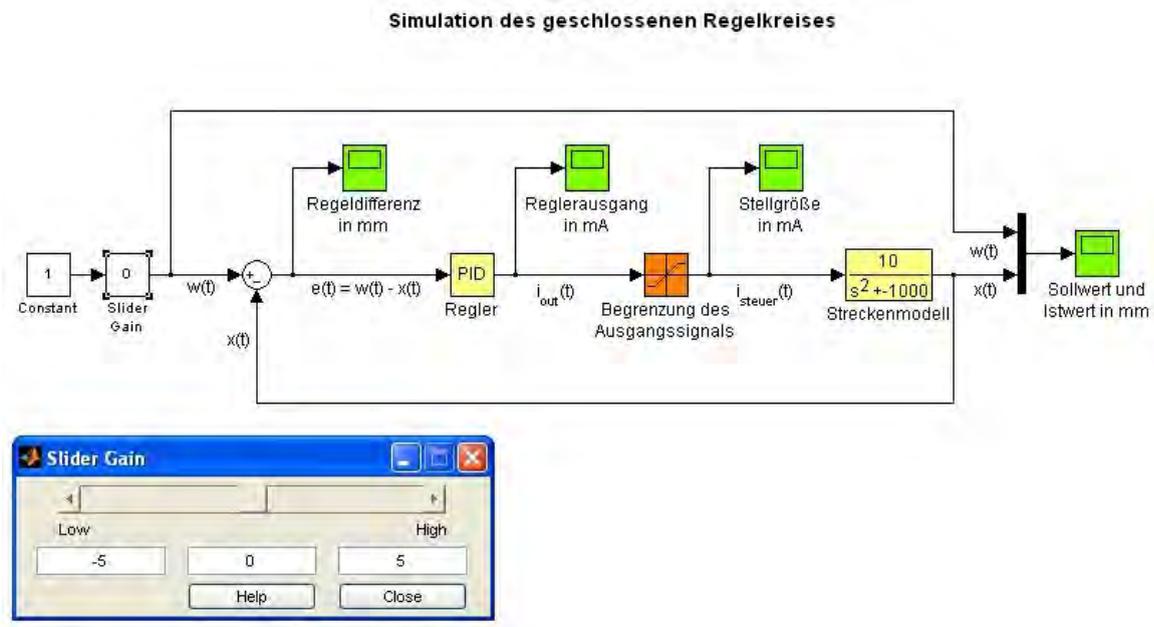


Abbildung D.1: Simulink-Modell `loop_sim_rt` mit geöffnetem *Slider Gain*

Um jeweils 10 Sekunden der Signalgraphen anzeigen zu können, ist ein entsprechend großer Datenpuffer notwendig. Wählen Sie dafür über die Menüleiste *Tools* → *External Mode Control Panel...* → *Signal & Triggering* einen ausreichend großen *Duration*-Wert (vgl. Vorbereitung).

Simulieren Sie sowohl mit PD-, als auch mit PID-Regler. Speichern Sie dazu das Modell unter `loop_sim_rtPD` und `loop_sim_rtPID`, nachdem Sie die passenden Reglerparameter eingesetzt haben. Initialisieren Sie die Generierung des C-Codes mit *Strg+B*, verbinden nach abgeschlossener Kompilierung das Modell über den Button  mit dem Real-Time Target und starten dann die Simulationen. Beobachten Sie wieder hauptsächlich Istwert, d.h. den Positionsverlauf, und Steuerspulenstrom. Die Position können Sie jetzt über den Schieberegler beeinflussen. Die Reglerparameter sind für weitere Optimierung nach den oben genannten Gesichtspunkten während der Laufzeit abänderbar. Sobald Sie mit dem Ergebnis zufrieden sind, kann die Regelung der realen Strecke mit diesen Parametern im folgenden Teil dieses Praktikums untersucht werden.

Vorbereitung: Berechnen Sie die Größe des Datenpuffers (Anzahl der Werte), um bei einer Abtastrate von 2 kHz 10 Sekunden des eingelesenen Signals speichern zu können.

Teil E:

Echtzeitexperiment

1. Setup

Öffnen Sie `loop_real_todo` und speichern das Model unter `loop_real`. In den beiden *Gain*-Blöcken zur Signalanpassung sind wieder die jeweiligen Konvertierungsparameter einzusetzen. Der Eingang der Regelstrecke muss auf den spezifizierten Ausgangsspannungsbereich der Datenerfassungskarte, also das reale Schnittstellensignal (vgl Abb. A.2), begrenzt werden. Überprüfen Sie, ob ein ausreichend großer Datenpuffer für die Anzeige von mindestens 10 Sekunden der Signalverlaufsgraphen eingestellt ist und speichern dann Ihr Modell.

2. Echtzeitregelung der realen Strecke

Die vorangegangene Simulation kommt dem Verhalten des realen Regelkreises sehr nahe. Allerdings wurden bei der Modellierung auch Streckeneigenschaften, wie beispielsweise der geringe Rauschanteil im Positionssignal, nicht berücksichtigt. Untersuchen Sie, wie und wo sich diese Mängel bemerkbar machen. Optimieren Sie ggf. die Reglerparameter weiter, um vor allem einen geringen Energiverbrauch des Systems (möglichst kleine Sprünge im Stellsignal) beim Halten des Sollwerts zu erzielen.

2.1. PD-Regler

Der PID-Reglerblock wurde für diese Echtzeitregelung etwas abgeändert. Sie können damit die Peglerparameter zur Laufzeit über die Ihnen bereits bekannten Slider Gains beeinflussen. Initialisieren Sie die Slider für P, I und D mit den Werten aus der Simulation und setzen die Endwerte so, dass Sie noch Spielraum für Variationen haben. Speichern Sie dann Ihr Modell unter `loop_realPD`.

Starten Sie wieder nach C-Codeerstellung und bestehender Verbindung mit dem Target die Regelung. Untersuchen Sie vor allem den Positionsverlauf und die Steuerspannung. Versuchen Sie, die Regelung nach o.g. Gesichtspunkt zu optimieren. Erweitern Sie den Sollwertebereich, um die Grenzen der Regelung zu bestimmen. Falls Ihnen die Verläufe der Graphen für eine Beobachtung zu kurz sind, erhöhen Sie den Duration-Wert und lassen dann den Code für das Modell neu erstellen. Speichern Sie, sobald Sie mit Ihrem Ergebnis zufrieden sind.

2.2. PID-Regler

Ändern Sie die Werte in den Slidern analog zur Beschreibung in Abschnitt 2.1. Speichern Sie Ihr Modell danach unter `loop_realPID` und gehen im weiteren wie bei der Untersuchung mit PD-Regler vor. Beenden Sie das Experiment, wenn Sie ein annehmbares Ergebnis erzielt haben und speichern dann erneut.

3. Einbindung des externen Sollwertgebers

Der Sollwert wird nun durch das Ausgangssignal eines als Spannungsteiler geschalteten Schiebepotentiometers, anstatt des vorher im Modell verwendeten Sliders bereitgestellt. Dafür ist die Versorgung des Spannungsteilers mit $+10\text{ V}$ über einen analogen Ausgang und das Einlesen der geteilten Spannung ($+10 \dots 0\text{ V}$) auf einem Eingang der Datenerfassungskarte nötig. Das eingelesene Signal muss dann noch so verschoben werden, dass ein Sollwertebereich von -5 bis $+5$ entsteht.

Speichern Sie Ihr Modell unter `loop_realHWcont`. Ein Doppelklick auf das Bild der Regelstrecke öffnet das darunter liegende Subsystem mit analogem Ein- und Ausgang. Kopieren Sie diese beiden Blöcke und setzen Sie, nachdem Sie etwas Platz geschaffen haben, im linken Bereich des Blockschaltbilds ein. Verwenden Sie im Feld *Output Channels* bzw. *Input Channels* die in Abbildung A.2 für die Anbindung des Sollwertgebers beschriebenen Kanäle. Die Nummerierung der Ein- und Ausgangskanäle der Datenerfassungskarte weicht von der bei *Analog Output* und *Analog Input* verwendeten ab.

Es gilt:

$$AO\ i = \text{Output Channel } (i + 1)$$

$$AI\ i = \text{Input Channel } (i + 1)$$

Vervollständigen Sie das Modell wie in Abbildung E.1 auf der nächsten Seite gezeigt. Die Farbe eines Blocks können Sie nach Rechtsklick mit dem Mauszeiger darüber durch *Background Color* verändern. Lassen Sie den C-Code erstellen, verbinden Sie das Modell mit dem Target und starten dann die Regelung. Erweitern Sie den Regelbereich, indem Sie einen *Gain*-Block verwenden und den Offset entsprechend anpassen. Experimentieren Sie sowohl mit PID- als auch mit PD-Regler.

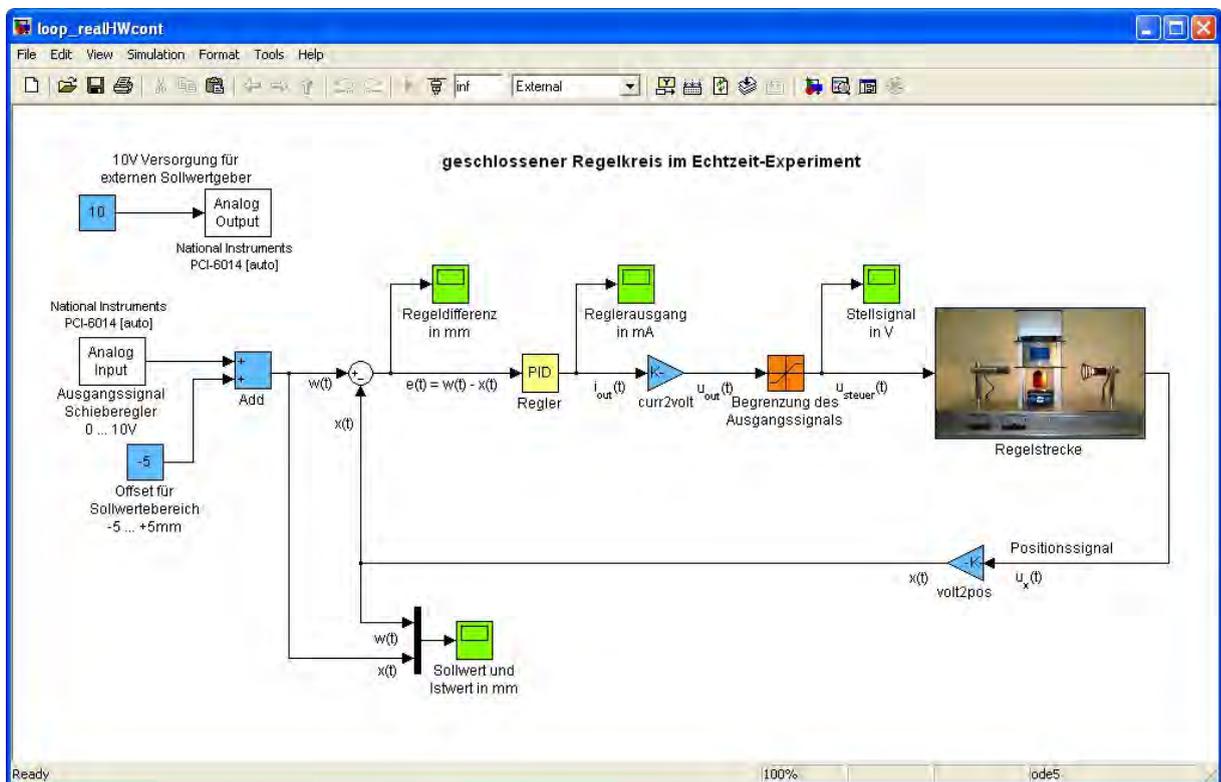


Abbildung E.1: Simulink-Modell zur Einbindung des externen Sollwertgebers

Gratulation, jetzt haben Sie's geschafft!

