Leopold Wiener, BSc

Entwicklung eines Regelalgorithmus für höhenverstellbare Schreibtische

Masterarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur

eingereicht an der
Technischen Universität Graz

Betreuer
Univ.-Prof. Dipl.-Ing. Dr.techn Martin Horn

Institut für Regelungs- und Automatisierungstechnik

Graz, Mai 2016

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Kurzfassung

Ziel dieser Arbeit war es, einen neuen Regelalgorithmus für die Firma Logicdata, welche Steuerungen für elektrisch verstellbare Möbel baut und entwickelt, zu entwerfen. Es wurden drei Algorithmen entworfen und getestet. Der erste passt die vorgegebene Sollposition an, wenn ein zu großer Unterschied zwischen Sollposition und tatsächlicher entsteht. Der zweite vergrößert die Abbrems- und Beschleunigungszeit der Sollgröße, wenn während des Abbrems- bzw. Beschleunigungsvorgangs ein zu großer Unterschied zwischen Sollposition und Istposition entsteht. Bei dem dritten Regelalgorithmus wird ein flachheitsbasierter Vorsteuerungsentwurf verwendet. Dazu mussten ein mathematisches Modell aufgestellt und die dazugehörigen Parameter identifiziert werden. Durch ein abschließendes Testen der Regelalgorithmen konnte erkannt werden, dass die Anpassung der Sollgröße bzw. die Anpassung der Abbrems- und Beschleunigungszeit sehr gut funktionierten und die pendelnden Schwingungen deutlich reduziert werden konnten. Durch die flachheitsbasierte Vorsteuerung konnte ein sehr gutes Führungsverhalten erzielt werden, ob hier die pendelnden Schwingungen korrigiert werden, muss anhand eines realen Modells mit den pendelnden Schwingungen untersucht werden.

Abstract

The aim of this study was a new control algorithm for Logicdata which builds and develops controls for electrically adjustable furniture, to the prevention of oscillating vibrations. The first adjusts the preset desired position if the difference between target position and actual results is too large. The second increases the braking and accelerating time if the difference between position setpoint and actual target size is too large during the braking or acceleration process. In the third control algorithm, a flatness-based feedforward control design is used. This required a mathematical model to be set up and the corresponding parameters needed to be identified. During a final test of the control algorithms, it could be recognized that the adaptation of the desired size or adjusting the braking and accelerating time works very well and the oscillating vibrations were reduced significantly. By the flatness-based feedforward the reference-variable response of the control system is very good. Whether the oscillating vibrations were reduced needs to be tested on a real model with oscillating vibrations.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

In diesem Kapitel werden die Motivation dieser Arbeit, die Aufgabenbeschreibung sowie die Zielsetzung dargestellt. Außerdem werden die Methodik sowie der Aufbau dieser Arbeit beschrieben.

1.1. Einführung

Regelungstechnik ist ein wesentlicher Bestandteil des alltäglichen Lebens. Immer, wenn ein Sollwert mit einem Ist-Wert verglichen und daraus eine Korrektur abgeleitet wird, handelt es sich um eine Regelung. Beispiele dafür sind ein Heizthermostat oder die Regelung der Körpertemperatur. In einem Auto sorgen mehrere Regler für Fahrsicherheit und Fahrkomfort, unter anderem die Geschwindigkeitsregelung des Tempomats oder auch die Regelung des Schlupfs bei einem ABS.

Geregelt werden können physikalische, chemische oder andere Größen. Eine der häufigsten ist die Positionsregelung. Die Firma Logicdata beschäftigt sich unter anderem mit Positionsregelung mittels Mikrocontroller für Gleichstrommotoren, die auch in höhenverstellbaren Tischen verbaut werden.

In dieser Arbeit werden zwei permanenterregte Gleichstrommaschinen mit Schneckengetriebe, welche in einem Tisch verbaut sind, positionsgeregelt (siehe Abbildung 1.1). Die Position der Tischbeine soll möglichst gut übereinstimmen, damit sich die Tischplatte, auch bei unterschiedlichen Belastungen, nicht neigt. Außerdem soll sich die Tischplatte so sanft wie möglich bewegen, ohne zu großer Einbußen bei der Bewegungsgeschwindigkeit in Kauf nehmen zu müssen, da sich dies auf die Benutzerfreundlichkeit auswirken kann, woraus sich ein negativer Einfluss auf die Wirtschaftlichkeit der Antriebe ergeben kann.

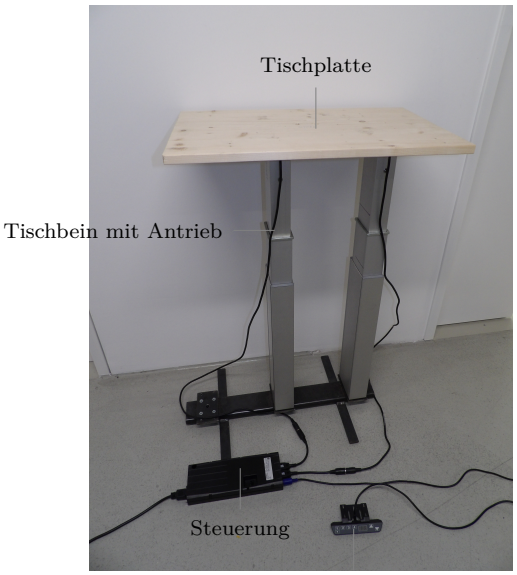


Abbildung 1.1.: Verwendeter Tisch mit Steuerung und Bediengerät von Logicdata

1.2. Aufgabenbeschreibung

Logicdata baut und entwickelt Steuerungen für elektrisch verstellbare Möbel. Die Businessunit LogicOffice hat sich auf Elektronik und Antriebstechnik für elektrisch höhenverstellbare Schreibtische spezialisiert. Üblicherweise werden für solche Tische zwei oder mehrere bürstenbehaftete Gleichstrommotoren mit Inkrementalgeber über eine Steuerung synchronisiert.

Zur Synchronregelung wird aktuell pro Motor ein digitaler PI-Positionsregler mit gemeinsamer Sollwertvorgabe verwendet. Um auch bei höherer Belastung ein Auseinanderlaufen der Motoren zu verhindern, wird bei maximaler Aussteuerung eines Motors die Sollwertvorgabe verringert. Dieses Konzept funktioniert bei Motoren mit ähnlichem Verhalten sehr einfach und gut. Es hat jedoch bei mechanisch eher schlechteren Antrieben oder bei sehr großen Toleranzen zwischen den zu synchronisierenden Antrieben den Nachteil, dass beim Anfahren kurze, pendelnde Schwingungen entstehen. Diese sind besonders stark, wenn beim Stehenbleiben ein Positionsunterschied entsteht. Ziel dieser Arbeit ist es nun, einen neuen Regelungsalgorithmus zu finden, mit dem die pendelnden

Schwingungen verringert beziehungsweise für den Benutzer nicht mehr spürbar gemacht werden können.

1.3. Methodik

Zum Entwurf mancher Regelalgorithmen ist es notwendig, ein mathematisches Modell aufzustellen. In dieser Arbeit wurde dies mittels Differentialgleichungen in Form eines „Average Modells“ erledigt.

Dieses mathematische Modell enthält alle wichtigen Informationen über die verwendeten Antriebe, jedoch sind nicht alle Größen bekannt und müssen daher bestimmt werden. In dieser Abhandlung wird ein einfacher Weg dargestellt wie diese Daten gefunden werden können.

Anschließend wird der bestehende PI-Regler getestet und drei neue Regelungsansätze untersucht. Bei den ersten beiden Ansätze handelt es sich um eine Erweiterung des bestehenden Reglers. Ein besseres Verhalten kann einerseits durch Anpassen der Beschleunigungs- bzw. Abbremsdauer erreicht werden und andererseits durch nachstellen der Sollposition. Das heißt, läuft ein Motor „über das Ziel hinaus“ so wird die Sollposition nachgestellt, damit der zweite Antrieb zum ersten aufschließen kann. Der zweite Ansatz beschäftigt sich mit einem flachheitsbasierten Vorsteuerungsentwurf und einem P-Regler zum Ausgleichen von Parameterschwankungen.

Abschließend werden diese Entwürfe anhand des realen Systems getestet.

1.4. Aufbau der Arbeit

Zu Beginn beschäftigt sich diese Arbeit mit den theoretischen Grundlagen der Modellbildung und der Parameteridentifikation einer bürstenbehafteten Gleichstrommaschine sowie mit dessen Reglerentwurf.
Der Aufbau sowie die Durchführung der Versuche zur Identifikation der Parameter werden im anschließenden Kapitel behandelt. Hier sind auch die ermittelten Werte zu finden. Weiters wird hier gezeigt, wie der Regler implementiert und getestet wurde.

Abschließend werden die Ergebnisse dieser Arbeit zusammengefasst und diskutiert.

1.5. Zielsetzung

Im Rahmen dieser Diplomarbeit soll ein neuer Regelalgorithmus entwickelt werden, der insbesondere das Anfahrverhalten mit Sanftstart verbessert. Der Regelalgorithmus soll so konzipiert sein, dass er auf der bestehenden Hardware funktionieren würde. Es ist allerdings nicht Teil der Diplomarbeit, den Regelalgorithmus in die bestehende Firmware zu implementieren.
Für den Entwurf der flachheitsbasierten Vorsteuerung, muss das System (hier der Gleichstrommotor) hinreichend bekannt sein. Deshalb ist es notwendig, die benötigten Parameter zu identifizieren. Es soll außerdem verifiziert werden, ob die entworfenen Regelkreise ein ausreichend gutes Verhalten besitzen.

2. Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen zur Modellbildung und zur Parameteridentifikation einer Gleichstrommaschine beschrieben. Anschließend werden verschiede Möglichkeiten erarbeitet, um den in Kapitel 1.2 beschriebenen Problemen entgegenzuwirken.

2.1. Modellbildung

Um ein System mathematisch hinreichend genau beschreiben zu können, muss ein Ersatzschaltbild erstellt werden, welches ausreichende Informationen des realen Systems beinhaltet. Durch Einstellen des pulsweitenmodellierten Signals u_{PWM} kann die Eingangsspannung des verwendeten Antriebs geregelt werden.

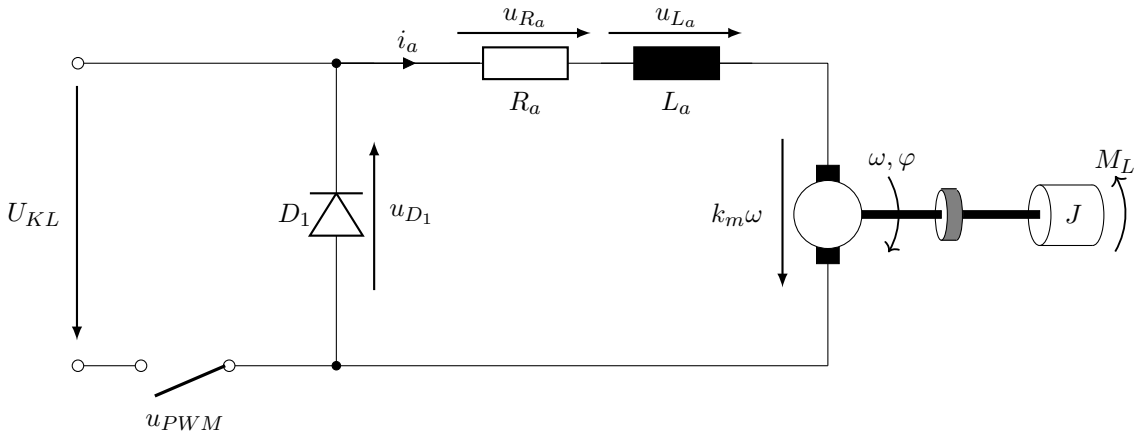


Abbildung 2.1.: Allgemeines Ersatzschaltbild des Gleichstrommotors

In Abbildung 2.1 ist das dazugehörige Ersatzschaltbild dargestellt. Der Schalter u_{PWM} soll symbolisch die regelbare Spannung durch das PWM-Signal darstellen. Ist der Schalter geschlossen, fließt der Strom i_a durch den Motor, wird der „Schalter“ geöffnet, „treibt“ die Spule den Strom über die Freilaufdiode D_1 weiter, dies entspricht einem Tiefsetzsteller. Somit kann durch u_{PWM} die Eingangsspannung zwischen 0 V und U_{KL} eingestellt bzw. geregelt werden, wobei U_{KL} die angelegte Versorgungsspannung ist.

Der elektrische Teil des Motors wird mit dem ohm'schen Ankerwiderstand R_a, der Ankerinduktivität L_a und durch der induzierten Spannung im Motor $k_m\omega$ dargestellt, außerdem wird hier das mechanische Modell skizziert.

2.1.1. Theorie zum Mittelwertmodell

Für den Vorsteuerungsentwurf muss ein mathematisches Modell aufgestellt werden. Da es sich jedoch um ein schaltendes Netzwerk handelt - es wird zwischen zwei Modellen hin und her geschaltet - (siehe Abbildung 2.2) können Probleme beim Reglerentwurf sowie bei der Simulation entstehen. Abhilfe schafft hier das Mittelwertmodel, welches beide Modelle „vereint“.

In Abbildung 2.2 gibt T_E die Dauer des Schaltzustandes „High“ und T_A die Dauer des Schaltzustandes „Low“ an, womit sich die Periodendauer T_S zu

$$T_S = T_E + T_A \tag{2.1}$$

ergibt [Michel, 2011].

Um ein „Mittelwertmodell“ zu erhalten, wird ein arithmetischer Mittelwert aus den beiden mathematischen Modellen (siehe Gleichung 2.5 mit den Gleichungen 2.18 und 2.24) gebildet. Es ist ersichtlich, dass bei höheren Frequenzen der Pulsweitenmodulation das Mittelwertmodell genauer ist.

$$\begin{aligned} \dot{x}_E &= f_E(x_E) \\ \dot{x}_A &= f_A(x_A) \\ \dot{x} &= \frac{1}{T_S} \cdot (f_E(x_E) \cdot T_E + f_A(x_A) \cdot T_A) \end{aligned} \tag{2.2}$$

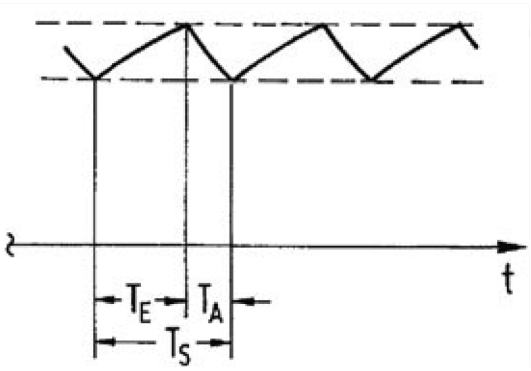


Abbildung 2.2.: Verlauf eines schaltenden Netzwerkes [Michel, 2011]

Mit

$$\frac{T_E}{T_S} = p \tag{2.3}$$

und

$$\frac{T_A}{T_S} = 1 - p \tag{2.4}$$

ergibt sich das „Mittelwertmodell“ zu

$$\dot{x} = f_E(x_E) \cdot p + f_A(x_A) \cdot (1 - p)$$

$$\boxed{\dot{x} = f_A(x_A) + p \cdot (f_E(x_E) - f_A(x_A))} \tag{2.5}$$

2.1.2. Modell für PWM-Signal „High“

Der Betriebszustand „High“ entspricht einem geschlossenen Schalter u_{PWM} in Abbildung 2.1, und ist eine Zeitdauer T_E (siehe Abbildung 2.2) aktiv.

2.1.2.1. Ersatzschaltbild

Ist der Schalter u_{PWM} geschlossen, fällt an der Freilaufdiode D_1 die gesamte Spannung U_{KL} ab, da diese in Sperrrichtung betrieben wird. Sie ist für dieses Modell somit irrelevant. Daraus ergibt sich das Ersatzschaltbild, wie in Abbildung 2.3 dargestellt.

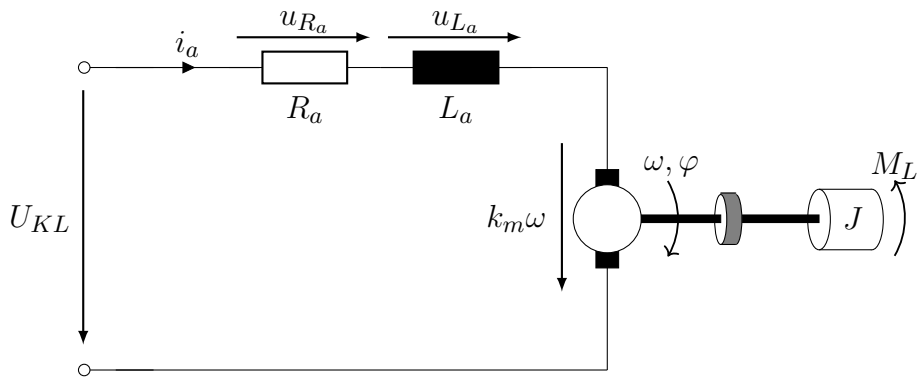


Abbildung 2.3.: Ersatzschaltbild des Gleichstrommotors bei PWM „High“

2.1.2.2. Aufstellen der Differentialgleichungen

Wie in Abbildung 2.3 ersichtlich wird, entsteht das System aus einem elektrischen und einem mechanischen Teil. Für beide Teile muss eine Differentialgleichung aufgestellt werden, um das gesamte Verhalten zu beschreiben.

Für den elektrischen Systemanteil wird zunächst die Kirchoff'sche Maschengleichung

$$U_{KL} = u_{R_a}(t) + u_{L_a}(t) + k_m \cdot \omega(t) \tag{2.6}$$

aufgestellt. Mit

$$\begin{aligned} u_{R_a}(t) &= R_a \cdot i_a(t) \\ u_{L_a}(t) &= L_a \cdot \frac{\mathrm{d}i(t)}{\mathrm{d}t} \end{aligned} \tag{2.7}$$

eingesetzt, ergibt sich die erste Differentialgleichung.

$$\frac{\mathrm{d}i}{\mathrm{d}t} = -\frac{R_a}{L_a} \cdot i(t) - \frac{k_m}{L_a} \cdot \omega(t) + \frac{1}{L_a} \cdot U_{KL} \tag{2.8}$$

Um die Position (hier der Winkel φ) regeln zu können, wird eine weitere Differentialgleichung aufgestellt.

$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = \omega \tag{2.9}$$

Zur Beschreibung des mechanischen Systemanteils, wird der Momentensatz

$$J \cdot \frac{\mathrm{d}\omega}{\mathrm{d}t} = k_m \cdot i_a - M_L \tag{2.10}$$

verwendet. $k_m \cdot i_a$ beschreibt das Moment, welches im Inneren des Motors erzeugt wird und J das Trägheitsmoment der Maschine.

Im Lastmoment M_L steckt die Gravitationskraft

$$F_g = m \cdot g \ , \tag{2.11}$$

wobei sich die Masse m aus dem Eigengewicht m_0 und der äußeren Zusatzlast m_z zusammensetzt. Die Erdbeschleunigung g ist mit $9{,}81\,\mathrm{m/s^2}$ definiert.

Wird die Gravitationskraft mit dem Übersetzungsverhältnis k des Schneckengetriebes auf die Antriebsseite umgerechnet, ergibt sich der erste Teil des Lastmoments zu

$$\begin{aligned} M_{\mathrm{L1}} &= F_g \cdot k \\ M_{L1} &= (m_0 + m_z) \cdot g \cdot k \end{aligned} \tag{2.12}$$

Einen weiteren Teil des Lastmoments bildet die geschwindigkeitsabhängige Reibung (viskose Reibung).

$$M_{L2} = d_{\mathrm{r}} \cdot \omega \tag{2.13}$$

Daraus ergibt sich für das gesamte Lastmoment.

$$\begin{aligned} M_L &= M_{L1} + M_{L2} \\ M_L &= (m_0 + m_z) \cdot g \cdot k + d_r \cdot \omega \end{aligned} \tag{2.14}$$

Eingesetzt in Gleichung 2.10, führt dies zur gesuchten Differenzialgleichung

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = \frac{1}{J} \cdot \Big(k_m \cdot i_a - (m_0 + m_z) \cdot g \cdot k - d_r \cdot \omega \Big) \tag{2.15}$$

Somit wurde folgendes Differentialgleichungssystem gefunden.

$$\begin{aligned} \frac{\mathrm{d}i}{\mathrm{d}t} &= -\frac{R_a}{L_a} \cdot i(t) - \frac{k_m}{L_a} \cdot \omega(t) + \frac{1}{L_a} \cdot U_{KL} \\ \frac{\mathrm{d}\varphi}{\mathrm{d}t} &= \omega \\ \frac{\mathrm{d}\omega}{\mathrm{d}t} &= \frac{1}{J} \cdot \Big(k_m \cdot i_a - (m_0 + m_z) \cdot g \cdot k - d_r \cdot \omega \Big) \end{aligned} \tag{2.16}$$

Mit $x_1 = i(t)$, $x_2 = \varphi(t)$ und $x_3 = \omega(t)$ ergibt sich aus Gleichung 2.16

$$\begin{aligned} \dot{x}_1 &= -\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 + \frac{1}{L_a} \cdot U_{KL} \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= \frac{1}{J} \cdot \Big(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \Big) \end{aligned} \tag{2.17}$$

und liegt somit in der gesuchten Form

$$x_E = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

vor.

$$\boxed{f_E(x_E) = \begin{cases} \dot{x}_1 = -\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 + \frac{1}{L_a} \cdot U_{KL} \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = \frac{1}{J} \cdot \Big(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \Big) \end{cases}} \tag{2.18}$$

2.1.3. Modell für PWM-Signal „Low“

Der Betriebszustand „Low“ entspricht einem offenen Schalter u_{PWM} in Abbildung 2.1 und ist eine Zeitdauer T_A (siehe Abbildung 2.2) aktiv.

2.1.3.1. Ersatzschaltbild

Der offene Schalter u_{PWM} führt dazu, dass die Spannungsquelle U_{KL} nicht mehr in den Stromkreis eingebunden ist. Durch Eigeninduktion treibt die Induktivität L_a den Strom i_a über die Freilaufdiode D_1 weiter, bis entweder der Strom Null wird (siehe Abbildung 2.2) oder der Schalter u_{PWM} wieder geöffnet wird.

Dies führt zu einem Ersatzschaltbild, anhand dessen die gesuchten Differenzialgleichungen aufgestellt werden können (siehe Abbildung 2.4).

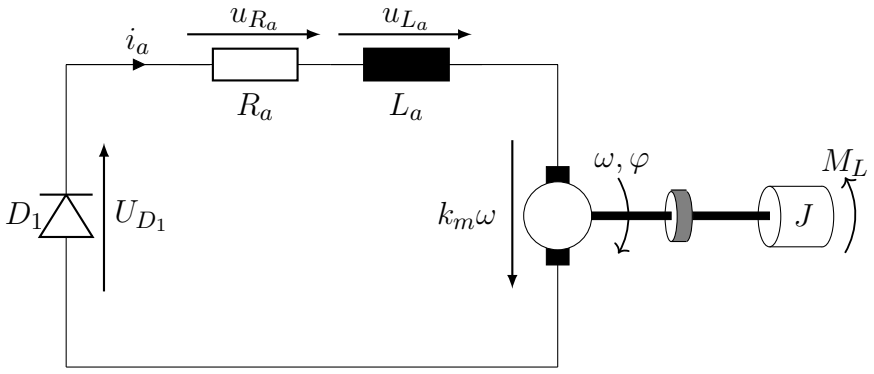


Abbildung 2.4.: Ersatzschaltbild des Gleichstrommotors bei PWM „Low“

2.1.3.2. Aufstellen der Differentialgleichungen

Um die Differentialgleichungen zu ermitteln, gilt die gleiche Vorgehensweise wie in Kapitel 2.1.2.1.

Mit der Maschengleichung

$$0 = u_{L_A}(t) + u_{R_a}(t) + U_D + k_m \cdot \omega(t) \tag{2.19}$$

ergibt sich die erste Differentialgleichung zu

$$L_a \cdot \frac{\mathrm{d}i(t)}{\mathrm{d}t} = -R_a \cdot i(t) - k_m \cdot \omega(t) - U_D \tag{2.20}$$

und somit

$$\frac{\mathrm{d}i(t)}{\mathrm{d}t} = -\frac{R_a}{L_a} \cdot i(t) - \frac{k_m}{L_a} \cdot \omega(t) - \frac{1}{L_a} \cdot U_D \tag{2.21}$$

x_1, x_2 und x_3 wieder eingesetzt ergibt

$$\dot{x}_1 = -\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 - \frac{1}{L_a} \cdot U_D \tag{2.22}$$

Daraus erhält man

$$\begin{aligned} \dot{x}_1 &= -\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 - \frac{1}{L_a} \cdot U_D \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= \frac{1}{J} \cdot \Big(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \Big) \end{aligned} \tag{2.23}$$

und liegt in der gesuchten Form

$$x_A = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boxed{f_A(x_A) = \begin{cases} \dot{x}_1 = -\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 - \frac{1}{L_a} \cdot U_D \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = \frac{1}{J} \cdot \Big(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \Big) \end{cases}} \tag{2.24}$$

vor.

2.1.4. Mittelwertmodell

Aus diesen ermittelten Differentialgleichungen (siehe Gleichungen 2.18 und 2.24) wird durch Einsetzten in Gleichung 2.5 ein Mittelwertmodell erstellt.

$$\begin{aligned} f_1(x) &= \dot{x}_{A_1} + p \cdot (\dot{x}_{E_1} - \dot{x}_{A_1}) \\ f_2(x) &= \dot{x}_{A_2} + p \cdot (\dot{x}_{E_2} - \dot{x}_{A_2}) \\ f_3(x) &= \dot{x}_{A_3} + p \cdot (\dot{x}_{E_3} - \dot{x}_{A_3}) \end{aligned} \tag{2.25}$$

$$\begin{aligned} f_1(x) &= -\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 - \frac{1}{L_a} \cdot U_D + \\ &\quad + p \cdot \left(\cancel{-\frac{R_a}{L_a} \cdot x_1} \cancel{- \frac{k_m}{L_a} \cdot x_3} + \frac{1}{L_a} \cdot U_{KL} + \cancel{\frac{R_a}{L_a} \cdot x_1} \cancel{+ \frac{k_m}{L_a} \cdot x_3} + \frac{1}{L_a} \cdot U_D \right) \\ f_2(x) &= x_3 + p \cdot (\cancel{x_3 - x_3}) \\ f_3(x) &= \frac{1}{J} \cdot \left(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \right) + \\ &\quad + p \cdot \left(\cancel{\frac{1}{J} \cdot \left(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \right) -} \right. \\ &\quad \left. \cancel{- \frac{1}{J} \cdot \left(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \right)} \right) \end{aligned} \tag{2.26}$$

Somit ergibt sich das Mittelwertmodell:

$$\begin{aligned} \dot{x}_1 &= -\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 - \frac{1}{L_a} \cdot U_D + \frac{U_{KL} + U_D}{L_a} \cdot p \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= \frac{1}{J} \cdot \left(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \right) \end{aligned} \tag{2.27}$$

Das Trägheitsmoment J setzt sich aus dem Trägheitsmoment der Maschine J_0 und der äußeren Belastung über das Übersetzungsverhältnis k zusammen

$$J = J_0 + k^2 \cdot (m_0 + m_z) \tag{2.28}$$

daraus ergibt sich das Modell wiederum zu

$$\boxed{\begin{aligned}\dot{x}_1 &= -\frac{R_a}{L_a}\cdot x_1 - \frac{k_m}{L_a}\cdot x_3 - \frac{1}{L_a}\cdot U_D + \frac{U_{KL}+U_D}{L_a}\cdot p \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= \frac{1}{J_0 + k^2\cdot(m_0+m_z)}\cdot\Big(k_m\cdot x_1 - (m_0+m_z)\cdot g\cdot k - d_r\cdot x_3\Big)\end{aligned}} \tag{2.29}$$

2.2. Parameteridentifikation

Um die in Kapitel 2.1.4 ermittelten Differenzialgleichungen für den Entwurf der flachheitsbasierten Vorsteuerung verwenden zu können, muss der Ankerwiderstand R_a, die Ankerinduktivität L_a, die Drehmomentkonstante k_m, das Trägheitsmoment J_0, das Übersetzungsverhältnis des Schneckengetriebes k sowie der Reibungskoeffizient d_r bestimmt werden.

2.2.1. Ankerwiderstand R_a

Um den Ankerwiderstand R_a bestimmen zu können, muss eine kosntante Spannung am Widerstand anliegen, denn damit ist die Ankerinduktivität kurzgeschlossen. Dies kann erreicht werden, indem der „Schalter u_{PWM}“geschlossen wird. Um eine induzierte Spannung $k_m\omega$ zu verhindern, darf sich der Motor nicht bewegen. Bei kleinen Spannungen ($< \pm 3\,\mathrm{V}$) reicht das Reibmoment aus, um den Motor auf einer konstanten Position zu halten. In Abbildung 2.5 wird dies graphisch dargestellt, außerdem ist zu erkennen, dass die Messgeräte für eine spannungsrichtige Messung angeordnet sind. Das bedeutet, dass der gemessene Ankerstrom um den, durch den Innenwiderstand des Voltmeters fließenden, Strom korrigiert werden muss.

Der Widerstand R_a kann wie folgt berechnet werden.

$$R_a = \frac{U_{R_a}}{I_{gemessen} - I_v} \tag{2.30}$$

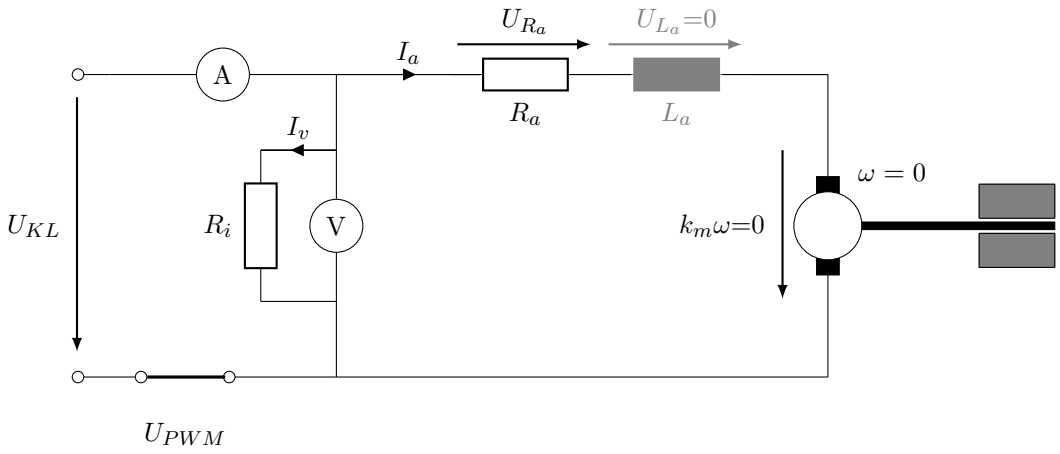


Abbildung 2.5.: Ersatzschaltbild zur Messung des Ankerwiderstandes R_a

Setzt man nun

$$I_v = \frac{U_{R_a}}{R_i} \tag{2.31}$$

in Gleichung 2.30 ein, so wird der Ankerwiderstand R_a durch

$$\boxed{R_a = \frac{U_{R_a}}{I_{gemessen} - \frac{U_{R_a}}{R_i}}} \tag{2.32}$$

bestimmt.

2.2.2. Ankerinduktivität L_a

Um die Ankerinduktivität bestimmen zu können, darf keine Spannung $k_m\omega$ induziert werden. Es wird eine Eingangsspannung u_{KL} zwischen $\pm 3\,\mathrm{V}$ gewählt. Zum Berechnen der Ankerinduktivität L_a wird das in Abbildung 3.16 dargestellte Ersatzschaltbild der Gleichstrommaschine herangezogen.

Es gibt drei in der Praxis häufig verwendete Verfahren zur Bestimmung der Ankerinduktivität:

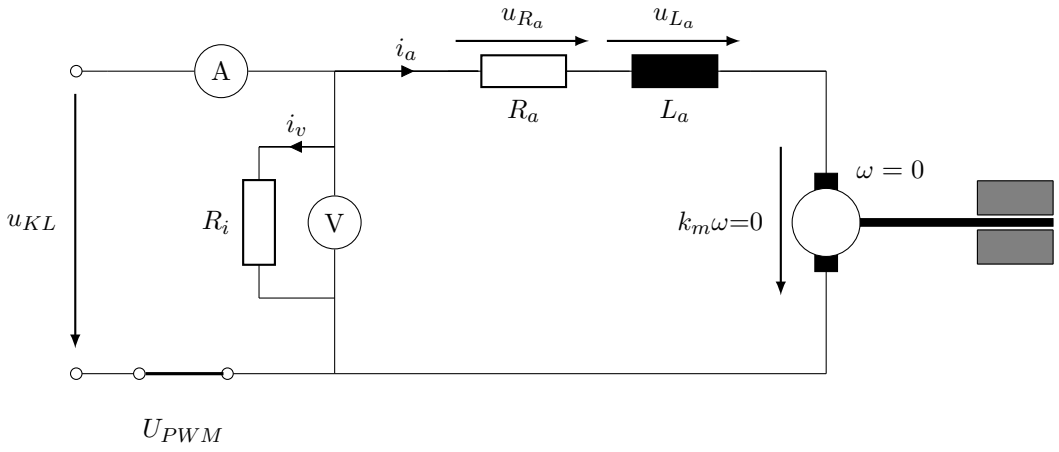


Abbildung 2.6.: Ersatzschaltbild zur Messung der Ankerinduktivität L_a

1. Messen der Anstiegszeit τ bei einem Sprung
2. Bestimmung der Impedanz mittels Strom- und Spannungsmessung
3. Anpassung der Frequenz und Messung der Phasenverschiebung

2.2.2.1. Ermittlung der Ankerinduktivität mittels Messung der Anstiegszeit τ

Um die Ankerinduktivität mittels Anstiegszeit bestimmen zu können, muss als Eingangsspannung ein Sprung $u_{KL} = \sigma(t)$ angelegt werden. Durch Messen der Zeit zwischen dem Anlegen des Sprunges und dem Erreichen von 63 % der angelegten Spannung erhält man die Anstiegszeit τ (siehe Abbildung 2.7).

Die Anstiegszeit einer Induktivität ist definiert als:

$$\tau = \frac{R}{L} \tag{2.33}$$

Durch Umformen und Einsetzen des bekannten Ankerwiderstandes R_a lässt sich die gesuchte Ankerinduktivität L_a mit

$$\boxed{L_a = \frac{R_a}{\tau}} \tag{2.34}$$

bestimmen.

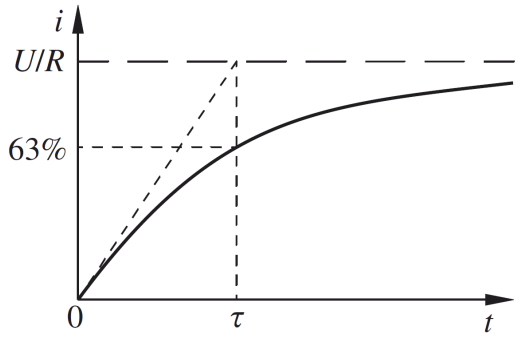


Abbildung 2.7.: Einschaltvorgang einer Spule [Plaßmann und Schulz, 2013]

2.2.2.2. Ermittlung der Ankerinduktivität mittels Strom- und Spannungsmessung

Soll die Ankerinduktivität mittels Strom- und Spannungsmessung erfolgen, so muss am Eingang ein Sinussignal angelegt werden, dessen Amplitude 3 V nicht überschreitet, damit eine Spannungsinduktion im Motor verhindert wird.

Zur Impedanzbestimmung muss der Betrag von $\underline{i}_a$ und $\underline{u}_{KL}$ bestimmt werden. Die Messung erfolgt wieder spannungsrichtig (siehe Abbildung 3.16), weshalb eine Korrektur des gemessenen Stroms vorgenommen werden muss.

Somit ergibt sich $\underline{i}_a$ zu

$$\underline{i}_a = \underline{i}_{gemessen} - \frac{\underline{u}_{KL}}{R_i} \tag{2.35}$$

Die Impedanz $\underline{Z}$ kann mit

$$|\underline{Z}| = \frac{|\underline{u}_{KL}|}{|\underline{i}_a|} \tag{2.36}$$

berechnet werden.

Der Zusammenhang zwischen dem ohm'schen Widerstand $\underline{Z}_R$, dem komplexen Widerstand $\underline{Z}_L$ und der Gesamtimpedanz $\underline{Z}$ soll in Abbildung 2.8 gezeigt werden.

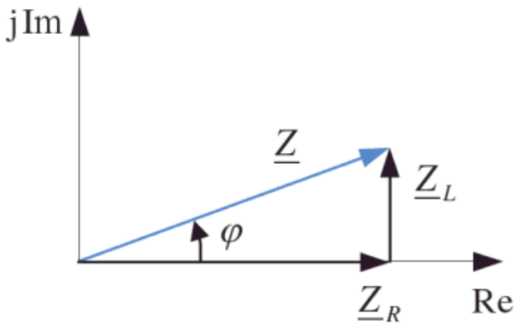


Abbildung 2.8.: Zugehöriges Zeigerdiagramm [Albach, 2011]

Der dazugehörige mathematische Zusammenhang lautet:

$$|\underline{Z}| = \sqrt{\underline{Z}_R^2 + \underline{Z}_L^2} \tag{2.37}$$

Mit

$$\begin{aligned} |\underline{Z}_R| &= R_a \\ |\underline{Z}_L| &= \omega \cdot L_a \end{aligned} \tag{2.38}$$

ergibt sich somit die gesuchte Induktivität.

$$L_a = \frac{\sqrt{|\underline{Z}|^2 - R_a^2}}{\omega} \tag{2.39}$$

Durch Einsetzen der Gesamtimpedanz $|\underline{Z}|$ kann die Ankerinduktivität mit

$$\boxed{L_a = \frac{\sqrt{\frac{|\underline{u}_a|^2}{|\underline{i}_a|^2} - R_a^2}}{\omega}} \tag{2.40}$$

berechnet werden.

2.2.2.3. Ermittlung der Ankerinduktivität mittels Anpassung der Phasenverschiebung φ

Auch bei diesem Verfahren ist es notwendig, dass die Amplitude der Klemmspannung u_{KL} 3 V nicht überschreitet. Bei dieser Art der Bestimmung der Ankerinduktivität wird die Frequenz der Versorgungsspannung so lange verändert, bis sich eine Phasenverschiebung φ von 45° einstellt. Durch die gemeinsame Darstellung von Strom und Spannung auf einem Oszilloskop wird die Anpassung der Phasenverschiebung erheblich vereinfacht. Bei einer Phasenverschiebung von 45° entspricht der Betrag der Impedanz $|\underline{Z}_L|$ dem Betrag der Impedanz $|\underline{Z}_R|$ (siehe Abbildung 2.8 und Gleichungen 2.38). Dies wird aus dem Zusammenhang

$$\begin{aligned} |\underline{Z}_L| &= |\underline{Z}| \cdot cos(\varphi) \\ |\underline{Z}_R| &= |\underline{Z}| \cdot sin(\varphi) \end{aligned} \tag{2.41}$$

mit

$$cos(45°) = sin(45°) \tag{2.42}$$

ersichtlich. Somit gilt

$$R_a = \omega \cdot L_a \tag{2.43}$$

und die gesuchte Ankerimpedanz L_a kann mit

$$\boxed{L_a = \frac{R_a}{\omega}} \tag{2.44}$$

berechnet werden.

2.2.3. Drehmomentkonstante k_m

Zur Bestimmung der Drehmomentkonstante k_m wird das System im eingeschwungenen Zustand betrachtet. Das bedeutet, dass der Ankerstrom konstant bleibt, wodurch die Ableitung zu Null wird. Außerdem entspricht konstanter Strom bei einer Spule einem Kurzschluss, womit der Spannungsabfall an L_a zu

Null wird (siehe Abbildung 2.9). Hierbei muss darauf geachtet werden, dass die Klemmspannung U_{KL} größer als 3V ist, damit sich der Motor dreht und Spannung induziert wird. Andernfalls wird das Produkt von Drehmomentkonstante und Drehgeschwindigkeit zu Null und k_m ist nicht mehr bestimmbar.

Aus Gleichung 2.29 wird ersichtlich, dass bei konstantem Strom und im eingeschwungenen Zustand auch die Änderung der Winkelgeschwindigkeit ω zu null, also konstant wird.

Durch Umformung der zweiten Differenzialgeleichung erhält man

$$\begin{aligned} \dot{\varphi} &= \omega \\ \varphi &= \int_0^t \omega \cdot d\tau \\ \varphi &= \omega \cdot \int_0^t 1 \cdot d\tau \\ \varphi &= \omega \cdot t \\ \omega &= \frac{\varphi}{t} \end{aligned} \tag{2.45}$$

Es kann zwar keine absolute Position, jedoch eine relative bestimmt werden.

$$\begin{aligned} \omega &= \frac{\Delta\varphi}{\Delta t} \\ \boxed{\omega &= \frac{\varphi(t_2) - \varphi(t_1)}{t_2 - t_1}} \end{aligned} \tag{2.46}$$

Wird nun die erste Differenzialgleichung mit $p = 1$ betrachtet (Tastverhältnis ist eins bei einer Klemmspannung von $U_{KL} = 10\,\mathrm{V}$)

$$\frac{\mathrm{d}I_a}{\mathrm{d}t} = 0 = -\frac{R_a}{L_a} \cdot I_a - \frac{k_m}{L_a} \cdot \omega + \frac{1}{L_a} \cdot U_{KL} \tag{2.47}$$

wird ersichtlich, dass hier drei unbekannte Parameter auftauchen I_a, U_a und ω. Deshalb müssen I_a, U_a, φ und die Zeit t gemessen werden.

Damit ergibt sich die Drehmomentkonstante k_m zu

$$k_m = (U_{KL} - R_a \cdot I_a) \cdot \frac{1}{\omega} \tag{2.48}$$

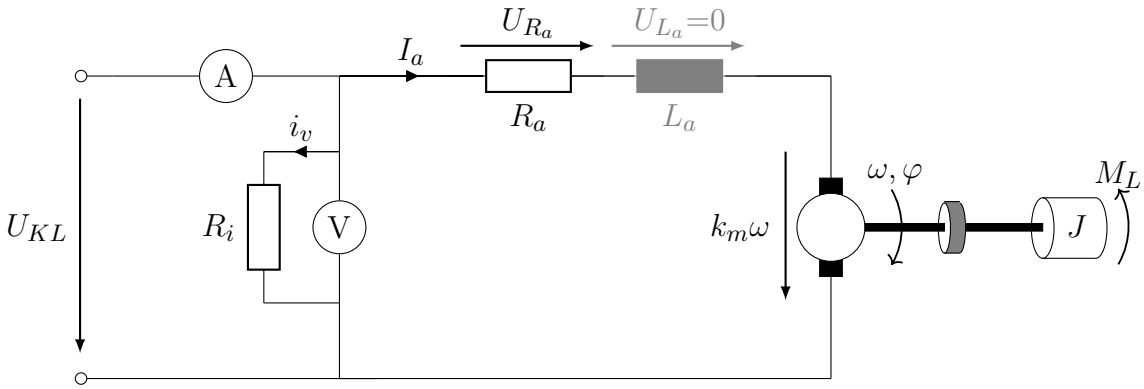


Abbildung 2.9.: Ersatzschaltbild zur Messung der Drehmomentkonstante k_m

Die Messfehler durch die spannungsrichtige Messung (siehe Abbildung 2.9) müssen wieder berücksichtigt werden. Somit ergibt sich die gesuchte Drehmomentkonstante k_m zu

$$\boxed{k_m = \left(U_{KL} - R_a \cdot \left(I_{gemessen} - \frac{U_{KL}}{R_i}\right)\right) \cdot \frac{t_2 - t_1}{\varphi(t_2) - \varphi(t_1)}} \tag{2.49}$$

2.2.4. Trägheitsmoment J_0, Übersetzungsverhältnis k und Reibungsparameter d_r

Das direkte Messen der Größen (Trägheitsmoment J_0, Übersetzungsverhältnis k und Reibungsparameter d_r) ist eine sehr schwierige Aufgabe, deshalb wurde dies hier mittels eines Optimierungsalgorithmus gelöst. Bei dieser Methode müssen der Eingangsverlauf (also der Verlauf des Tastverhältnisses p) und der Verlauf der Ausgangsgröße (also der Winkel φ) gemessen werden.

Es werden die Parameter J_0, k und d_r geschätzt, der Ausgang φ berechnet und die Abweichung betrachtet. Im nächsten Schritt werden die gesuchten

Parameter verändert und untersucht, ob die Abweichung kleiner wird. Ist dies nicht der Fall, so werden die Parameter in eine andere Richtung verändert. Wird die Abweichung jedoch kleiner, so werden die Parameter zum neuen Ausgangspunkt. Dies wird so lange wiederholt, bis in keiner Richtung eine Verkleinerung der Abweichung erzielt werden kann, also bis ein Minimum des Fehlers erreicht wird.

Dabei kann es passieren, dass der Algorithmus in lokalen Minima „gefangen“ wird. Um dies zu vermeiden, wird, nachdem ein Optimum gefunden wurde, ein Parameter mit einer größeren Schrittweite verändert und die Optimierung beginnt von neuem. Dies wird einige Male wiederholt, womit die Wahrscheinlichkeit, ein globales Minimum zu finden, erhöht wird.

Hat man ein globales Minimum gefunden, entsprechen die verwendeten Größen näherungsweise den realen.

2.3. Reglerentwurf

Der bisher von der Firma Logicdata implementierte Regler funktioniert in der Regel sehr gut, nur bei sehr unterschiedlichen Motoren kann es zu kurzen, pendelnden Schwingungen kommen. Die Ursache dafür ist ein Auseinanderlaufen der Antriebe (Δy), die durch ein Überschwingen eines Reglers zustande kommen können. Wie in Kapitel 2.3.1.2 beschrieben, kann der Motor immer nur in eine Richtung geregelt werden. „Schießt“ der Motor also über das Ziel hinaus, kann er nicht zurückfahren. Fährt der Tisch danach weiter, muss der Positionsunterschied ausgeglichen werden, was sich durch ein kurzes „Hüpfen“ äußern kann. Dies soll in Abbildung 2.10 verdeutlicht werden.

Beide Tischbeine starten zum Zeitpunkt t_1 und fahren bis zum Zeitpunkt t_2 (rot) nach oben. Es ist zu erkennen, dass Tischbein 2 erst nach der gewünschten Position y_{soll} zum Stillstand kommt. Es ist also zu einem Positionsunterschied der beiden Tischbeine Δy gekommen, welches beim erneuten Anfahren (blau) ausgeglichen werden muss und sich in Form eines „Hüpfens“ (starke Beschleunigung) äußern kann.

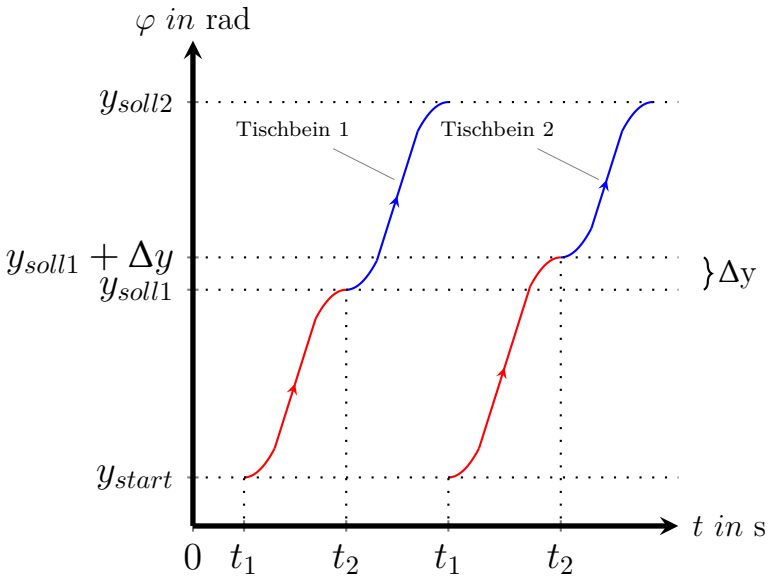


Abbildung 2.10.: Verlauf von zwei Tischbeinen mit einem Überschwingen

2.3.1. Bestehendes Regelkonzept

Die Firma Logicdata verwendet einen PI-Regler für die Positionsregelung der Antriebe, welcher das Tastverhältnis des PWM-Signals als Stellgröße für den Motor liefert. Die daraus resultierende Spannung bewirkt eine Bewegung des Motors, welche mittels Hallsensoren detektiert wird. Aus diesen Hallimpulsen wird mittels Encoder der aktuelle Winkel, also die Position, berechnet, dieser wird vom Sollwinkel abgezogen. Der daraus erhaltene Regelfehler wird wieder vom PI-Regler zur Berechnung der Stellgröße verwendet. Hierbei ist zu beachten, dass bei Überschreiten der Stellgröße u von 90 % der maximalen Auslastung des Motors die Sollgröße angepasst wird. Weiters ist eine Stromrichtungsänderung, also die Änderung der Bewegungsrichtung, während der Regelung nicht möglich.

Eine Veranschaulichung dieser Zusammenhänge ist in Abbildung 2.11 dargestellt.

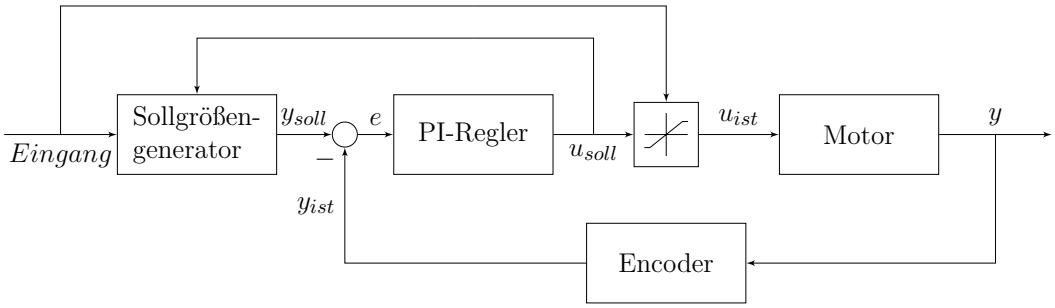


Abbildung 2.11.: Blockdiagramm des bestehenden Regelalgorithmus

2.3.1.1. Implementierter PI-Regler

Der bestehende PI-Regler hat die Form

$$R(q) = \frac{k_P}{\frac{q}{\omega_1} + 1} + k_I \frac{1 - q\frac{T_a}{2}}{q\left(\frac{q}{\omega_1} + 1\right)} \tag{2.50}$$

oder im diskreten Zeitbereich

$$\begin{aligned} u_k = \frac{4}{2 + T_a\omega_1} \cdot u_{k-1} - \frac{2 - T_a\omega_1}{2 + T_a\omega_1} \cdot u_{k-2} + \frac{T_a\omega_1 k_p}{2 + T_a\omega_1} \cdot e_k + \\ + \frac{T_a^2\omega_1 k_I}{2 + T_a\omega_1} \cdot e_{k-1} + \frac{T_a^2\omega_1 k_I - T_a\omega_1 k_P}{2 + T_a\omega_1} \cdot e_{k-2} \end{aligned} \tag{2.51}$$

2.3.1.2. Stellgrößenbeschränkungen

Um diese Regelung implementieren zu können, müssen die Stellgrößenbeschränkungen berücksichtigt werden. Eine dieser Beschränkungen betrifft die Motorauslastung, also die maximale Geschwindigkeit des Motors. Diese ist erreicht, sobald die gesamte Spannung U_{KL} am Motor anliegt, also das Tastverhältnis p der Pulsweitenmodluation gleich 1 ist. Die Motorauslastung wird auf 90 % begrenzt ($p = 0.9$), um eine Stellgröße p von über 1 zu vermeiden und somit ein Auseinanderlaufen der Antriebe einzuschränken (siehe Abbildung 2.13).

Weiters kann sich der Motor immer nur in eine Richtung bewegen, das heißt, erst wenn der Motor still steht, können die Klemmen 1 & 2 (wie in Abbildung 2.12 dargestellt) vertauscht werden, wodurch sich der Motor in die entgegengesetzte Richtung zu drehen beginnt. Das bedeutet, dass eine Richtungsänderung durch den Regler nicht möglich ist! In Abbildung 2.11 sind diese Zusammenhänge nochmals dargestellt.

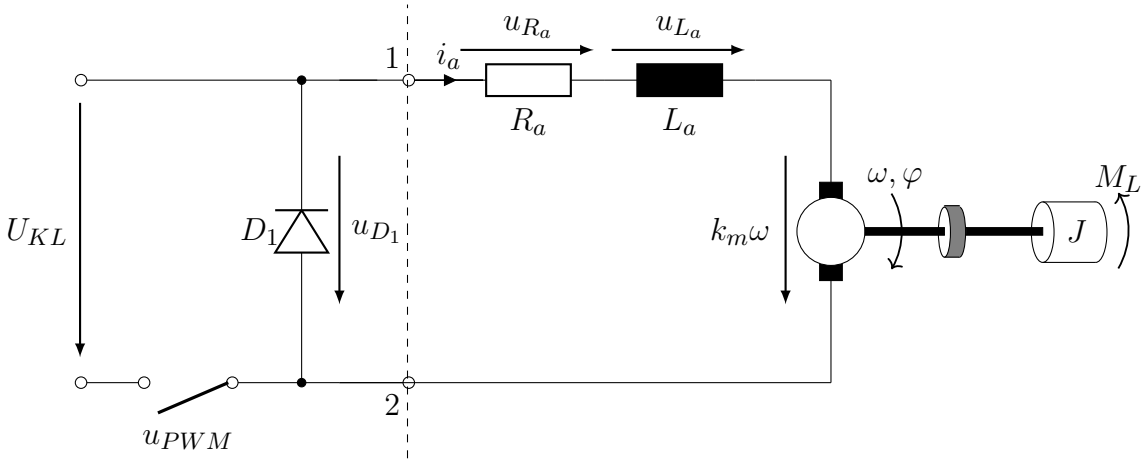


Abbildung 2.12.: Allgemeines Ersatzschaltbild des Gleichstrommotors

2.3.1.3. Entwurf des Sollgrößenverlaufs

Das Eingangssignal „Eingang“ kann drei Zustände annehmen, „Hinauffahren“, „Hinunterfahren“ und „Null“. Wechselt das Eingangssignal von „Null“ auf „Hinauffahren“ (siehe Abbildung 2.13), so soll der Motor innerhalb von 0,6 s auf die maximale Geschwindigkeit beschleunigen (rot). Erreicht die Stellgröße u_{soll} einen höheren Wert als 90 % der maximalen Auslastung, so wird die Höchstgeschwindigkeit verringert. Somit ist gewährleistet, dass kein Positionsunterschied der Antriebe durch Überlastung entsteht. Nach 0,6 s hat der Antrieb seine maximale Geschwindigkeit erreicht und bewegt sich konstant mit dieser (grün), bis das Eingangssignal wieder zurück auf „Null“ geht. Ist dies der Fall, so wird der Motor innerhalb von 0,6 s bis zum Stillstand abgebremst (**blau**). In Abbildung 2.13 ist mit der durchgehenden Linie die gewünschte Geschwindigkeit ω und mit der gestrichelten Linie die dazugehörige Sollposition φ dargestellt.

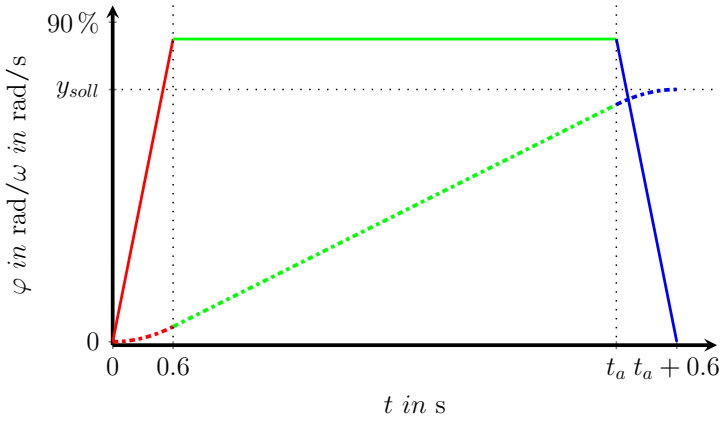


Abbildung 2.13.: Sollgrößenverlauf

2.3.2. Adaption der Sollgröße

Um dem Problem des Überschwingens entgegenzuwirken, kann zu der Sollposition die Differenz zwischen Soll- und Istposition addiert werden. Fährt ein Motor über die Sollposition, wird diese angepasst und der zweite Motor fährt nach. Dadurch wird verhindert, dass ein zu großer Positionsunterschied der Antriebe am Ende einer Bewegung entsteht (Δy). Siehe dazu Abbildung 2.14.

2.3.3. Anpassen der Abbrems- und Beschleunigungszeit

Ein weiterer Ansatz zum Lösen des Problems besteht darin, bei einem bestehenden Positionsunterschied (Δy) der Antriebe nicht zu abrupt zu beschleunigen. Wird also ein Positionsunterschied erkannt, und der Motor würde zu schnell versuchen diesen auszugleichen (dies äußert sich in Form eines „Hüpfers“ (siehe Abbildung 2.10)), so kann man eingreifen, und die verfügbare Zeit zum Erreichen der Maximalgeschwindigkeit (siehe Abbildung 2.13) erhöhen, was eine Verringerung der Steigung der Geschwindigkeit (Verringerung der Beschleunigung) zur Folge hat (siehe Abbildung 2.15). Ein Überschwingen kann auch zustande kommen, wenn der Motor nicht schnell genug abbremsen kann. Somit kann durch eine Vergrößerung der Zeitspanne, welche zum Abbremsen zur Verfügung steht, ein Auseinanderlaufen der Antriebe verhindert werden.

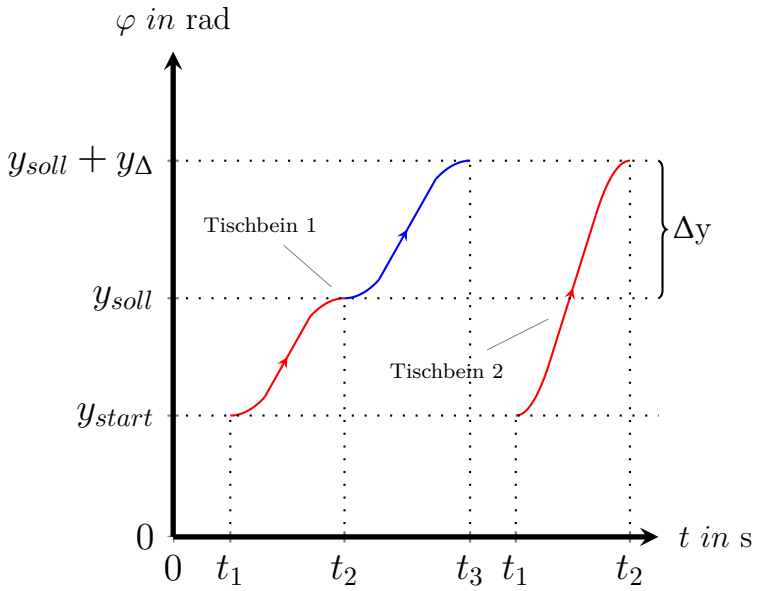


Abbildung 2.14.: Korrektur der Sollgröße

2.3.4. Flachheitsbasierte Vorsteuerung mit Regler zur Stabilisierung

Eine ideale Vorsteuerung ist die Inverse des zu regelnden Modells, womit eine Regelung überflüssig wäre. Da jedoch Modellunsicherheiten vorhanden sind (in diesem Fall sollen außerdem verschiedene Motoren des gleichen Typs geregelt werden) kann auf die Regelung nicht verzichtet werden. Vereinfacht soll dieses Konzept in Abbildung 2.16 dargestellt werden. Die Abweichung der Stellgröße u_V der Vorsteuerung wird durch die Stellgröße u_R des Reglers korrigiert und damit die tatsächliche Stellgröße u erzeugt, welche dem Tastverhältnis p entspricht.

Um die Methode des flachheitsbasierten Vorsteuerungsentwurfs anwenden zu können, muss das Modell ein flaches System sein.

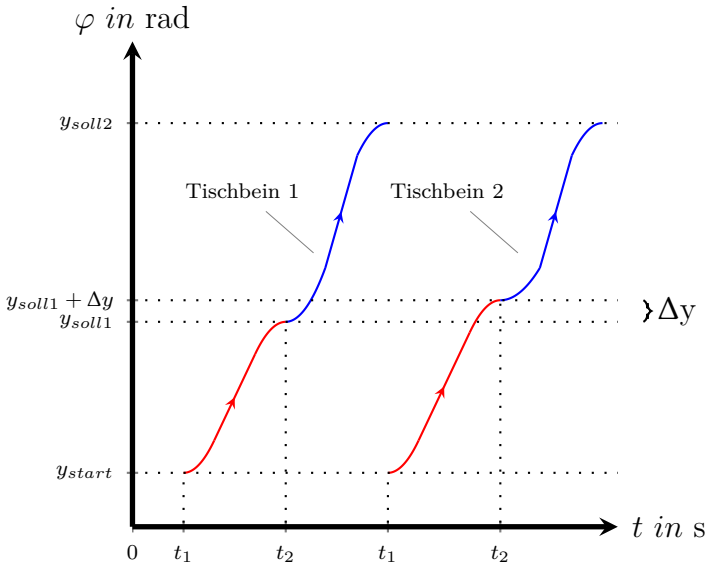


Abbildung 2.15.: Korrektur der Anstiegszeit

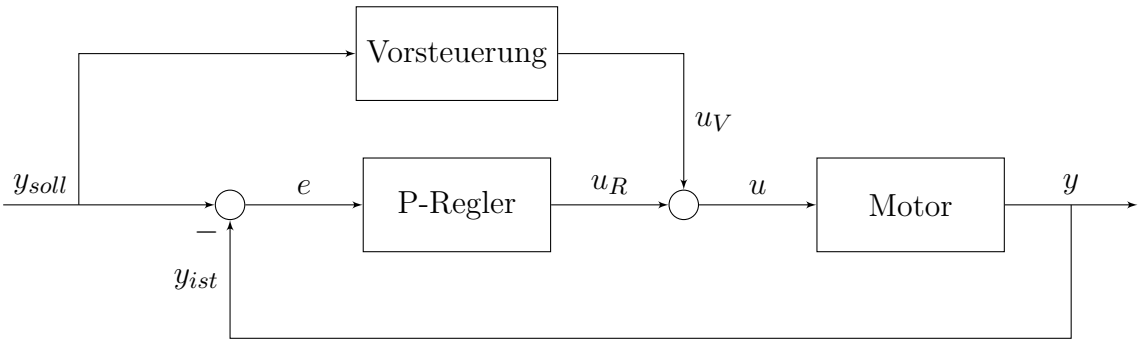


Abbildung 2.16.: Blockschaltbild der Flachheitsbasierten Vorsteuerung inklusive Regelung zur Stabilisierung

Definition.

$$„\dot{x} = f(x,u), \qquad x(0) = x_0 \in \mathbb{R}^n, \qquad u \in \mathbb{R}^m \tag{2.52}$$

Ein nichtlineares System (2.52) heißt (differentiell) [1] flach, wenn es einen fiktiven Ausgang

$$\begin{aligned} y &= \phi\left(x, u_1, ..., \overset{(\alpha_1)}{u_1}, ..., u_m, ..., \overset{(\alpha_m)}{u_m}\right) = \\ &= \phi\left(x, u, \dot{u}, ..., \overset{(\alpha)}{u}\right) \end{aligned} \tag{2.53}$$

mit dim (y) = dim (u) gibt, der die folgenden Bedingungen erfüllt. Die Zustände $x_i, i = 1, ..., n$ und die Eingänge $u_i, i = 1, ..., m$ können als Funktionen der Komponenten $y_i, i = 1, ..., m$ und einer endlichen Anzahl ihrer Zeitableitungen $\overset{(k)}{y_i}$ eindeutig ausgedrückt werden:

$$\begin{aligned} x &= \psi\left(y_1, ..., \overset{(\beta_1)}{y_1}, ...y_m, ..., \overset{(\beta_m)}{y_m}\right) = \\ &= \psi_1\left(y, \dot{y}, ..., \overset{(\beta)}{y}\right), \end{aligned} \tag{2.54}$$

$$\begin{aligned} u &= \psi_2\left(y_1, ..., \overset{(\beta_1+1)}{y_1}, ..., y_m, ..., \overset{(\beta_m)}{y_m}\right) = \\ &= \psi_2\left(y, \dot{y}, ..., \overset{(\beta+1)}{y}\right). \end{aligned} \tag{2.55}$$

Sind diese Bedingungen zumindest lokal erfüllt, so heißt der fiktive Ausgang „flacher Ausgang". Wegen dim (y) = dim (u) und (2.55) sind die Komponenten von y differentiell unabhängig, d.h. sie erfüllen keine Differentialgleichungen der Form

$$\varphi\left(y, \dot{y}, ..., \overset{(\gamma)}{y}\right) = 0 \tag{2.56}$$

[1] *Der Begriff differentiell wird verwendet, da aus den Komponenten von y durch Differentiation, d.h. ohne Integration von Differentialgleichungen, alle Zustände und Eingänge sowie deren Zeitableitungen bestimmt werden können.*

> *Diese Bedingung wird häufig anstelle der Forderung dim (y) = dim (u) in der Definition flacher Systeme benutzt, ihr Nachweis ist jedoch schwieriger. Aufgrund der differentiellen Unabhängigkeit sind die Trajektorien für die Komponenten von y voneinander unabhängig.“*

[Rothfuß, Rudolph und Zeitz, 1997]

Aus den Gleichungen 2.60 ist zu erkennen, dass die Systemgrößen $\underline{x}$ durch den Ausgang y und dessen Ableitungen ausgedrückt werden können. In Gleichung 2.63 wird ersichtlich, dass auch die Eingangsspannung p durch diese bestimmt werden kann. Da $dim(u = p) = 1 = dim(y = \varphi)$ ebenfalls gegeben ist, handelt es sich in diesem Fall um ein flaches System. Aus den Gleichungen 2.57 und 2.58 ist erkennbar, dass der relative Grad r der Systemordnung n entspricht, somit ist y auch ein flacher Ausgang.

Durch die Bestimmung des relativen Grades r erhält man vier Gleichungen mit drei unbekannten Systemgrößen $\underline{x}$ der Ausgangsgröße y und deren Ableitungen und die gesuchte Stellgröße p. Wird y_{soll} als Eingangsgröße verwendet, kann damit und mit deren Ableitungen die gesuchte Stellgröße berechnet werden.

2.3.4.1. Bestimmung des relativen Grades

Die Ausgangsgröße y, die geregelt wird, entspricht dem Winkel φ. Um nun den relativen Grad r zu bestimmen, wird der Ausgang y so lange differenziert, bis die Eingangsgröße, also das Tastverhältnis p, in der Lösung der Differentialgleichung vorkommt. Der relative Grad r ist dann gleich der erforderlichen Anzahl der Ableitungen.

$$y = x_2$$

$$\dot{y} = \dot{x}_2$$

$$\dot{y} = x_3$$

$$\ddot{y} = \dot{x}_2$$

$$\ddot{y} = \frac{1}{J_0 + k^2 \cdot (m_0 + m_z)} \cdot \left(k_m \cdot x_1 - (m_0 + m_z) \cdot g \cdot k - d_r \cdot x_3 \right)$$

$$\dddot{y} = \frac{1}{J_0 + k^2 \cdot (m_0 + m_z)} \cdot \left(k_m \cdot \dot{x}_1 - d_r \cdot \dot{x}_3 \right)$$

$$\dddot{y} = \frac{1}{J_0 + k^2 \cdot (m_0 + m_z)} \cdot \left(k_m \cdot \left(-\frac{R_a}{L_a} \cdot x_1 - \frac{k_m}{L_a} \cdot x_3 - \frac{1}{L_a} \cdot U_D + \frac{U_{KL} + U_D}{L_a} \cdot p \right) + \right.$$

$$\left. + d_r \cdot \left(\frac{1}{J_0 + k^2 \cdot (m_0 + m_z)} \cdot \left(- k_m \cdot x_1 + (m_0 + m_z) \cdot g \cdot k + d_r \cdot x_3 \right) \right) \right) \tag{2.57}$$

$$\boxed{\begin{aligned}\dddot{y} = \frac{1}{J_0 + k^2 \cdot (m_0 + m_z)} \cdot \Bigg[& \left(-\frac{k_m \cdot R_a}{L_a} - \frac{d_r \cdot k_m}{J_0 + k^2 \cdot (m_0 + m_z)} \right) \cdot x_1 + \\ & + \left(-\frac{k_m^2}{L_a} + \frac{d_r^2}{J_0 + k^2 \cdot (m_0 + m_z)} \right) \cdot x_3 + \\ & + \left(-\frac{k_m \cdot U_D}{L_a} + \frac{(m_0 + m_z) \cdot g \cdot k \cdot d_r}{J_0 + k^2 \cdot (m_0 + m_z)} \right) \\ & + \left(\frac{km \cdot (U_{KL} + U_D)}{L_a} \right) \cdot p \Bigg]\end{aligned}} \tag{2.58}$$

Somit ergibt sich der relative Grad r zu

$$\boxed{\boxed{r = 3}} \tag{2.59}$$

2.3.4.2. Berechnung der flachheitsbasierten Vorsteuerung

In Gleichung 2.58 liegt die r-te Ableitung des Ausgangs y in Abhängigkeit der Systemgrößen $\underline{x}$ und des Eingangs u vor. Die dabei unbekannten Systemgrößen können durch geeignete Umformung der Gleichungen 2.57 berechnet werden. Sie ergeben sich somit zu

$$\boxed{\begin{aligned} x_1 &= \frac{\ddot{y} \left(J_0 + k^2 \cdot (m_0 + m_z) \right) + (m_0 + m_z) \cdot g \cdot k + d_r \cdot \dot{y}}{k_m} \\ x_2 &= y \\ x_3 &= \dot{y} \end{aligned}} \tag{2.60}$$

Durch Einsetzten der Gleichungen 2.60 in Gleichung 2.58 kann p in Abhängigkeit von y und dessen Ableitungen berechnet werden.

$$
\begin{aligned}
p = & \left[\ddot{y} \cdot \left(J_0 + k^2 \cdot (m_0 + m_z) \right) + \left(\frac{R_a}{L_a} + \frac{d_r}{J_0 + k^2 \cdot (m_0 + m_z)} \right) \cdot \right. \\
& \cdot \left(\ddot{y} \left(J_0 + k^2 \cdot (m_0 + m_z) \right) + (m_0 + m_z) \cdot g \cdot k + d_r \cdot \dot{y} \right) + \\
& \left. + \left(\frac{k_m^2}{L_a} - \frac{d_r^2}{J_0 + k^2 \cdot (m_0 + m_z)} \right) \cdot \dot{y} + \left(\frac{k_m \cdot U_D}{L_a} - \frac{(m_0 + m_z) \cdot g \cdot k \cdot d_r}{J_0 + k^2 \cdot (m_0 + m_z)} \right) \right] \cdot \\
& \cdot \frac{L_a}{k_m \cdot (U_{KL} + U_D)}
\end{aligned}
\tag{2.61}
$$

Durch Vereinfachung erhält man

$$
\begin{aligned}
p = & \left[\ddot{y} \cdot L_a \cdot \left(J_0 + k^2 \cdot (m_0 + m_z) \right) + \right. \\
& + \ddot{y} \cdot \left(R_a \cdot \left(J_0 + k^2 \cdot (m_0 + m_z) \right) + d_r \cdot L_a \right) + \\
& + \dot{y} \left(R_a \cdot d_r + \cancel{\frac{d_r^2 \cdot L_a}{J_0 + k^2 \cdot (m_0 + m_z)}} + k_m^2 - \cancel{\frac{d_r^2 \cdot L_a}{J_0 + k^2 \cdot (m_0 + m_z)}} \right) + \\
& + R_a \cdot (m_0 + m_z) \cdot g \cdot k + \cancel{\frac{(m_0 + m_z) \cdot g \cdot k \cdot d_r}{J_0 + k^2 \cdot (m_0 + m_z)}} \cdot L_a + \\
& \left. + k_m \cdot U_D - \cancel{\frac{(m_0 + m_z) \cdot g \cdot k \cdot d_r}{J_0 + k^2 \cdot (m_0 + m_z)}} \cdot L_a \right] \cdot \frac{1}{k_m \cdot (U_{KL} + U_D)}
\end{aligned}
\tag{2.62}
$$

damit ergibt sich p zu

$$p = \Bigg[\dddot{y}_a \cdot L_a \cdot \Big(J_0 + k^2 \cdot (m_0 + m_z)\Big) + \\ + \ddot{y} \cdot \Big(R_a \cdot \Big(J_0 + k^2 \cdot (m_0 + m_z)\Big) + d_r \cdot L_a\Big) + \\ + \dot{y} \left(R_a \cdot d_r + k_m^2\right) + \\ + R_a \cdot (m_0 + m_z) \cdot g \cdot k + k_m \cdot U_D \Bigg] \cdot \frac{1}{k_m \cdot (U_{KL} + U_D)} \tag{2.63}$$

Somit kann durch Erstellen eines geeigneten Verlaufs des gewünschten Ausgangs y_{soll} und durch Bildung dessen Ableitungen p problemlos berechnet werden. Es ist bei der Wahl des Ausgangsverlaufs wichtig, darauf zu achten, dass dieser n mal, also 3 mal, nach der Zeit abgeleitet werden kann. Da y der Position, also dem Winkel φ entspricht, kann auch die Ableitung, und somit die Winkelgeschwindigkeit ω vorgeben werden. Diese muss also zweimal differenzierbar und einmal integrierbar sein. Der bisherige Verlauf der Sollgröße (siehe Abbildung 2.13) kann nicht verwendet werden, da die Übergänge der Geraden in der zweiten Ableitung eine Unstetigkeitsstelle aufweisen. Es wird die Beschleunigung sowie der Abbremsvorgang als Arbeitspunktwechsel betrachtet. Hierbei muss innerhalb der Zeit T (0,6 s) ein Übergang von z_0 (Stillstand) auf z_T (maximale Geschwindigkeit) erreicht werden (siehe Abbildung 2.17). Die Wahl von

$$z_d(t) = z_T \cdot \left(3 \cdot \left(\frac{t}{T}\right)^2 - 2 \cdot \left(\frac{t}{T}\right)^3\right) \tag{2.64}$$

erfüllt die Bedingung, dass $z_d(t)$ zum Zeitpunkt $t = 0$ Null ist und zum Zeitpunkt $t = T$ die maximale Geschwindigkeit z_T erreicht wird. Außerdem werden dadurch, dass die nächsten zwei Ableitungen zum Zeitpunkt $t = 0$ sowie zum Zeitpunkt $t = T$ Null werden eine weitere Bedingung erfüllt. Wird dieses Polynom auch beim Abbremsen verwendet ergibt sich der gewünschte Verlauf der Geschwindigkeit wie in Abbildung 2.18b gezeigt wird.

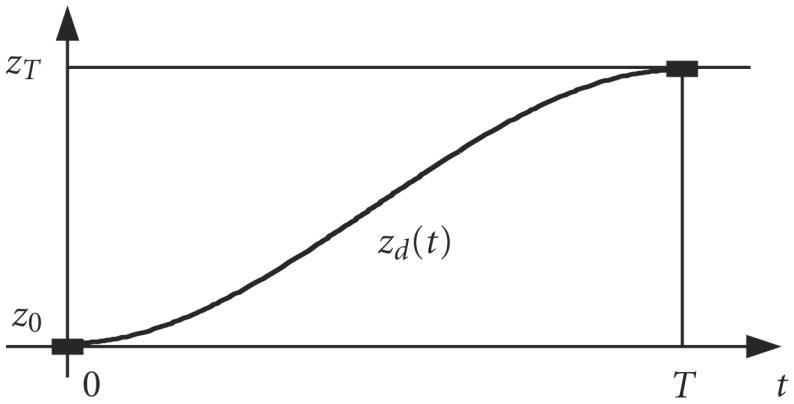
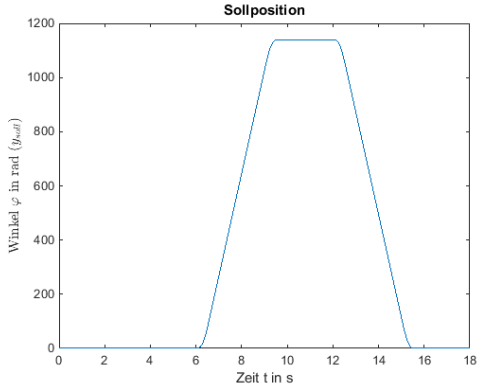


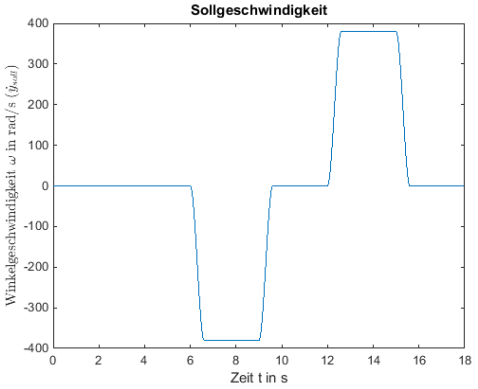
Abbildung 2.17.: Arbeitspunktwechsel [Rothfuß, Rudolph und Zeitz, 1997]

2.3.4.3. Entwurf der Regelung

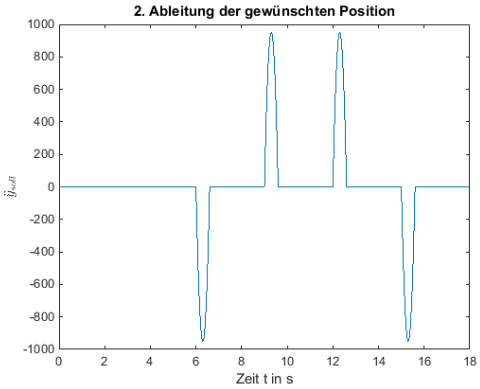
Der Vorteil einer zusätzlichen Vorsteuerung bei einer Regelung besteht darin, dass die Vorsteuerung den Großteil der Stellgröße erzeugt, somit muss die Regelung lediglich die Abweichungen korrigieren. Dafür reicht bereits ein P-Regler aus.



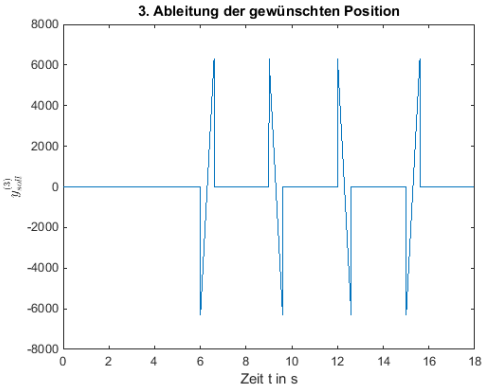
(a) Verlauf der gewünschten Position



(b) Verlauf der gewünschten Geschwindigkeit



(c) 2. Ableitung der gewünschten Position



(d) 3. Ableitung der gewünschten Position

Abbildung 2.18.: Entwurf des Verlaufs der gewünschten Position und die nächsten drei Ableitungen der Position

3. Praktische Durchführung

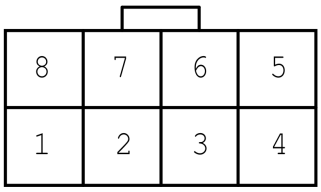
3.1. Testen des bisher verwendeten Regelalgorithmus mittels Arduino Due

Zu Beginn soll die von der Firma Logicdata implementierte Regelung mit Simulink und Arduino Due aufgebaut und getestet werden.

3.1.1. Beschaltung des Motors

Die verwendeten Antriebe haben folgende Anschlüsse:

1. Eingangsspannung für Motor Minus
2. Masse
3. 5V-Versorgung für die Hallsensoren
4. Hallsensorsignal Position 1
5. Eingangsspannung für Motor Plus
6. Masse
7. Masse
8. Hallsensorsignal Position 2



Durch die Spannungsdifferenz zwischen „Eingangsspannung für Motor Plus“ und „Eingangsspannung für Motor Minus“ kann die Geschwindigkeit der Antriebe eingestellt werden. Um an den Ausgängen der Hallsensoren Impulse messen zu können, muss ein Pull-up-Widerstand verwendet werden, da es es sich um Open-Collector-Ausgänge handelt. Diese Pull-up-Widerstände müssen mit einer 5 V-Versorgung verbunden werden, also die gleiche Spannung, mit der die Hallsensoren versorgt werden. In Abbildung 3.3 stellen R_{11} und R_{21} die Pull-up Widerstände dar und werden mit $R_{11} = R_{21} = 4{,}7\,\mathrm{k\Omega}$ dimensioniert.

3.1.2. Beschaltung von Arduino Due

Nachdem das Arduino Due Board mit dem Computer verbunden ist, müssen die benötigten Treiber und Zusatzpakete für Simulink installiert werden (siehe Anhang A). In Anhang A.3 ist außerdem die Pinbelegung des Arduino Due zu finden.

3.1.2.1. Spannungsgenerierung für die Antriebe mittels Arduino Due und Motor Driver Shield

Zur Steuerung der Eingangsspannungen der Motoren wird das Pololu Dual VNH5019 Motor Driver Shield für Arduino verwendet [Pololu, o.D.].

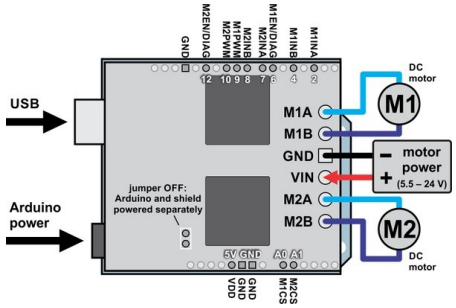


Abbildung 3.1.: Pololu Dual VNH5019 Motor Driver Shield für Arduino [Pololu, o.D.]

Zwischen den Klemmen GND und VIN wird die maximale Spannung von 24 V angeschlossen, somit kann die Ausgangsspannung des Motor Driver Shields anhand des PWM-Ausgangs (Pin 9 bzw. 10 bei Arduino Due) zwischen 0 V und 24 V geregelt werden (siehe Kapitel 3.1.4.4). Durch die digitalen Pins 2 und 4 bzw. 7 und 8 kann zusätzlich die Stromrichtung (Bewegungsrichtung) eingestellt werden. Mit den Pins 6 und 12 können die einzelnen Antriebe aktiviert bzw. deaktiviert werden.

3.1.2.2. Messen der Position mittels Arduino Due

Die Betriebsspannung des Arduino Due beträgt 3,3 V (siehe [Arduino, 2016a]), somit müssen die Eingänge dementsprechend angesteuert werden. Wie in Ka-

pitel 3.1.1 beschrieben, liefern die Hallsensoren Impulse von 5 V Amplitude. Damit diese mittels Arduino Due gemessen werden können, muss mit Hilfe von Spannungsteilern auf die maximale Spannung von 3,3 V skaliert werden.

Durch den Spannungsteiler (siehe Abbildung 3.3)

$$\frac{3.3}{5} = \frac{R_{13}}{R_{13} + R_{12}} \tag{3.1}$$

und der Wahl von R_{13} gleich 82 kΩ, ergibt sich für R_{12}

$$\begin{aligned} R_{12} &= 82\,\text{k}\Omega \cdot \left(\frac{5}{3.3} - 1\right) \\ R_{12} &= 42{,}2424\,\text{k}\Omega \end{aligned} \tag{3.2}$$

Bei der Verwendung von für $R_{12} = 47\,\text{k}\Omega$, ergeben sich bei einem Impuls von 5 V Amplitude eine Eingangsspannung von 3,1783 V, somit wird das Arduino Due Board im vorgeschriebenen Maße betrieben.

Diese Spannungsteiler müssen bei allen Hallsensorsignalen eingebaut werden (siehe Abbildung 3.3).

3.1.2.3. Implementieren der Schalter „Hinauf“ und „Hinunter“

Um das System möglichst realistisch nachzubauen, werden zwei Schalter, einer zum Hinauf- und einer zum Hinunterfahren, eingebaut. Hierbei muss wieder beachtet werden, dass als Eingangssignale bei Arduino Due nur Spannungen bis maximal 3,3 V erlaubt sind. In Abbildung 3.2 ist die Beschaltung der Schalter dargestellt.

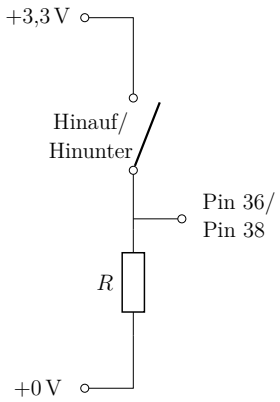


Abbildung 3.2.: Schalter zum Hinauf- bzw. Hinunterfahren

3.1.3. Gesamte Beschaltung

Abschließend ist in Abbildung 3.3 die gesamte Beschaltung des Antriebes mit dem PC über das Arduino Due Board dargestellt. [Arduino, 2016b]

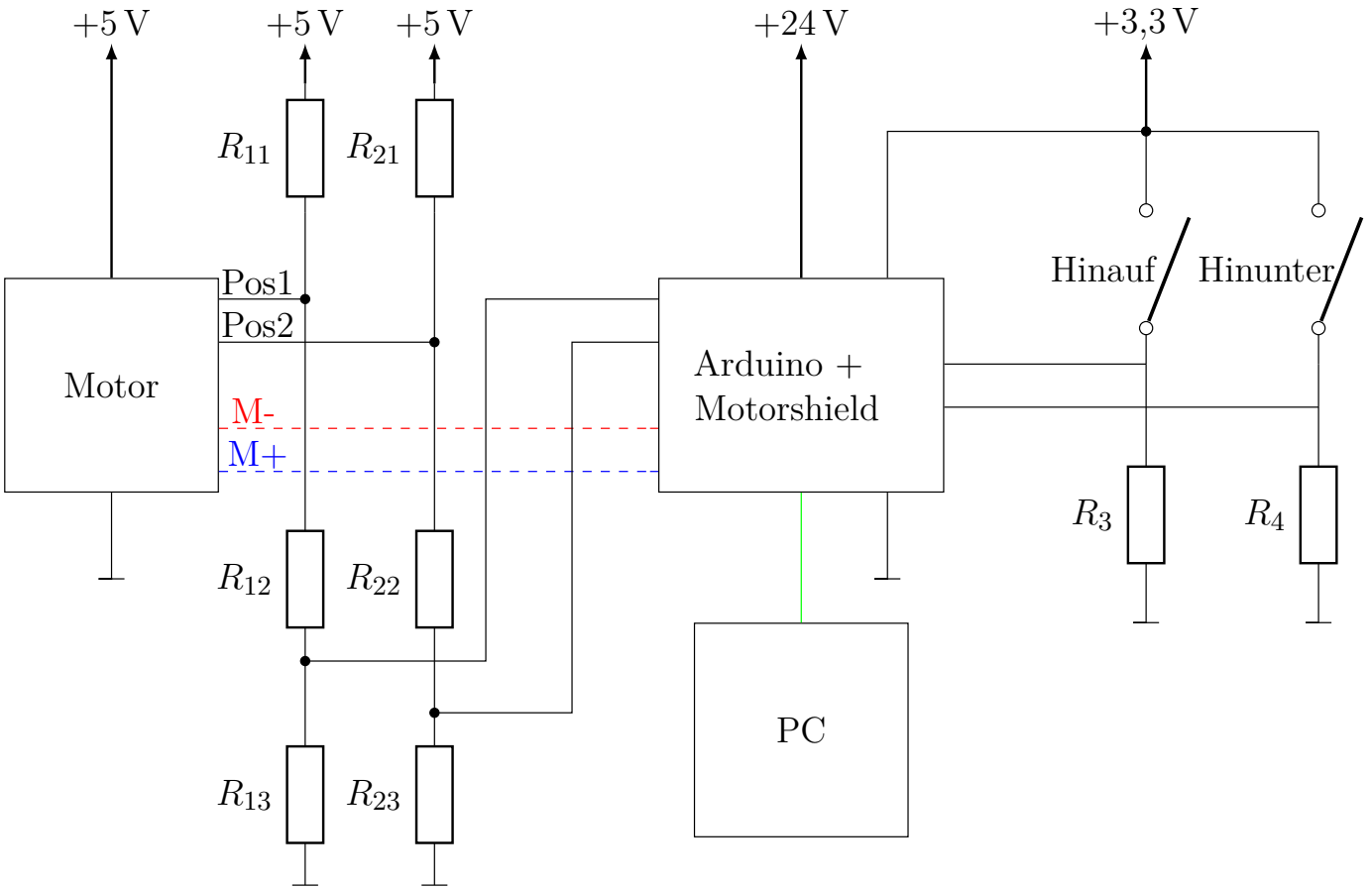


Abbildung 3.3.: Gesamte Beschaltung

3.1.4. Implementierung in Matlab Simulink

Ist das zusätzliche Paket für Arduino Due installiert (siehe Anhang A.1), können die dazugehörigen Simulink-Blöcke unter „Simulink Support Package for Arduino Hardware“ / „Common“ gefunden werden.

3.1.4.1. Einlesen des Signals Hinauf bzw. Hinunter

Die „Hinauf/Hinunter“ Impulse sind an digitalen Eingängen angeschlossen, somit müssen dementsprechende Bausteine in Simulink implementiert und die dazugehörige Pin-Nummer (36 bzw. 38) eingegeben werden. Ein Test führt zu einem wie in Abbildung 3.6 dargestellten Ergebnis. Werden die Signale nicht digital sondern analog eingelesen (zum Beispiel mit dem Quanser-NI E-Series Terminal Board), so muss das Signal weiter bearbeitet werden. In Abbildung 3.4 ist eine Möglichkeit dargestellt, um Störungen zu filtern.

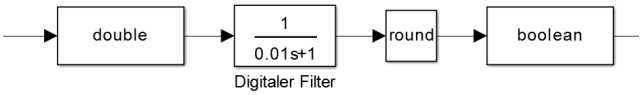


Abbildung 3.4.: digitaler Filter

Diese Impulse müssen, damit sie zum Vorsteuerungsentwurf verwendet werden können, in ein Signal zusammengeführt werden. Ein Beispiel dafür ist in Abbildung 3.5 dargestellt.

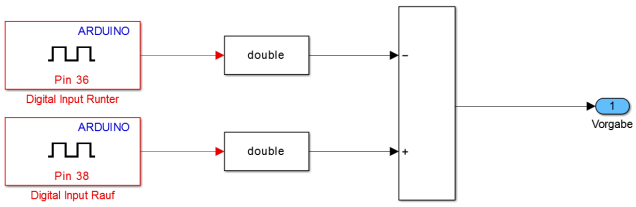


Abbildung 3.5.: Zusammenführen der Signale „Hinauf“ und „Hinunter“

Ein Beispielsignal für die Vorgabe der Richtung ist in Abbildung 3.6 dargestellt.

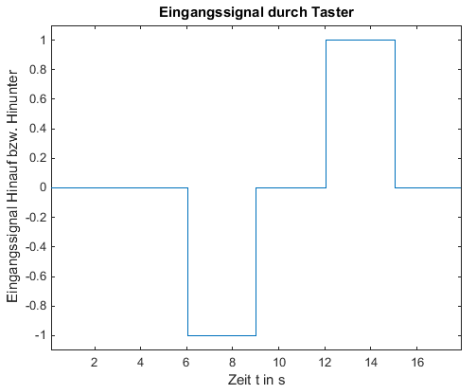


Abbildung 3.6.: Beispielsignal für die Positionsvorgabe durch die Taster „Hinauf“ und „Hinunter“

3.1.4.2. Sollgrößengenerierung

Bei der Sollgrößengenerierung wird die Stellgröße p benötigt, um zu erkennen, ob eine Auslastung von mindestens 90 % vorliegt. Zusammen mit der „Vorgabe“ (-1,0,+1 siehe Abbildung 3.6) wird ein Sollverlauf für die Kreisfrequenz ω konstruiert (siehe Anhang B.1 und Kapitel 2.3.1.3). In den Funktionsblöcken werden jeweils nur positive Werte der Vorgabe berücksichtigt. Der obere Block berechnet die gewünschte Kreisfrequenz für positive und der untere für negative Vorgaben. Danach werden diese differenziert, also der gewünschte Winkel φ generiert, zusammengeführt und weitergegeben.

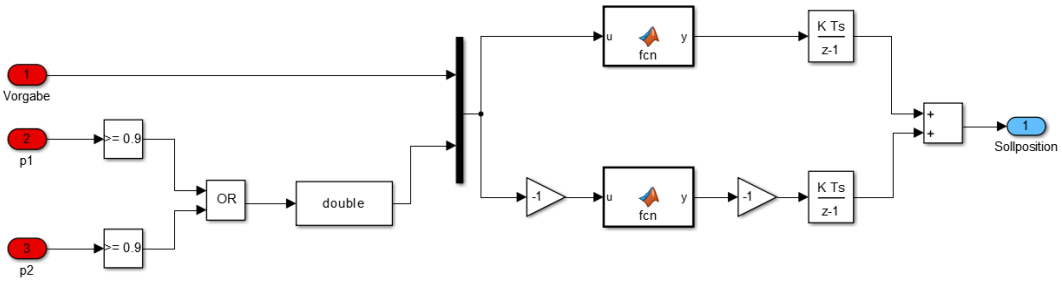
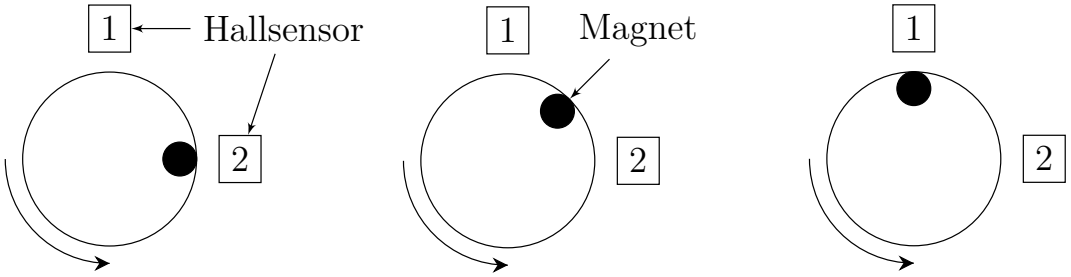


Abbildung 3.7.: Sollgrößengenerator

3.1.4.3. Positionsbestimmung

Zur Positionsbestimmung liefern zwei Hallsensoren, welche räumlich 90° voneinander verschoben sind, Impulse, wenn ein magnetisierter Teil des Motors an ihnen vorbei kommt (dies soll in Abbildung 3.8 verdeutlicht werden). Hier wird ersichtlich, dass nur eine Positionsänderung bestimmt werden kann und keine absolute Position.



(a) Magnet bei Hallsensor 2

(b) Magnet zwischen Hallsensor 1 & 2

(c) Magnet bei Hallsensor 1

Abbildung 3.8.: Positionsbestimmung mit zwei räumlich um 90° verschobenen Hallsensoren

Eingelesen werden müssen die Positionsimpulse gleich wie in Kapitel 3.1.4.1. Ein Beispiel für die eingelesenen Impulse ist in Abbildung 3.9 dargestellt (hier handelt es sich um eine Aufwärtsbewegung). Das zweite Positionssignal wurde zur Verbesserung der Anschaulichkeit mit dem Faktor 0,9 multipliziert.

Um nun aus diesen Hallimpulsen eine Positionsänderung zu erhalten, muss zuerst die Richtung erkannt werden. Zu Beginn liefern beide Hallsensoren keine Spannung (wenn der Magnet nicht in Reichweite der Hallsensoren ist), bewegt sich der Motor, so liefert zuerst Hallsensor 2 „High“ und Hallsensor 1 „Low“ (Abbildung 3.8a), danach Hallsensor 1 & 2 „High“ (Abbildung 3.8b) und danach Hallsensor 2 „Low“ und Hallsensor 1 „High“ (Abbildung 3.8c). Bewegt sich der Motor weiter, liefern die beiden Hallsensoren wieder keine Spannung, also „Low“, danach beginnt alles wieder von vorne, bis sich der Motor nicht mehr bewegt.

In Tabelle 3.1 soll dieser Zusammenhang noch einmal dargestellt werden, wobei eine „1“ einer Detektion des Magneten am Hallsensor und eine „0“

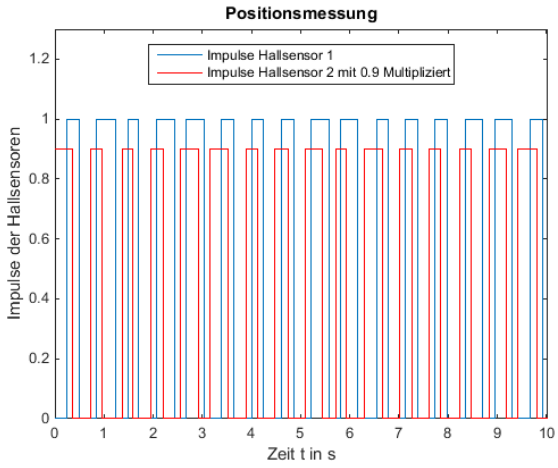


Abbildung 3.9.: Impulse der Hallsensoren

keiner Detektion entspricht. Tabelle 3.1a soll die Abfolge der Impulse beim „Hinauffahren“ und Tabelle 3.1b beim „Hinunterfahren“ zeigen.

	S_1	S_2
t_1	0	0
t_2	0	1
t_3	1	1
t_4	1	0
t_5	0	0

(a) Richtung „Hinauf“

	S_1	S_2
t_1	0	0
t_2	1	0
t_3	1	1
t_4	0	1
t_5	0	0

(b) Richtung „Hinunter“

Tabelle 3.1.: Hallsensorabläufe zur Erkennung der Drehrichtung des Motors

3.1.4.4. Ansteuerung Motor

Wie in Kapitel A.4 beschrieben, muss der PWM-Ausgang über die Pins 9 und 10 angesteuert werden. Hier wird ein Signal zwischen 0 und 255 erwartet, somit muss der vom Regler ausgegebene Wert p mit dem Faktor 255 multipliziert werden. Außerdem werden die Pins 2 und 4 bzw. 7 und 8 benötigt um die Drehrichtung des Motors, also die Stromrichtung, einzustellen. Hier muss 1 und

0 (hinauf) bzw. 0 und 1 (hinunter) eingestellt werden. Diese Signale werden durch Differenzieren der Sollgröße, wie in Abbildung 3.10 dargestellt, ermittelt.

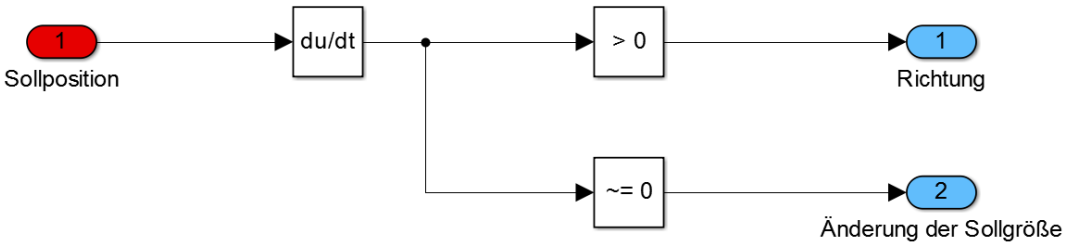


Abbildung 3.10.: Ermittlung der Drehrichtung und Erkennung einer Änderung der Sollgröße

Die PWM-Ausgänge des Pololu Dual VNH5019 Motor Driver Shield werden durch die Pins 6 bzw. 12 aktiviert bzw. deaktiviert, womit dieser auf eins gesetzt werden muss.

3.1.4.5. Reglerimplementierung in Simulink

Der Regler 2.51 wird in Simulink, wie in Abbildung 3.11 dargestellt, implementiert.

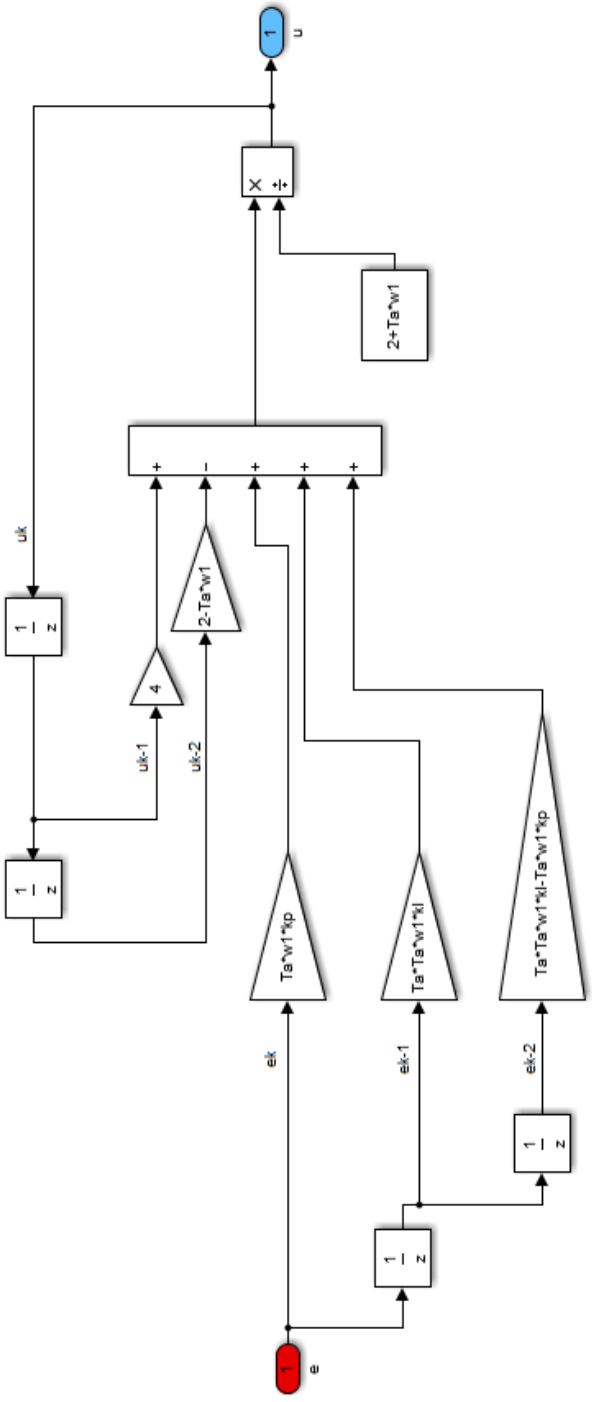


Abbildung 3.11.: Implementierung des verwendeten Reglers in Simulink

Um das Tastverhältnis p zu erhalten, muss die ermittelte Stellgröße u normiert werden. Außerdem muss das Vorzeichen von p berücksichtigt werden und bei konstanter Sollgröße (also bei Stillstand) 0 sein. Wie dies umgesetzt werden kann

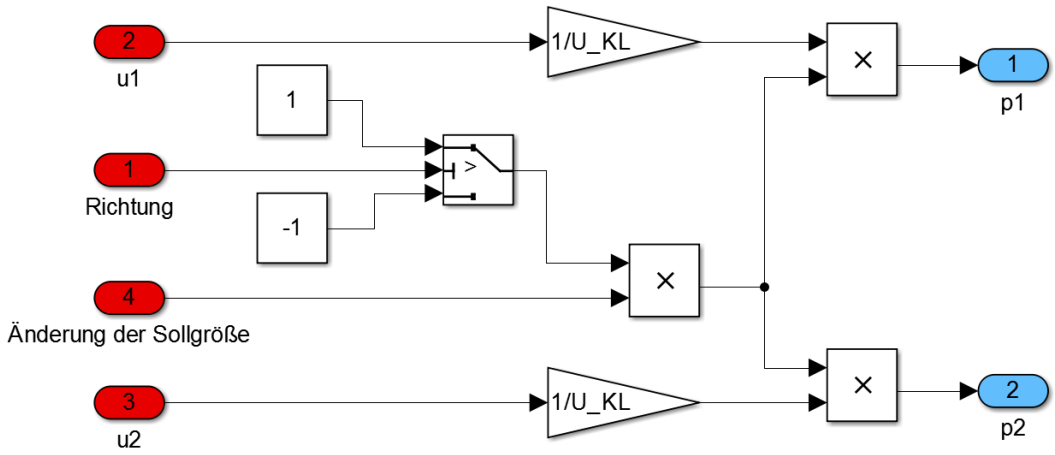


Abbildung 3.12.: Anpassen der Stellgröße

wird in Abbildung 3.12 gezeigt, hierbei kann das Signal „Änderung der Sollgröße“ entweder 1 für eine Änderung oder 0 für keine Änderung der Sollgröße annehmen (siehe Abbildung 3.10). Außerdem wird p durch die Stellgrößenbeschränkung zwischen 0 und 1 beschränkt und anschließend mit dem Faktor 255 multipliziert (siehe Abbildung 3.13).

Wie in Kapitel 2.3.1.3 beschrieben, soll die Änderung der Sollgröße bei 90 % der Auslastung beschränkt werden. Diese Grenze wird bei $p = 0.9$ erreicht, somit muss p der Sollgrößengenerierung übergeben werden.

3.1.4.6. Simulink Modell des bisherigen Regelalgorithmus

Eine Übersicht über den verwendeten Regelalgorithmus soll in Abbildung 3.13 geschaffen werden.

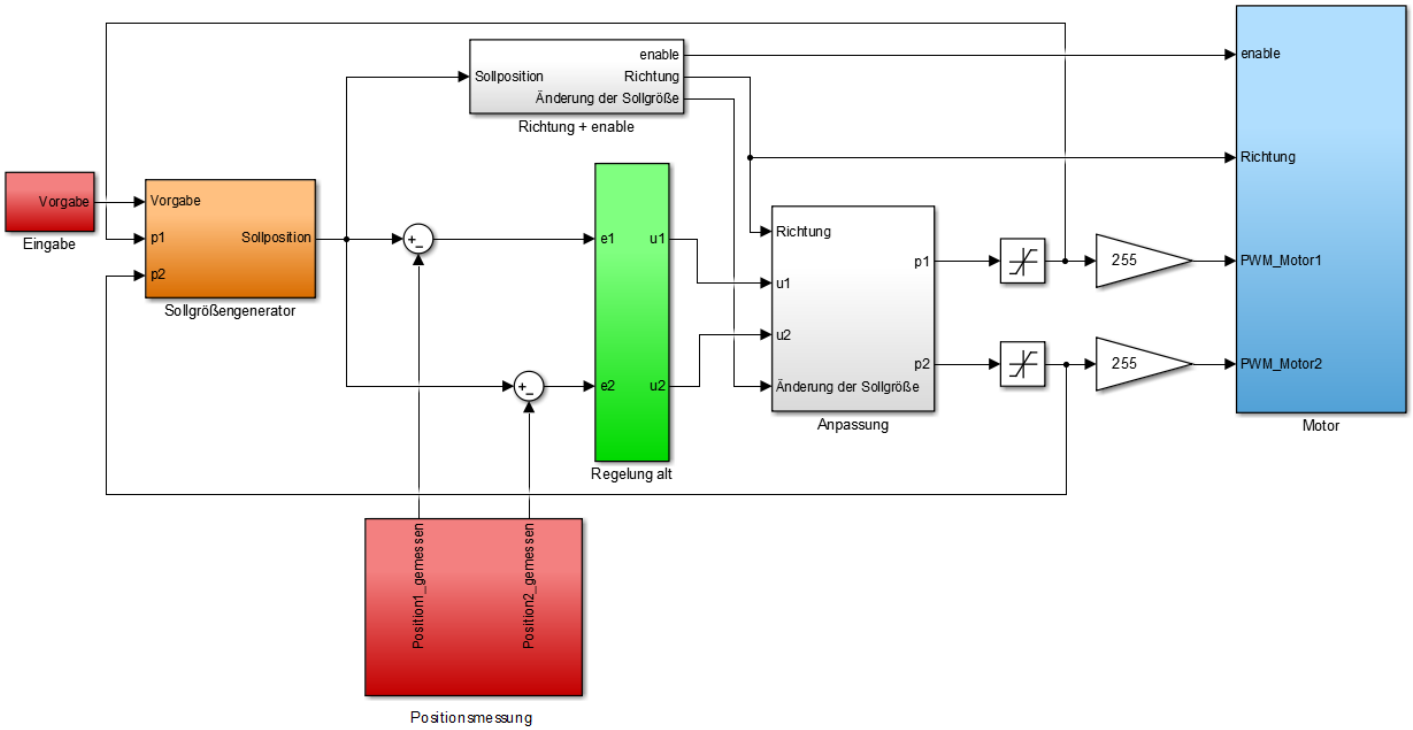
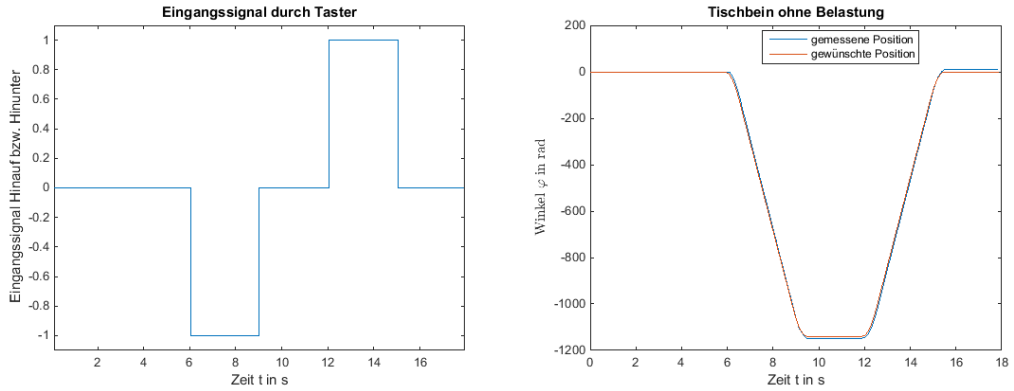


Abbildung 3.13.: Gesamte Schaltung des verwendeten Regelalgorithmus

3.1.5. Inbetriebnahme und erste Messungen

Sind nun die Motoren, das Arduino Due, das Motor Driver Shield und der Computer richtig verbunden, sind die korrekten Einstellungen in Maltab Simulink eingestellt und die Spannungsversorgungen aktiv, können die ersten Tests absolviert werden.



(a) Eingangssignal durch Taster

(b) Verlauf der Sollposition und der Istposition

Abbildung 3.14.: Erste Messungen

In Abbildung 3.14 sind die ersten Messungen dargestellt. In Abbildung 3.14a ist das Signal der Taster dargestellt, nach 6 s wird der Schalter zum Hinunterfahren geschlossen und für 2 s gehalten. Nach dem loslassen des Tasters geht das Signal auf 0 zurück. Zum Zeitpunkt $t = 12\,\text{s}$ passiert dasselbe mit dem Taster fürs Hinauffahren. In Abbildung 3.14b ist der dazugehörige Verlauf von Soll- und Istposition abgebildet. Man erkennt, dass das Tischbein mit der vorhandenen PI-Regelung, wenn das Tischbein nicht belastet wird, der Sollposition sehr gut folgen kann. Weitere Messungen sind in Kapitel **??** dargestellt.

3.2. Parameteridentifikation

Um eine flachheitsbasierte Folgeregelung implementieren zu können, ist es notwendig, die Maschinenparameter zu kennen. Mit Hilfe verschiedener Versuche

ist es möglich, diese Parameter zu identifizieren. Da der entwickelte Regelalgorithmus bei unterschiedlichen Antrieben gleicher Bauart verwendet werden soll, müssen Parameterabweichungen von der Regelung kompensiert werden.

3.2.1. Ankerwiderstand R_a

3.2.1.1. Versuchsaufbau

Das in Abbildung 3.15 eingezeichnete Widerstandsmessgerät muss mit den Klemmen 1 und 5 (siehe Kapitel 3.1.1) des Motors verbunden werden.

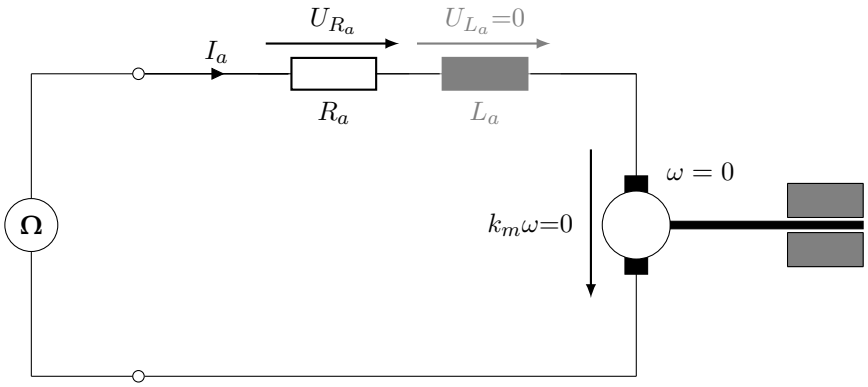


Abbildung 3.15.: Ersatzschaltbild zur Messung des Ankerwiderstandes R_a

3.2.1.2. Versuchsdurchführung

Das in Kapitel 2.2.1 beschriebene Verfahren, wird auch von handelsüblichen Widerstandsmessgeräten verwendet. Hier wurde das LCR-Messgerät Voltcraft LCR-9063 verwendet, um die Widerstandswerte an verschiedenen Positionen zu messen und daraus den Mittelwert zu berechnen.

Versuch	R_a
1	4,8 Ω
2	2,6 Ω
3	2,6 Ω
4	2,6 Ω
5	3,5 Ω
6	2,6 Ω
7	4,1 Ω
8	2,6 Ω
Summe	25,4 Ω

Tabelle 3.2.: Aufgenommene Werte für den Ankerwiderstand R_a

3.2.1.3. Ergebnisse

Die Ergebnisse der einzelnen Messungen sind in Tabelle 3.2 dargestellt.

Aus diesen Werten wird nun der Mittelwert nach Gleichung 3.3 berechnet.

$$R_a = \frac{1}{N} \cdot \sum_{i=1}^{N} R_{a,i}$$
$$R_a = \frac{25{,}4}{8} \tag{3.3}$$

Somit ergibt sich der Ankerwiderstand R_a zu

$$\boxed{R_a = 3{,}175\,\Omega} \tag{3.4}$$

3.2.2. Ankerinduktivität L_a

3.2.2.1. Versuchsaufbau

Das in Abbildung 3.16 eingezeichnete Impedanzmessgerät muss wie in Kapitel 3.2.1 mit den Klemmen 1 und 5 (siehe Kapitel 3.1.1) des Motors verbunden werden.

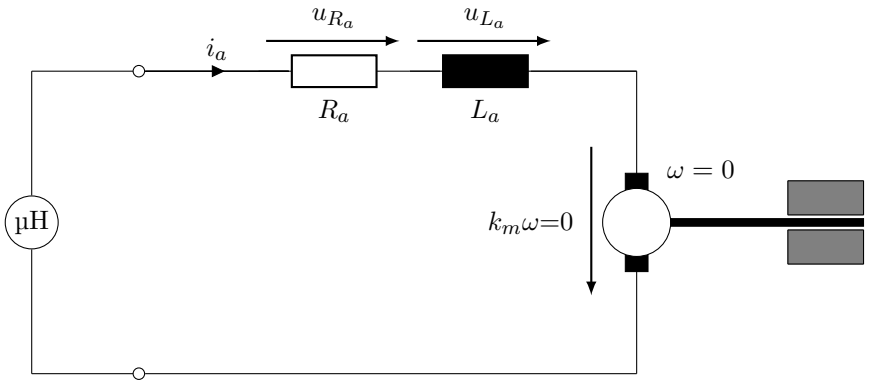


Abbildung 3.16.: Ersatzschaltbild zur Messung der Ankerinduktivität L_a

3.2.2.2. Versuchsdurchführung

Die in Kapitel 2.2.2 beschriebenen Verfahren zur Induktivitätsmessung benötigen sehr schnelle und genaue Messgeräte, um sinnvolle Ergebnisse zu liefern. Da die vorhandenen Messgeräte dies nicht erfüllen, wird wieder das LCR-Messgerät Voltcraft LCR-9063 verwendet. Auch hier wurden wieder Messungen an verschiedenen Positionen durchgeführt und aus diesen der Mittelwert berechnet.

3.2.2.3. Ergebnisse

Versuch	L_a
1	1,462 mH
2	1,096 mH
3	1,359 mH
4	1,11 mH
5	1,115 mH
6	1,1 mH
7	1,106 mH
8	1,12 mH
Summe	9,422 mH

Tabelle 3.3.: Aufgenommene Werte für die Ankerinduktivität L_a

Wird nun der Mittelwert der Messwerte von Tabelle 3.3 nach Gleichung 3.5 berechnet,

$$L_a = \frac{1}{N} \cdot \sum_{i=1}^{N} L_{a,i}$$
$$L_a = \frac{9{,}422}{8} \tag{3.5}$$

so ergibt sich die Ankerinduktivität L_a zu

$$\boxed{L_a = 1{,}1778\,\text{mH}} \tag{3.6}$$

3.2.3. Drehmomentkonstante k_m

3.2.3.1. Versuchsaufbau

Um die Drehmomentkonstante k_m bestimmen zu können, müssen, wie in Kapitel 2.2.3 beschrieben, die Größen U_{KL}, I_a, Hallsensorsignal 1, Hallsensorsignal 2 und die Zeit gemessen werden. Die Zeit wird in Simulink automatisch gespeichert, wenn Daten eingelesen werden.

Die Signale der Hallsensoren werden wie in Kapitel 3.1.2.2 beschrieben gemessen. Dies wird in Abbildung 3.17 durch eine Verbindung zwischen dem Motor und dem PC angedeutet. Die Messung der Größen U_{KL} und I_a sind ebenso in dieser Abbildung dargestellt.

Es werden zwei Messwiderstände verwendet und die Spannungsabfälle an diesen aufgezeichnet. Der Spannungsabfall am Messwiderstand R_{iA} wird auf einen Strom umgerechnet und entspricht näherungsweise dem Ankerstrom I_a. Durch korrigieren des Messfehlers, welcher durch die Messung der Spannung an R_{iV} entsteht, kann I_a exakt bestimmt werden. Durch das Messen des Spannungsabfalls an R_{iV} kann die Klemmspannung berechnet werden. Dazu muss die gemessene Spannung invertiert werden, da bei der Spannungsmessung mit dem Quanser-NI E-Series Terminal Potentialdifferenzen nur mit einem Bezugspotential gemessen werden können.

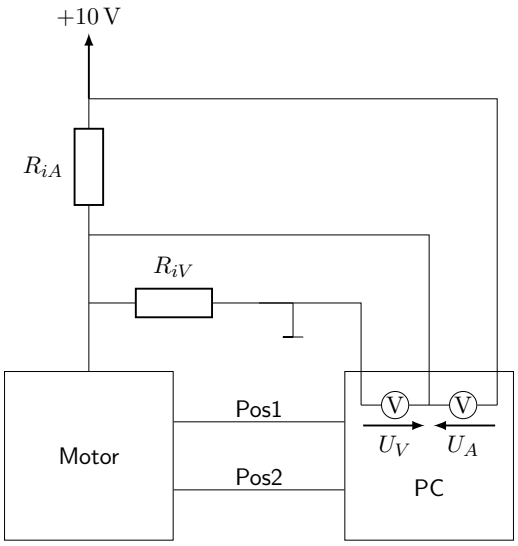


Abbildung 3.17.: Messaufbau zur Bestimmung der Drehmomentkonstante k_m

3.2.3.2. Versuchsdurchführung

Es wird eine konstante Spannung an den Motor gelegt, welche groß genug sein muss, damit sich der Motor bewegt und eine Spannung induziert wird.

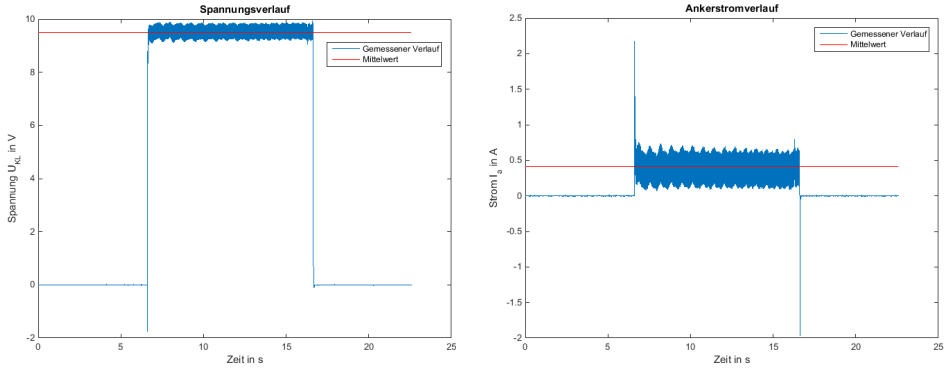
Die Spannung U_{KL} kann einfach berechnet werden.

$$U_{KL} = -U_V \tag{3.7}$$

Um den Strom I_a zu bestimmen, wird der Spannungsabfall U_A gemessen und durch den Messwiderstand R_{iA} auf den Strom zurückgerechnet. Durch Abziehen des berechneten Stroms welcher durch den Messwiderstand R_{iV} fließt vom berechneten Strom der durch R_{iA} fließt, ergibt sich der gesuchte Ankerstrom.

$$I_a = \frac{U_A}{R_{iA}} - \frac{U_{KL}}{R_{iV}} \tag{3.8}$$

Die Messungen der Größen U_{KL} (siehe Abbildung 3.18a) und I_a (siehe Abbildung 3.18b) werden über einen Zeitraum von ca. 10 s gemessen. Bei annähernd konstanten Werten von U_a und I_a muss der Mittelwert gebildet werden (siehe Abbildung 3.18)



(a) Spannungsverlauf U_{KL} (b) Ankerstromverlauf I_a

Abbildung 3.18.: Gemessene Verläufe von Strom und Spannung beim Hinauffahren zur Bestimmung von km

3.2.3.3. Ergebnisse

Um nun die Drehmomentkonstante k_m zu bestimmen, müssen die ermittelten Werte (siehe Tabelle 3.19) in Gleichung 2.49 eingesetzt werden.

$\lvert i_a \rvert = 0{,}5733\,\mathrm{A}$	$\lvert i_a \rvert = 0{,}4093\,\mathrm{A}$
$\lvert u_{KL} \rvert = 9{,}3375\,\mathrm{V}$	$\lvert u_{KL} \rvert = 9{,}4925\,\mathrm{V}$
$\lvert \omega \rvert = 214{,}335\,\mathrm{s}^{-}2$	$\lvert \omega \rvert = 242{,}0771\,\mathrm{s}^{-}2$
$k_m = 0{,}035\,072\,5$	$k_m = 0{,}033\,844\,5$

(a) Hinunterfahren (b) Hinauffahren

Abbildung 3.19.: Darstellung der aufgenommenen Werte für (a) Hinuter- bzw. (b) Hinauffahren

Wird aus diesen Werten der Mittelwert gebildet, ergibt sich die Drehmoment-

konstante k_m zu

$$\boxed{k_m = 0{,}034\,458\,5\,\mathrm{Nm/A}} \tag{3.9}$$

3.2.4. Trägheitsmoment J_0, Übersetzungsverhältnis k und Reibungsparameter d_r

Um die gesuchten Größen (Trägheitsmoment J_0, Übersetzungsverhältnis k und Reibungsparameter d_r) möglichst genau bestimmen zu können, werden mehrere Versuche mit unterschiedlichen Spannungen bzw. bei unterschiedlicher Belastung vorgenommen.

3.2.4.1. Versuchsaufbau

Um eine Optimierung vornehmen zu können, wird eine Eingangs- sowie eine Ausgangsgröße benötigt, weshalb hier die Spannung U_{KL} und die Rotorposition φ gemessen werden muss. Außerdem soll eine möglichst genaue Auflösung der Messung erfolgen. Um dies zu erreichen, muss in Simulink der Baustein „serial transmit“ verwendet werden (siehe Abbildung 3.20). Hier werden „A“ als Startbit und „0“ als Stopbit verwendet. Als Daten werden die gemessene Position, die Drehrichtung und der Eingangsverlauf übergeben. Die übertragenen Datenpakete werden im uint8 Format übermittelt, also werden keine Kommastellen übertragen. Da jedoch die gemessene Position möglichst genau übermittelt werden soll, wird sie mit 10 000 multipliziert und in 4 Datenblöcke im uint8 Format aufgeteilt (siehe Abbildung 3.21).

Diese Daten werden an den USB-Port gesendet, welche mit dem Programm HTerm ausgelesen werden. Die eingelesenen Dateien sollen im Format „Dec“ gespeichert werden, die einzelnen Blöcke durch „;“ getrennt und der Zeitstempel mitgegeben werden. Zu beachten ist außerdem, dass der USB-Port und die Baudrate übereinstimmen sowie die Daten auf 8 bits pro Block beschränkt werden.

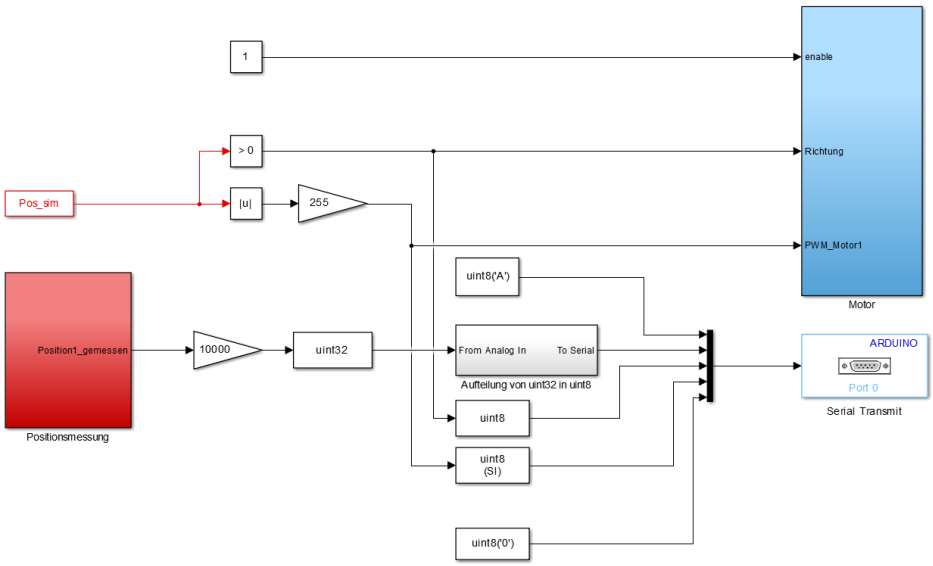


Abbildung 3.20.: Simulinkmodell zur Parameteridentifikation

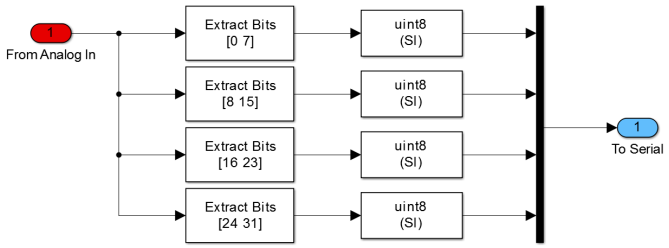


Abbildung 3.21.: Aufteilung von uint32 in uint8

Diese Textdateien werden mit dem Programm kompaktesDatenlesen.m (siehe Anhang B.4) ausgelesen. Hier werden fehlerhafte Blöcke eliminiert, ein Zeitvektor erstellt und die aufgeteilte Position wieder in einen Wert zusammengefügt und durch 10 000 dividiert.

Als Eingangsgröße wird ein Sprung, welcher 10 s bis 15 s anliegt, gewählt (je nach Spannung und Belastung fährt das Tischbein mehr oder weniger weit).

Somit stehen nun ein Vektor für die Eingangsgröße, einer für die Position und ein Zeitvektor zur Verfügung.

3.2.4.2. Versuchsdurchführung

Je mehr Informationen zur Verfügung stehen, desto genauer können die gesuchten Größen bestimmt werden. Deshalb werden mehrere Versuche mit unterschiedlichen Spannungen beziehungsweise bei unterschiedlicher Belastung vorgenommen.

Ein Überblick über die Versuche soll in Tabelle 3.4 geschaffen werden.

In Anhang B.2 ist das Hauptprogramm zur Parameteridentifikation abgebildet. Hier werden zuerst Startwerte definiert, die Daten aus den Textdateien ausgelesen und aufbereitet. Danach wird eine Funktion (siehe Anhang B.5) aufgerufen, in welcher die Optionen zur Optimierung definiert werden (z.B.: obere und untere Schranke, Maximale Iterationen,...). Es wird die zu optimierende Funktion mit „simulannealbnd“ aufgerufen. Dieser Befehl sucht ein globales Minimum des Fehlers, welcher von der verwendeten Funktion zurückgegeben wird.

Die zu optimierende Funktion (siehe Anhang B.6) berechnet die Lösung der Differentialgleichungen mit Hilfe des ode15s Befehls. Dieser wird hier anstelle von ode45 verwendet, da der ode45 Befehl in diesem Fall sehr langsam ist. Bei dem ode45 Befehl handelt es sich um ein explizites Verfahren. Aufgrund der sensiblen Stabilitätskriterien können hier nur kleine Schrittweiten verwendet werden. Dies ist bei ode15s nicht der Fall, da dieser ein implizites Verfahren ist, welches größere Schrittweiten erlaubt.

Die berechnete Lösung wird anschließend dargestellt, womit der laufende Prozess verfolgt werden kann.

Versuchsnummer	Versorgungsspannung	Belastung
1	9,87 V	0 kg
2	−9,92 V	0 kg
3	9,84 V	5,151 kg
4	−9,92 V	5,151 kg
5	9,81 V	10,309 kg
6	−9,92 V	10,309 kg
7	9,77 V	15,408 kg
8	−9,93 V	15,408 kg
9	9,74 V	20,561 kg
10	−9,94 V	20,561 kg
11	9,7 V	25,512 kg
12	−9,94 V	25,512 kg
13	9,66 V	30,413 kg
14	−9,95 V	30,413 kg
15	14,77 V	30,413 kg
16	−15,04 V	30,413 kg
17	14,81 V	25,512 kg
18	−15,02 V	25,512 kg
19	14,85 V	20,561 kg
20	−15,04 V	20,561 kg
21	14,88 V	15,408 kg
22	−15,03 V	15,408 kg
23	14,92 V	10,309 kg
24	−15,02 V	10,309 kg
25	14,96 V	5,151 kg
26	−15,02 V	5,151 kg
27	15 V	0 kg
28	−15,01 V	0 kg
29	19,99 V	0 kg
30	−20,03 V	0 kg
31	19,97 V	5,151 kg
32	−20,03 V	5,151 kg
33	19,93 V	10,309 kg
34	−20,03 V	10,309 kg
35	19,90 V	15,408 kg
36	−20,03 V	15,408 kg
37	19,86 V	20,561 kg
38	−20,04 V	20,561 kg
39	19,83 V	25,512 kg
40	−20,04 V	25,512 kg
41	19,78 V	30,413 kg
42	−20,05 V	30,413 kg

Tabelle 3.4.: Überblick über die Versuche

In Anhang B.7 werden zuerst die, dem entsprechenden Zeitpunkt zugehörigen, Größen (Drehrichtung, Eingangsspannung und die Belastung) bestimmt und danach die Differenzialgleichungen gelöst, wodurch ein Vektor mit drei Zustandsgrößen entsteht, welcher anschließend zurückgegeben wird.

3.2.4.3. Ergebnisse

Der Vergleich zwischen gemessener und simulierter Position φ wird in Abbildung 3.22 dargestellt. Dies ist für die Vorsteuerung ausreichend genau, da diese

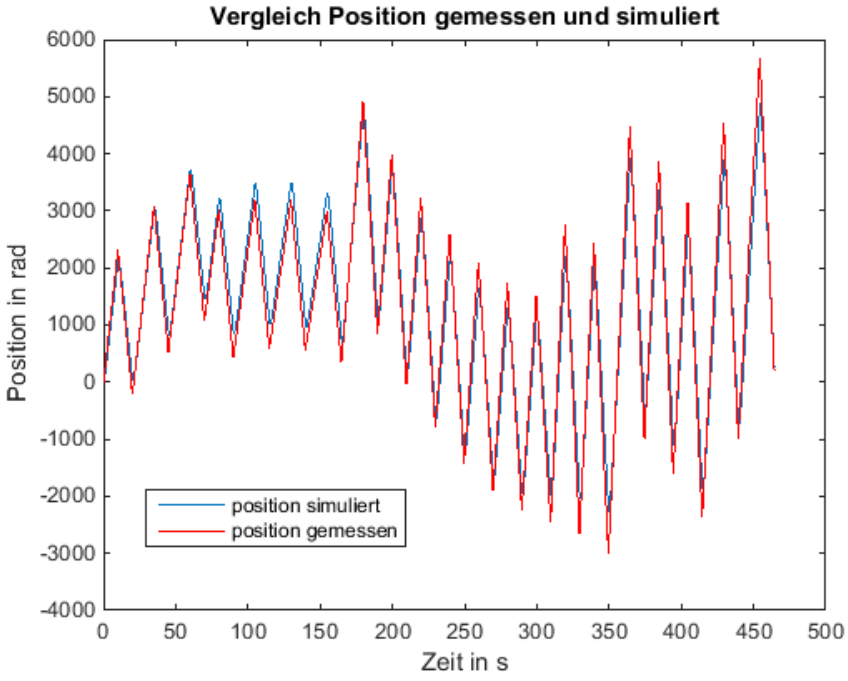


Abbildung 3.22.: Vergleich Messung und Simulation

bei verschiedenen Antrieben, welche leicht abweichende Parameter besitzen, eingesetzt werden soll.

Die ermittelten Parameter werden in Tabelle 3.5 aufgeführt.

J_0	k	d_r
$5{,}8047 \cdot 10^{-5}\,\mathrm{kgm}^2$	$8{,}1581 \cdot 10^{-5}$	$1{,}2023 \cdot 10^{-4}$

Tabelle 3.5.: Zusammenfassung der ermittelten Parameter J_0, k und d_r

3.2.5. Zusammenfassung der ermittelten Parameter

Abschließend werden alle ermittelten Parameter in Tabelle 3.6 aufgeführt.

R_a	L_a	k_m	J_0	k	d_r
$3{,}175\,\Omega$	$1{,}1778\,\mathrm{mH}$	$0{,}034\,458\,5\,\mathrm{Nm/A}$	$5{,}8047 \cdot 10^{-5}\,\mathrm{kgm}^2$	$8{,}1581 \cdot 10^{-5}$	$1{,}2023 \cdot 10^{-4}$

Tabelle 3.6.: Zusammenfassung aller ermittelten Parameter

3.3. Reglerimplementierung

3.3.1. Implementierung des Reglers mit Sollgrößenanpassung

Um eine Sollgrößenanpassung, wie in Kapitel 2.3.2 beschrieben, durchführen zu können, muss zuerst erkannt werden, ob ein Überschwingen vorliegt. Dies kann wie in Abbildung 3.23 dargestellt ausgeführt werden. Hier wurde als Grenze ein minimales Überschwingen von $\pm 10\,\mathrm{rad}$ vorausgesetzt, damit die Anpassung nicht bereits durch die Quantisierung der Positionsmessung aktiv wird. Außerdem ist unter diesem Wert kein spürbares „Hüpfen“ vorhanden.

Liegt ein Überschwingen vor, so muss erkannt werden, welches Tischbein davon betroffen ist. Dieses wird integriert und zur Sollgröße addiert (siehe Abbildung 3.24). Für den „Gain“ des Integrators hat sich ein Wert von 5 als gut erwiesen. Der gesamte Zusammenhang wird in Abbildung 3.25 dargestellt.

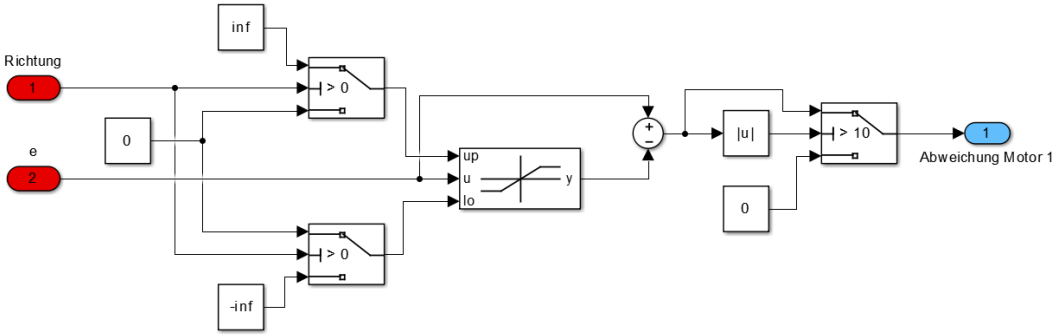


Abbildung 3.23.: Simulinkmodell zur Detektion von Überschwinger

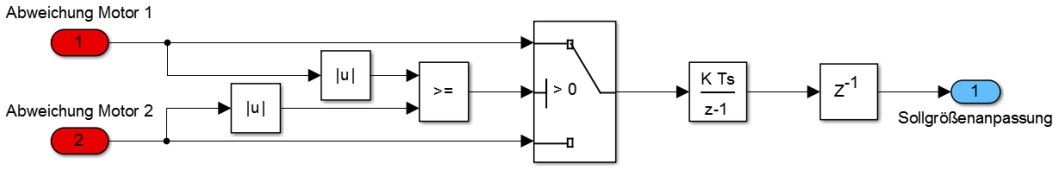


Abbildung 3.24.: Sollgrößenanpassung

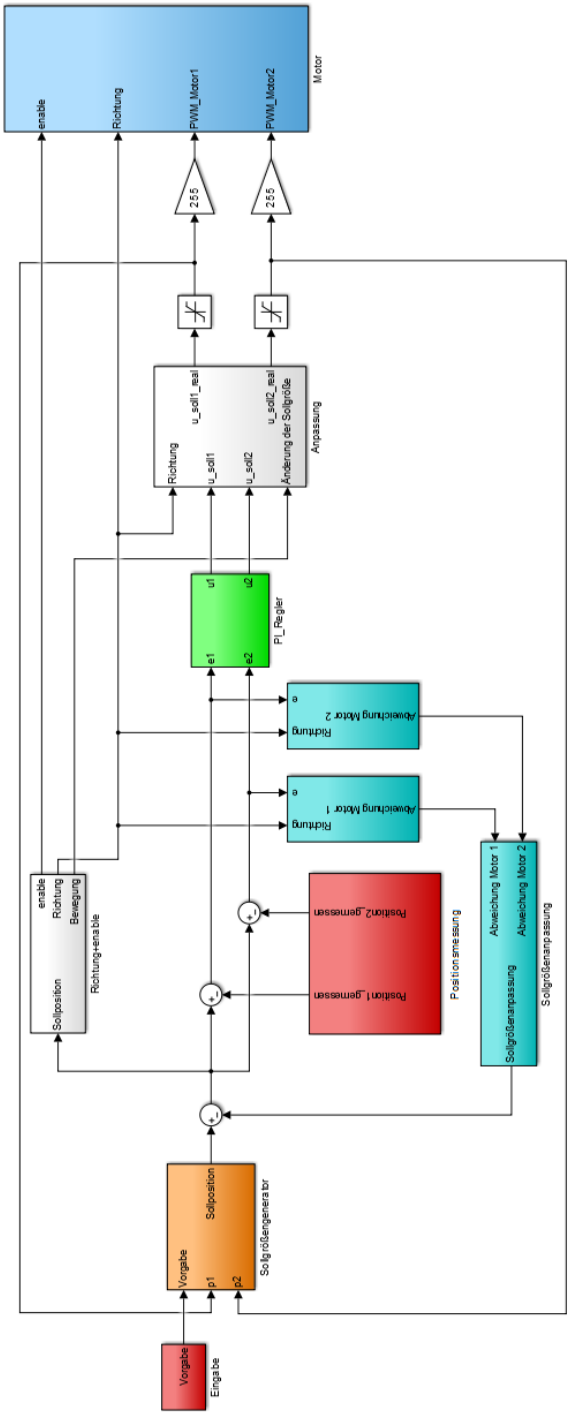


Abbildung 3.25.: Gesamtes Simulinkmodell des Reglers mit Sollgrößenanpassung

3.3.2. Implementierung des Reglers mit Anpassung der Abbrems- und Beschleunigungszeit

Zur Anpassung der Beschleunigungs- und Abbremszeit werden die aktuellen Positionen der Motoren und die Sollposition dem Sollgrößengenerator übergeben (siehe Abbildung 3.26). Fährt ein Antrieb während des Abbrems- beziehungsweise Beschleunigungsvorgangs schneller nach unten oder oben, so wird die dafür zur Verfügung stehende Zeit um 0,1 s verlängert. Kann dadurch das Auseinanderlaufen der Antriebe nicht innerhalb von 0,1 s verhindert werden, so wird die Abbrems- und Beschleunigungszeit weiter erhöht. Maximal wird die Zeit auf eine 1 s erhöht, damit sich der Tisch nicht zu lange ohne Einwirken des Benutzers bewegt (siehe Anhang B.9).

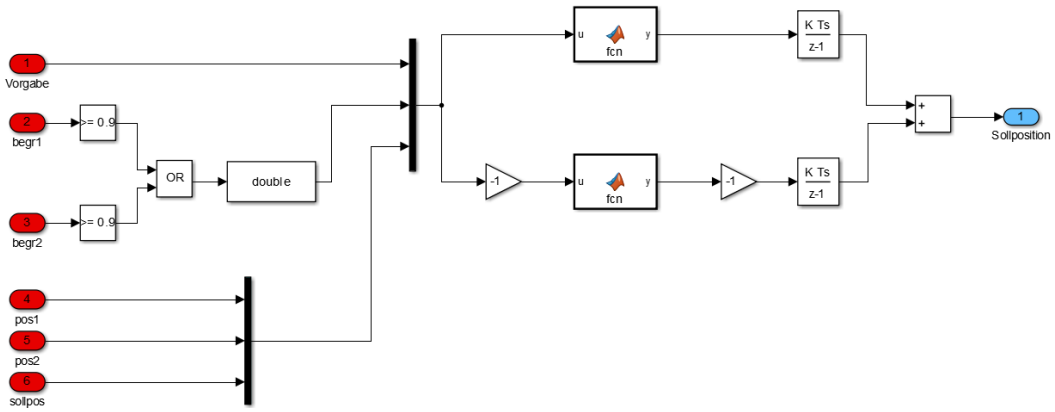


Abbildung 3.26.: Anpassung der Abbrems- und Beschleunigungszeit

Der gesamte Zusammenhang wird in Abbildung 3.27 dargestellt.

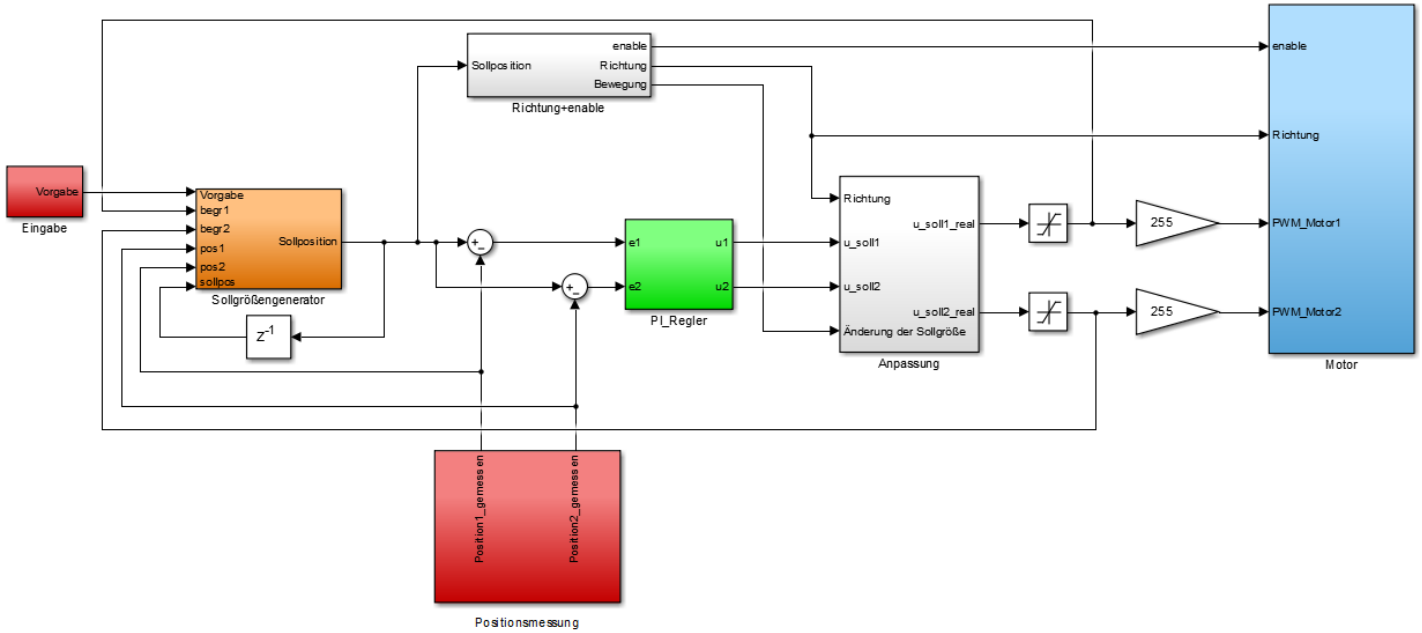


Abbildung 3.27.: Gesamtes Simulinkmodell des Reglers mit Anpassung der Abbrems- und Beschleunigungszeit

3.3.3. Implementierung der flachheitsbasierten Vorsteuerung

Bei der flachheitsbasierten Vorsteuerung muss in der Sollgrößengenerierung nicht nur der Verlauf der Sollposition erzeugt werden, sondern ebenfalls die n Ableitungen der Sollposition (siehe Abbildung 3.28) gebildet werden. Der

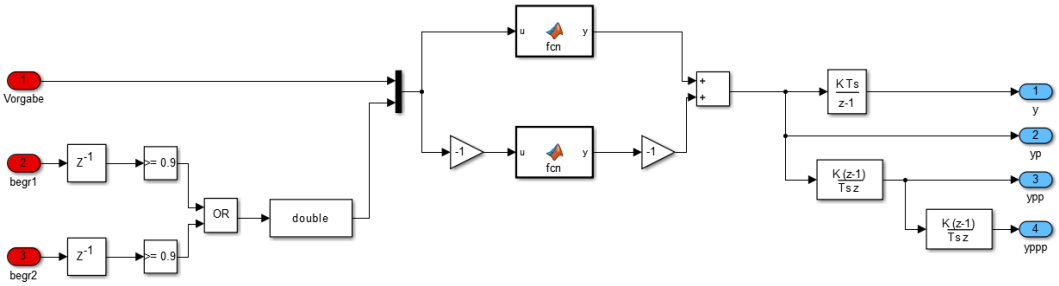


Abbildung 3.28.: Sollgrößengenerierung und Bilden der Ableitungen

in Gleichung 2.63 beschriebene Zusammenhang zwischen den Ableitungen der Sollgröße und der Stellgröße p wird, wie in Abbildung 3.29 dargestellt, realisiert.

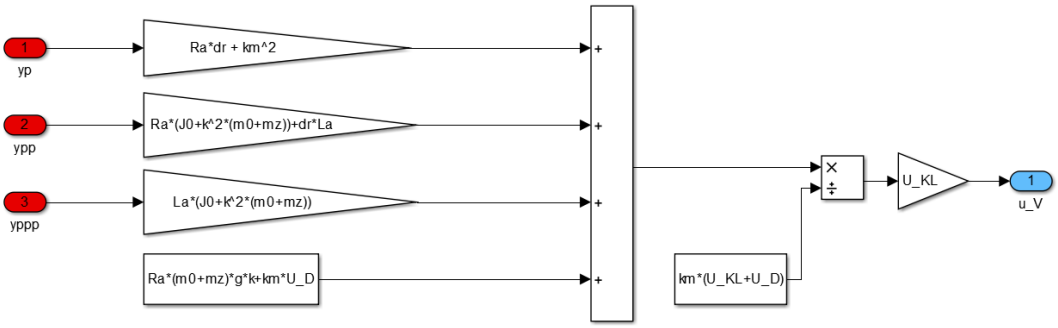


Abbildung 3.29.: Berechnen der Stellgröße u_V

Die gesamte Implementierung wird in Abbildung 3.30 dargestellt.

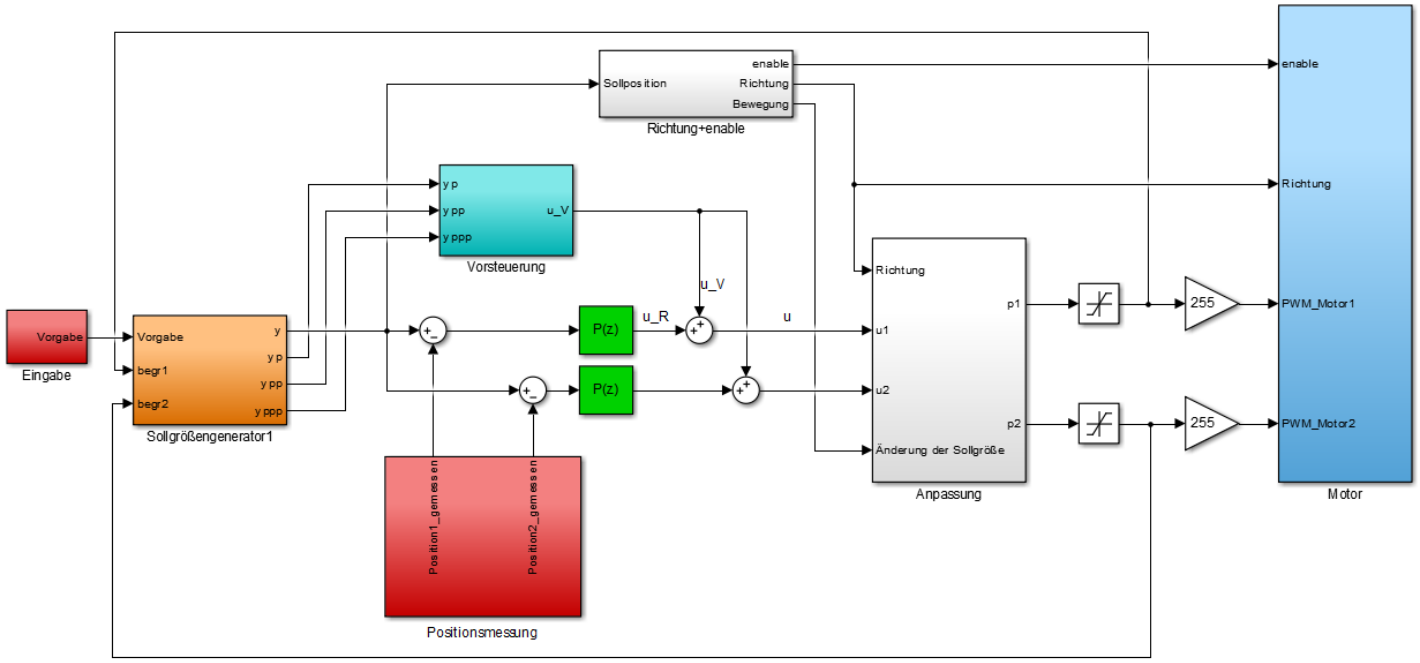


Abbildung 3.30.: Gesamtes Simulinkmodell des Reglers mit Anpassung der Abbrems- und Beschleunigungsszeit

4. Reglertest

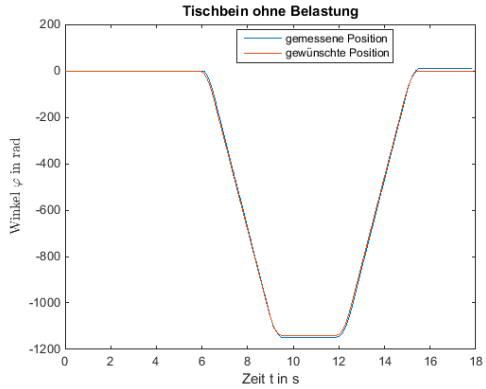
Die in dieser Arbeit entwickelten Regler wurden mit zwei verschiedenen Tischen getestet. Zum Einen mit einem etwas älteren Antrieb, welcher nicht mehr produziert wird, in vielen älteren Modellen aber verwendet wurde, welcher jedoch das vorhandene Problem deutlich zeigt; dieses wird im Weiteren „alter Tisch“ genannt. Und zum Anderen mit einem neueren Modell, welches aktuell in Tischen verbaut wird; dieses wird mit „neuer Tisch“ betitelt.

Es wird nur ein Tischbein der Tische belastet, diese werden mit den verschieden Regelungen getestet und die Messergebnisse untersucht.

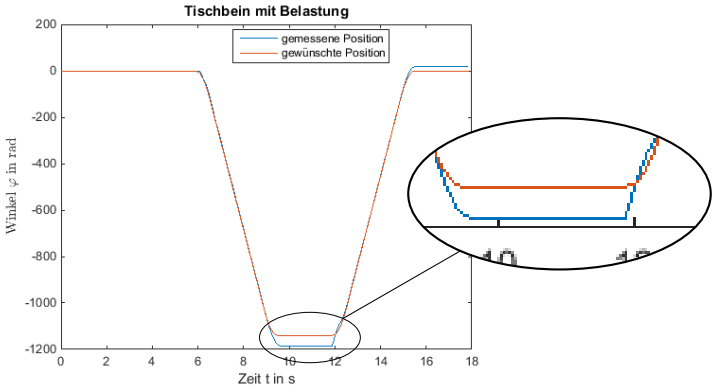
4.1. Test des bestehenden PI-Reglers

Fährt das belastete Tischbein hinunter beziehungsweise hinauf, so entsteht ein Positionsuntschied zwischen den Tischbeinen. Da dieser beim Hinunterfahren größer ist wird dieser Fall genauer untersucht. Das belastete Tischbein benötigt beim Abbremsen mehr Zeit und schafft es somit nicht rechtzeitig zum Stillstand zu kommen, was sich in einer Abweichung zwischen der gewünschten und der tatsächlichen Position äußert (siehe Abbildung 4.1a). Da sich das unbelastete Tischbein befindet sich der Sollposition (siehe Abbildung 4.1a), ist es zu einem Positionsunterschied gekommen. Es ist für den Regler nicht möglich diesen Unterschied auszugleichen, so lange sich die Sollposition nicht ändert. Fährt der Tisch nach unten, so fährt das belastete Tischbein etwas später weg, hat also keine Zeit um langsam zu beschleunigen, womit ein leichtes „Hüpfen“ entsteht. Fährt der Tisch jedoch hinauf, so versucht der Regler den Positionsunterschied schnellst möglich auszugleichen. Hierbei entsteht ein noch spürbarerer „Hüpfer“ für den Anwender spärbar wird, weswegen dieser Ablauf für die Untersuchung der Regelalgorithmen gewählt wird.

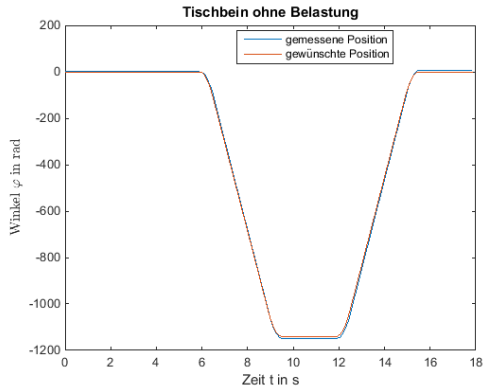
Bei dem neuen Tisch ist kein erkennbarer Unterschied zwischen belasteten und unbelasteten Tischbein zu erkennen (siehe Abbildung 4.1d), und somit ist für den Anwender kein „Hüpfen“ vorhanden.



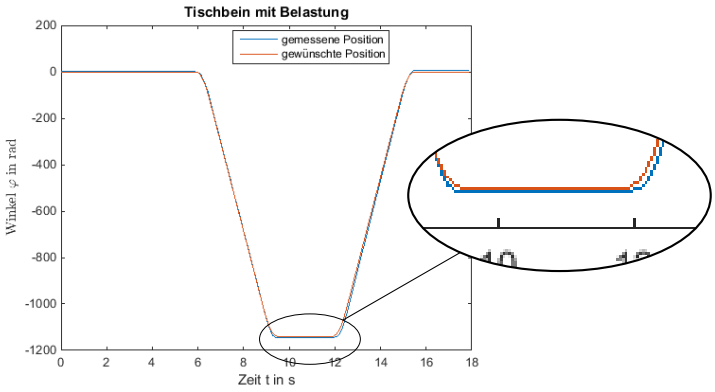
(a) ohne Belastung alter Tisch



(b) mit Belastung alter Tisch



(c) ohne Belastung neuer Tisch



(d) mit Belastung neuer Tisch

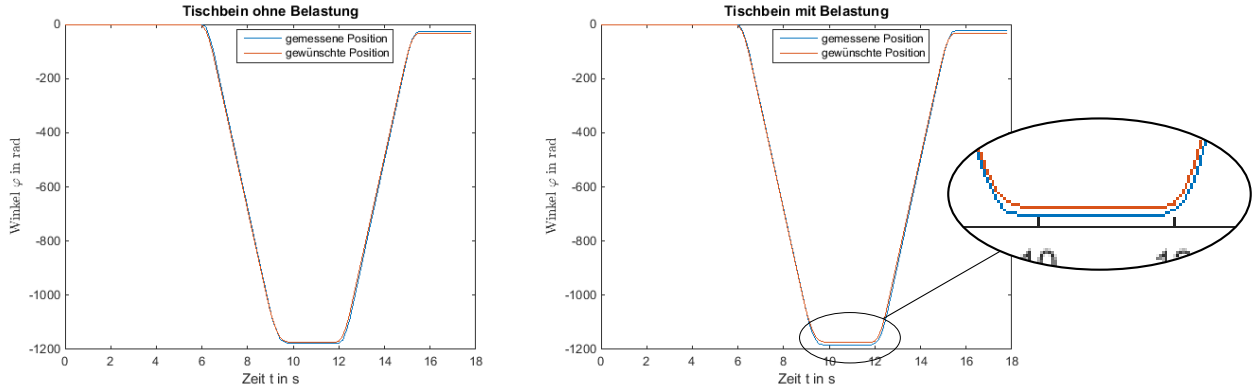
Abbildung 4.1.: Test des bestehenden PI-Reglers

4.2. Test des PI-Reglers mit Sollgrößenanpassung

Bei dem PI-Regler mit Sollgrößenanpassung (siehe Abbildung 4.2b) kommt es zu einer Anpassung der Sollgröße sobald die Abweichung größer als 10 rad wird.

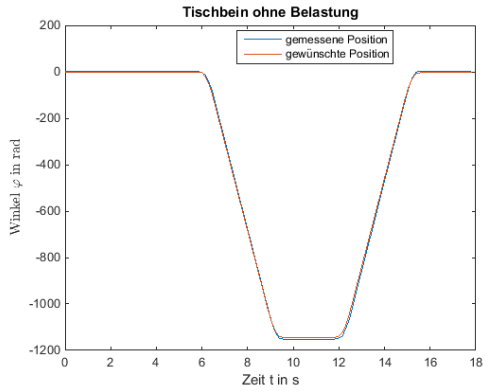
Unter dieser Grenze kommt es zu keiner Anpassung des Positionsunterschieds, dies ist notwendig damit die Anpassung nicht bereits durch die Quantisierung der Positionsmessung aktiv wird. Außerdem ist bei diesem geringen Positionsunterschied kein „Hüpfen“ für den Anwender spürbar und auch keine größere Steigung der Position beim Wegfahren erkennbar. Somit ist zu erkennen, dass diese Modifikation des Regelalgorithmus zum gewünschten Ergebnis führt.

Wie in Abbildung 4.2d gezeigt wird, entsteht durch die Anpassung der Sollgröße kein negativer Einfluss bei Tischen ohne dem beschriebenen Problem, da ohne einer Abweichung von mindestens 10 rad die Adaption nicht aktiv wird.

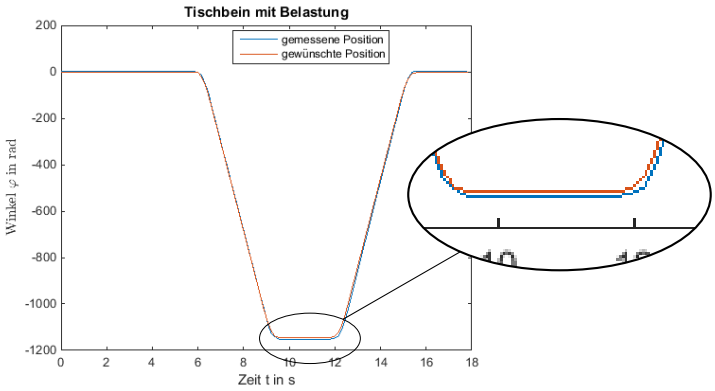


(a) ohne Belastung alter Tisch

(b) mit Belastung alter Tisch



(c) ohne Belastung neuer Tisch



(d) mit Belastung neuer Tisch

Abbildung 4.2.: Test des bestehenden PI-Reglers mit Sollgrößenanpassung

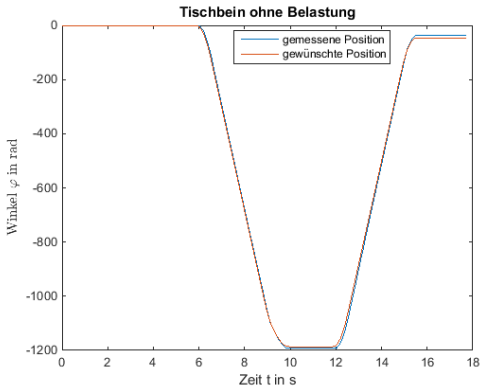
4.3. Test des PI-Reglers mit Anpassung der Abbrems- und Beschleunigungszeit

Durch Anpassen der Abbremszeit wird ein Positionsunterschied verringert und durch Anpassen der Beschleunigungszeit kann ein „Hüpfen“ verringert werden. Diese Adaption wird ebenfalls erst aktiv, sobald die gemessene von der gewünschten Position um mindestens 10 rad abweicht, jedoch nur wenn sich der Tisch im Abbrems- beziehungsweise Beschleunigungszustand befindet. Bei dieser Art der Adaption fährt das unbelastete Tischbein geringfügig weiter als vom Anwender gewünscht. Dies könnte möglicherweise ein Sicherheitsproblem darstellen.

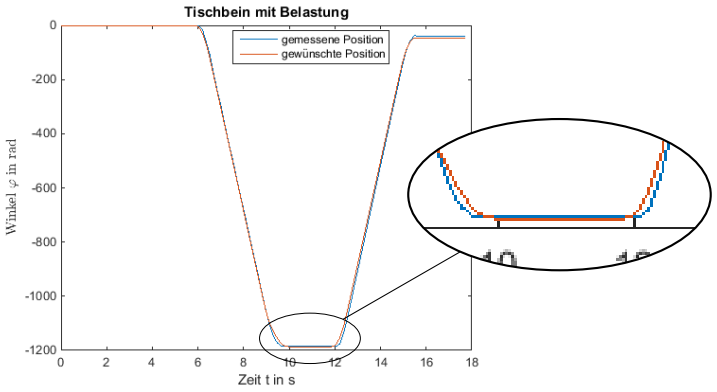
Die Adaption wird bei Tischen, bei denen kein Auseinanderlaufen der Antriebe stattfindet, nicht aktiv, und verändert somit das Verhalten dieser Tische nicht (siehe Abbildung 4.3d).

4.4. Test des P-Reglers mit Vorsteuerung

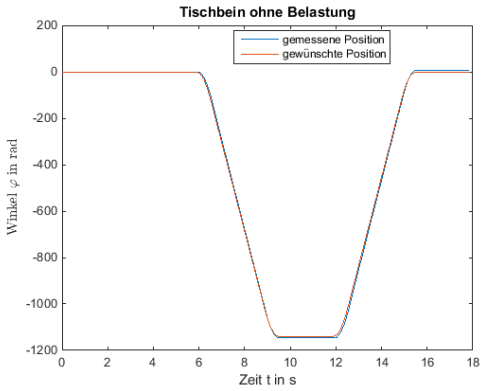
In Abbildung 4.4d ist der Verlauf der Antriebe mit der entworfenen Vorsteuerung dargestellt. Es ist erkennbar, dass der neue Tisch der gewünschten Position sehr genau folgen kann. Dies ist der erste Algorithmus, der beim neuen Tisch Verbesserungen erbracht hat.
Auch beim alten Tisch folgt der Tisch mit Vorsteuerung meistens der Sollposition besser als beim bestehenden PI-Regler (siehe Abbildung 4.4b). Nur beim Abbremsen des belasteten Antriebs entsteht wieder das bekannte Problem. In der praktischen Überprüfung der Vorsteuerung wurden jedoch die Parameter des neuen Tisches verwendet, welche wahrscheinlich stark von den Parametern des alten Tisches abweichen. Ob das Problem beim alten Tisch mittels Vorsteuerung behoben werden kann, kann nicht geklärt werden, da die dazugehörigen Parameter für die Vorsteuerung nicht identifiziert wurden.



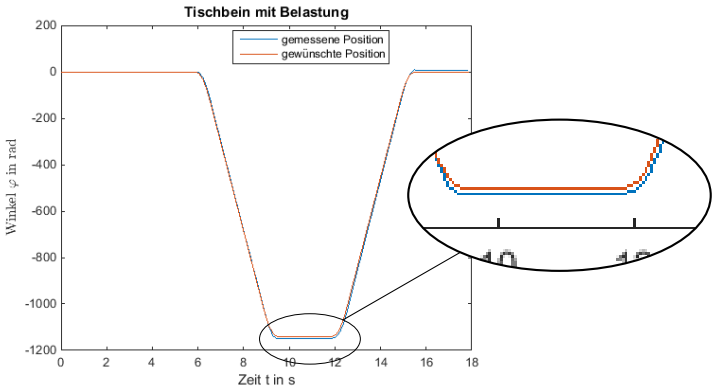
(a) ohne Belastung alter Tisch



(b) mit Belastung alter Tisch

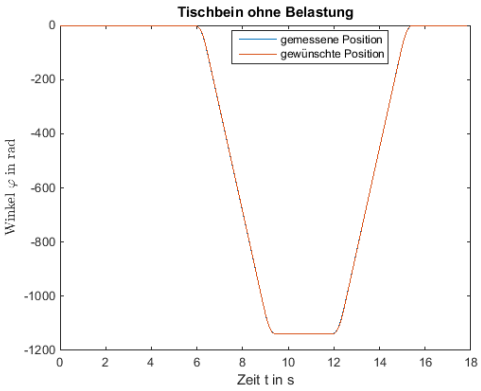


(c) ohne Belastung neuer Tisch

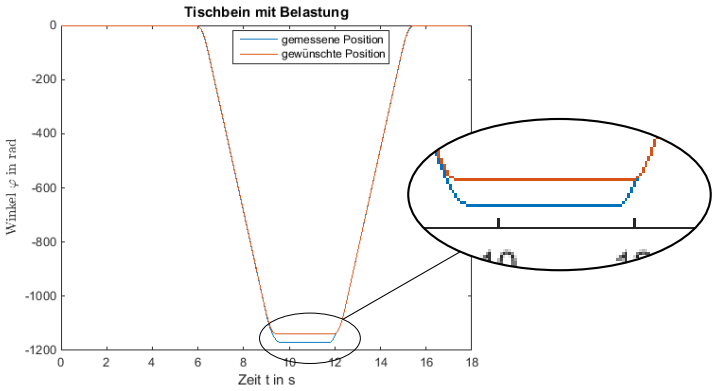


(d) mit Belastung neuer Tisch

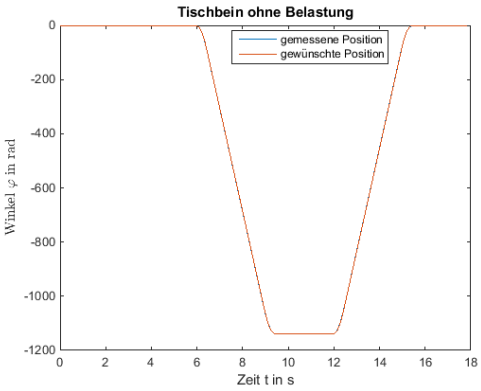
Abbildung 4.3.: Test des bestehenden PI-Reglers mit Anpassung der Abbrems- und Beschleunigungszeit



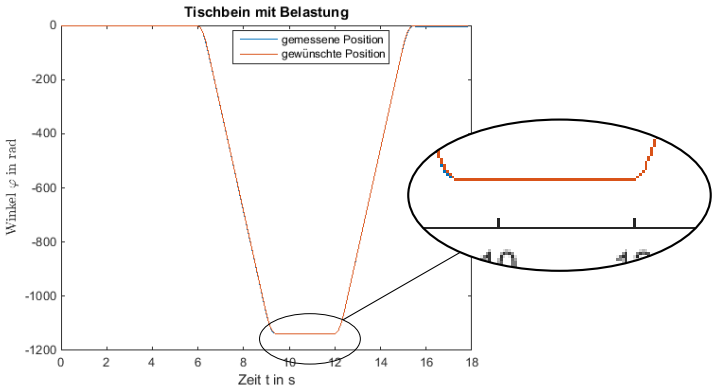
(a) ohne Belastung alter Tisch



(b) mit Belastung alter Tisch



(c) ohne Belastung neuer Tisch



(d) mit Belastung neuer Tisch

Abbildung 4.4.: Test des P-Reglers mit Vorsteuerung

4.5. Fazit

Durch Adaption der Sollposition sowie durch Anpassen der Abbrems- und Beschleunigungszeit kann dem Problem des „Hüpfens“ sehr gut entgegengewirkt werden. Mit Holfe der flachheitsbasierten Vorsteuerung kann zumindest das allgemeine Verhalten deutlich verbessert werden, ob das Auseinanderlaufen der Antriebe ebenso verhindert werden kann, muss weiter untersucht werden. Durch die Art der Regelalgorithmen können sie beliebig kombiniert werden. Es kann gezeigt werden, dass das allgemeine Verhalten durch die Vorsteuerung verbessert werden kann und durch Anpassen der Sollposition und/oder durch Anpassen der Abbrems- und Beschleunigungszeit die Positionsunterschiede vermindert werden können.

5. Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, einen neuen Regelalgorithmus für die Firma Logicdata, welche Steuerungen für elektrisch verstellbare Möbel baut und entwickelt, zu entwerfen. Bei dem aktuell verwendete PI-Regler kann es bei mechanisch eher schlechteren Antrieben oder bei sehr großen Toleranzen zwischen den zu synchronisierenden Antrieben beim Anfahren zu kurzen pendelnden Schwingungen kommen. Die Hauptanforderung des zu entwickelnden Algorithmus besteht darin, diese pendelnden Schwingungen zu verringern beziehungsweise für den Anwender nicht mehr spürbar zu machen. Eine weitere Anforderung an den Algorithmus ist es, dass dieser bei der vorhandenen Hardware implementiert werden kann. Somit steht zum Beispiel keine Strommessung zur Verfügung und es ist auch keine Richtungsänderung während des Regelungsvorgangs möglich, womit ein Überschwingen zu einem bleibenden Regelfehler führt.

Um dieses Ziel zu erreichen, wurden drei verschiedene Methoden untersucht. Zum einen wurde bei einem detektierten Überschwingen die Sollposition nachgeregelt, somit konnte ein Auseinanderlaufen der Antriebe verringert werden. Zum anderen wurde, wenn mindestens ein Antrieb zu weit über die Sollposition fährt, die Abbrems- und die Beschleunigungsdauer verlängert, womit der betroffene Antrieb mehr Zeit zum Abbremsen oder auch zum Beschleunigen zur Verfügung hat. Durch eine längere Abbremsdauer wird ein Auseinanderlaufen der Antriebe verringert und durch die längere zur Verfügung stehenden Zeit werden pendelnde Schwingungen reduziert. Die Idee der dritten Methode bestand darin, durch eine geeignete Vorsteuerung ein besseres Führungsverhalten zu erreichen und somit ein Auseinanderlaufen der Antriebe zu verhindern. Dazu mussten die Parameter eines Antriebs identifiziert werden, womit die Vorsteuerung entworfen werden konnte. Da der zu Anfang dieser Arbeit verwendete Tisch nicht mehr eingesetzt wird, sollte die Parameteridentifikation anhand eines neueren Tisches durchgeführt werden. Leider war zu diesem Zeitpunkt kein Tisch mit dem zu korrigierenden Problem der pendelnden Schwingungen

vorhanden. Deshalb wurde diese Regelung hauptsächlich auf Verhalten untersucht. Durch Testen der Vorsteuerung mit den identifizierten Parametern des neuen Tisches anhand des alten Tisches, wurde das Verhalten bei sehr groben Ungenauigkeiten der Parameter untersucht. Wurde der proportionale Regler auf den Wert 3,5 eingestellt, so wurde ein sehr gutes Verhalten beobachtet, jedoch konnten die pendelnden Schwinger des alten Tisches nicht verhindert werden.

Es ist zu untersuchen, ob dies auch bei einem Tisch mit den korrekten Parametern der Fall ist, oder ob die Probleme damit verhindert werden können. Eine weitere Möglichkeit, wie diese Probleme verhindert werden könnten, ist die Regelung der Position mit unterlagerter Stromregelung. Hier müsste außerdem ein Beobachter zur Ermittlung des Stroms entworfen werden.

Die in dieser Arbeit ermittelten Methoden zur Verhinderung pendelnder Schwingungen könnten auch beliebig kombiniert werden. Wie sich das auf das Verhalten von den neuen Antrieben mit dem Problem auswirkt, und ob sich sicherheitstechnische Probleme durch diese Implementierungen ergeben, muss in der Praxis mit der realen Steuerung untersucht werden.

Anhang

A. Arduino Due

A.1. Installation

Für die Verwendung von Arduino Due werden ein Treiber und ein zusätzliches Package für Simulink benötigt. Unter https://www.arduino.cc kann der benötigte Treiber heruntergeladen werden. Die Vorgehensweise zur Installation des zusätzlichen Packages für Simulink hängt von der verwendeten Matlab version ab. Es ist zu überprüfen, ob die Matlabversion mit Arduino Due kompatibel ist. Wenn dies der Fall ist, kann, hier am Beispiel von der Version 2014a, folgendermaßen vorgegangen werden:

1 Simulink starten
2 Unter „Tools“ / „Run on Target Hardware“ / „Install/Update Support Package... “ öffnen
3 „Install from Internet“
4 Unter „Arduino Due“ bei „Install“ das Häkchen setzten
5 In „MathWorks Account“ einloggen, (wenn noch keine Account besteht kann dieser sehr einfach erstellt werden)
6 Lizenzbedingungen bestätigen
7 Mit „next“ und anschließend „install“ wird das Package installiert und kann verwendet werden.

A.2. Einstellungen

Damit Simulink mit „Arduino Due“ kommunizieren kann, muss unter „Tools“ / „Run on Target Hardware“ / „Prepare to Run... “ geöffnet und unter „Target

hardware“ „Arduino Due“ ausgewählt werden. Manchmal wird der „COM Port“ nicht automatisch erkannt, dann muss unter „Set host COM port:“ „Manually“ ausgewählt und anschließend der von Arduino verwendete „COM Port“ (Port an dem USB Kabel angeschlossen ist) eingegeben werden. Dieser kann unter „Systemsteuerung“ / „Geräte-Manager“ / „Anschlüsse (COM & LPT)“ / „Arduino Due Programming Port (Gesuchter COM Port)“ gefunden werden.

Die Baudrate von Serial 0 sollte auf 115200 eingestellt werden, damit eine möglichst schnelle Übertragung erreicht werden kann. Dabei ist zu beachten, dass die Baudrate bei HTerm (siehe Kapitel 3.2.4.1) der selben wie der hier eingestellten entspricht.

A.3. Pin-Belegung

User Selectable Pins	Digital Input	0-53
	Digital Output	0-53
	Analog Input	0-11
	PWM	2-13
	Standard Servo Read	0-53
	Standard Servo Write	0-53
	Continuous Servo Write	0-53
	Serial Receive	Port 0: pin 0 Port 1: pin 19 Port 2: pin 17 Port 3: Pin 5
	Serial Transmit	Port 0: pin 1 Port 1: pin 18 Port 2: pin 6 Port 3:pin 4

Tabelle .1.: Pinbelegung Arduino Due

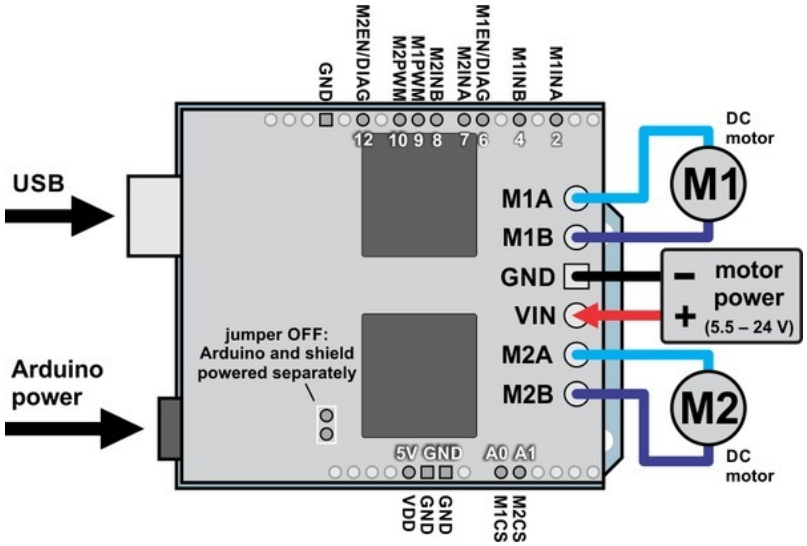


Abbildung .1.: Dual VNH5019 motor driver shield mit Arduino [Arduino, 2016a] NEU MACHEN

A.4. Motor driver shield

B. Matlab code

B.1. Sollgrößengenerator

```
function y  = fcn(u)

Ta = 0.001;
steigung = 380;%528.4706;
verzoegerung = 100;
bremsfaktor = 15;
sstime1 = 0.6;

persistent zaehler_steigen zaehler_sinken
persistent temp temp2 temp3
persistent begrenzung temp4
persistent sstime bremszustand bremszaehler
persistent zustand1 zustand2 zustand3 alt
```

```
14 neu = u(1);
15 ueberlauf = abs(u(2));
16
17 if isempty(zustand1)
18     zustand1 = 0;
19 end
20
21 if isempty(zustand2)
22     zustand2 = 0;
23 end
24
25 if isempty(zustand3)
26     zustand3 = 0;
27 end
28
29 if isempty(alt)
30     alt = 0;
31 end
32
33 if isempty(bremszaehler)
34     bremszaehler = 0;
35 end
36
37 if isempty(bremszustand)
38     bremszustand = 0;
39 end
40
41 if isempty(sstime)
42     sstime = 0.6;
43 end
44
45 if isempty(zaehler_steigen)
46     zaehler_steigen = 0;
47 end
48
49 if isempty(temp)
50     temp = 0;
51 end
52
53 if isempty(zaehler_sinken)
54     zaehler_sinken = 0;
55 end
56
57 if isempty(temp2)
```

```
58      temp2 = 0;
59  end
60
61  if isempty(temp3)
62      temp3 = 0;
63  end
64
65  if isempty(begrenzung)
66      begrenzung = 1;
67  end
68
69
70  if isempty(temp4)
71      temp4 = 0;
72  end
73
74
75  %% Begrenzung
76  if (ueberlauf > 0 && temp4 == 0 && begrenzung > 0)
77      temp4=verzoegerung;
78      begrenzung = begrenzung *0.9;
79  elseif (temp4 > 0)
80      temp4 = temp4 - 1;
81  end
82
83  %% Steigende Flanke erkennen (anfahren)
84  if neu == 1 && alt == 0  && zustand1 == 0 && zustand2 == 0 && zustand3 == 0
85      zaehler_steigen = 1;
86  end
87
88  if zaehler_steigen == 1
89      zustand1=1;
90      if temp < sstime/Ta;
91          temp = temp + 1;
92      else
93          zaehler_steigen = 0;
94          temp = 0;
95          zustand1 = 0;
96      end
97  end
98
99
100 %% Sinkende Flanke erkennen (abbremsen)
101 if alt == 1 && neu == 0 && zustand1 == 0 && zustand2 == 0
```

```
102     zaehler_sinken = 1;
103     zustand3 = 0;
104 end
105
106 if zaehler_sinken == 1
107     zustand2 = 1;
108     if temp2 < sstime/Ta;
109         temp2 = temp2 + 1;
110     else
111         zaehler_sinken = 0;
112         temp2 = 0;
113         zustand2 = 0;
114     end
115 end
116
117
118 %% Geschwindigkeit halten
119 if alt == 1 && neu == 1
120     temp3 = 1;
121 end
122
123 if temp3 == 1 && zaehler_steigen == 0 && zaehler_sinken == 0
124     zustand3 = 1;
125     temp3 = 0;
126 else
127     zustand3 = 0;
128 end
129
130
131 %% bestimmung der Sollgre
132 alt=neu;
133
134 y = steigung*begrenzung/sstime*(temp)*Ta*zustand1...
135     +begrenzung*steigung*zustand3...
136     +(steigung*begrenzung-steigung*begrenzung/sstime*temp2*Ta)*zustand2;
```

B.2. Hauptprogramm zur Parameteridentifikation

```
1 clear all; close all; clc;
2 %
3 %
4 load J0drk;
```

```
5
6  start(1) = J0;  %J0
7  start(2) = k;   %k
8  start(3) = dr;  %dr
9
10 m0=2.1; %Masse der Tischblatte
11
12 %% einlesen
13 Lstart = 1;
14 Lversuch = 42;
15
16 eingang = 0;
17 ausgang = 0;
18 zeit = 0;
19 zeit2 = 0;
20 mz = 0;
21 Ua = 0;
22
23 tmp_eingang = 0;
24 tmp_ausgang = 0;
25 tmp_zeit(1) = 0;
26 tmp_zeit2(1) = 0;
27 tmp_zeit1 = 0;
28
29 Ta = 1e-3;
30
31 for j = Lstart:Lversuch
32
33 [eingang1, ausgang1, zeit1, Ua1, mLast] = identifikation(j);
34 disp(j);
35 clear a;
36 a=find(abs(eingang1)>100); % Nur Hinauf oder Hinunter kein Positionshalten
37 % a=[1:length(eingang1)]; %Ganze Signale
38
39 eingang(length(eingang):length(eingang)+length(eingang1(a(1):a(end-1)))-1)...
40     = eingang1(a(1):a(end-1))' + tmp_eingang;
41 ausgang(length(ausgang):length(ausgang)+length(ausgang1(a(1):a(end-1)))-1)...
42     = ausgang1(a(1):a(end-1)) + tmp_ausgang;
43 zeit2(length(zeit2):length(zeit2)+length(zeit1(a(1):a(end-1)))-1)...
44     = zeit1(a(1):a(end-1))-zeit1(a(1)) + tmp_zeit2(j-Lstart+1);
45
46 mz(length(mz):length(mz)+length(zeit1(a(1):a(end-1)))-1) = mLast;
47 Ua(length(Ua):length(Ua)+length(zeit1(a(1):a(end-1)))-1) = Ua1;
48
```

```
49 tmp_eingang = 0;
50 tmp_ausgang = ausgang(end);
51 tmp_zeit(j-Lstart+2) = zeit(end)+Ta;
52 tmp_zeit2(j-Lstart+2) = zeit2(end)+Ta;
53 tmp_zeit1 = zeit(end)+Ta;
54 abschnitt(j-Lstart+2)=length(eingang);
55
56 end
57 clear zeit;
58 clear a;
59 zeit = zeit2;
60 Anz_Versuche = Lversuch-Lstart+1;
61
62 %% optimieren
63 % tic
64 paropt = optimierung1Finish(eingang, ausgang, zeit, mz,...
65     Ua, start, Anz_Versuche, abschnitt);
66 % toc
67 %%
68 J0 = paropt(1);
69 k = paropt(2);
70 dr = paropt(3);
71
72 ia0=0;
73 om0=0;
74 phi0=0.00;
75
76
77 Ra = 3.175;
78 La = 1.1778e-3;
79 km = 0.0344585;
80
81 g = 9.81;
82
83
84 eingangSim = [zeit' eingang'];
85 ausgangSim = [zeit' ausgang'];
86
87 UaSim = [zeit' Ua'];
88 mzSim = [zeit' mz'+m0];
89 %% Berechnen und Ploten der optimierten Werte
90 initx = [0 0 0];
91 minimize1Finish(paropt, zeit, initx, zeit, initx, ausgang, eingang, mz,...
92     Ua, Anz_Versuche,abschnitt);
```

B.3. Zuordnung von Versuchsnummer zu Spannung Masse und Dateinamen

```
function [eingang, ausgang, zeit, Ua, mLast] = identifikation(wahl)
%% Einlesen der Daten


if wahl == 1
    dateiName = 'rauf0kg10V.txt';
    verschiebung=4;
    Ua = 9.87;%bei 0kg
    mLast = 0;

elseif wahl == 2
    dateiName = 'runter0kg10V.txt';
    verschiebung=4;
    Ua = 9.92;%bei 0kg
    mLast = 0;

elseif wahl == 3
    dateiName = 'rauf5kg10V.txt';
    verschiebung=5;
    Ua = 9.84;%bei 5kg
    mLast = 5.151;

elseif wahl == 4
    dateiName = 'runter5kg10V.txt';
    verschiebung=6;
    Ua = 9.92;%bei 5kg
    mLast = 5.151;

elseif wahl == 5
    dateiName = 'rauf10kg10V.txt';
    verschiebung=6;
    Ua = 9.81;%bei 10kg
    mLast = 10.309;

elseif wahl == 6
    dateiName = 'runter10kg10V.txt';
    verschiebung=6;
    Ua = 9.92;%bei 10kg
    mLast = 10.309;
```

```
40
41 elseif wahl == 7
42     dateiName = 'rauf15kg10V.txt';
43     verschiebung=5;
44     Ua = 9.77;%bei 15kg
45     mLast = 15.408;
46
47 elseif wahl == 8
48     dateiName = 'runter15kg10V.txt';
49     verschiebung=7;
50     Ua = 9.93;%bei 15kg
51     mLast = 15.408;
52
53 elseif wahl == 9
54     dateiName = 'rauf20kg10V.txt';
55     verschiebung=6;
56     Ua = 9.74;%bei 20kg
57     mLast = 20.561;
58
59 elseif wahl == 10
60     dateiName = 'runter20kg10V.txt';
61     verschiebung=8;
62     Ua = 9.94;%bei 20kg
63     mLast = 20.561;
64
65 elseif wahl == 11
66     dateiName = 'rauf25kg10V.txt';
67     verschiebung=6;
68     Ua = 9.7;%bei 25kg
69     mLast = 25.512;
70
71 elseif wahl == 12
72     dateiName = 'runter25kg10V.txt';
73     verschiebung=6;
74     Ua = 9.94;%bei 25kg
75     mLast = 25.512;
76
77 elseif wahl == 13
78     dateiName = 'rauf30kg10V.txt';
79     verschiebung=5;
80     Ua = 9.66;%bei 30kg
81     mLast = 30.413;
82
83 elseif wahl == 14
```

```
84      dateiName = 'runter30kg10V.txt';
85      verschiebung=6;
86      Ua = 9.95;%bei 30kg
87      mLast = 30.413;
88
89  elseif wahl == 15
90      dateiName = 'rauf30kg15V.txt';
91      verschiebung=6;
92      Ua = 14.77;%17.77;%bei 30kg
93      mLast = 30.413;
94
95  elseif wahl == 16
96      dateiName = 'runter30kg15V.txt';
97      verschiebung=6;
98      Ua = 15.04;%bei 30kg
99      mLast = 30.413;
100
101 elseif wahl == 17
102     dateiName = 'rauf25kg15V.txt';
103     verschiebung=5;
104     Ua = 14.81;%bei 25kg
105     mLast = 25.512;
106
107 elseif wahl == 18
108     dateiName = 'runter25kg15V.txt';
109     verschiebung=6;
110     Ua = 15.02;%bei 25kg
111     mLast = 25.512;
112
113 elseif wahl == 19
114     dateiName = 'rauf20kg15V.txt';
115     verschiebung=6;
116     Ua = 14.85;%bei 20kg
117     mLast = 20.561;
118
119 elseif wahl == 20
120     dateiName = 'runter20kg15V.txt';
121     verschiebung=7;
122     Ua = 15.04;%bei 20kg
123     mLast = 20.561;
124
125 elseif wahl == 21
126     dateiName = 'rauf15kg15V.txt';
127     verschiebung=4;
```

```
128     Ua = 14.88;%bei 15kg
129     mLast = 15.408;
130 
131 elseif wahl == 22
132     dateiName = 'runter15kg15V.txt';
133     verschiebung=6;
134     Ua = 15.03;%bei 15kg
135     mLast = 15.408;
136 
137 elseif wahl == 23
138     dateiName = 'rauf10kg15V.txt';
139     verschiebung=6;
140     Ua = 14.92;%bei 10kg
141     mLast = 10.309;
142 
143 elseif wahl == 24
144     dateiName = 'runter10kg15V.txt';
145     verschiebung=6;
146     Ua = 15.02;%bei 10kg
147     mLast = 10.309;
148 
149 elseif wahl == 25
150     dateiName = 'rauf5kg15V.txt';
151     verschiebung=4;
152     Ua = 14.96;%bei 5kg
153     mLast = 5.151;
154 
155 elseif wahl == 26
156     dateiName = 'runter5kg15V.txt';
157     verschiebung=6;
158     Ua = 15.02;%bei 5kg
159     mLast = 5.151;
160 
161 elseif wahl == 27
162     dateiName = 'rauf0kg15V.txt';
163     verschiebung=6;
164     Ua = 15;%bei 0kg
165     mLast = 0;
166 
167 elseif wahl == 28
168     dateiName = 'runter0kg15V.txt';
169     verschiebung=6;
170     Ua = 15.01;%bei 0kg
171     mLast = 0;
```

```
172
173 elseif wahl == 29
174     dateiName = 'rauf0kg20V.txt';
175     verschiebung=6;
176     Ua = 19.99;%bei 0kg
177     mLast = 0;
178
179 elseif wahl == 30
180     dateiName = 'runter0kg20V.txt';
181     verschiebung=6;
182     Ua = 20.03;%bei 0kg
183    mLast = 0;
184
185 elseif wahl == 31
186     dateiName = 'rauf5kg20V.txt';
187     verschiebung=6;
188     Ua = 19.97;%bei 5kg
189     mLast = 5.151;
190
191 elseif wahl == 32
192     dateiName = 'runter5kg20V.txt';
193     verschiebung=6;
194     Ua = 20.03;%bei 5kg
195     mLast = 5.151;
196
197 elseif wahl == 33
198     dateiName = 'rauf10kg20V.txt';
199     verschiebung=6;
200     Ua = 19.93;%bei 10kg
201     mLast = 10.309;
202
203 elseif wahl == 34
204     dateiName = 'runter10kg20V.txt';
205     verschiebung=6;
206     Ua = 20.03;%bei 10kg
207     mLast = 10.309;
208
209 elseif wahl == 35
210     dateiName = 'rauf15kg20V.txt';
211     verschiebung=6;
212     Ua = 19.90;%bei 15kg
213     mLast = 15.408;
214
215 elseif wahl == 36
```

```
216     dateiName = 'runter15kg20V.txt';
217     verschiebung=5;
218     Ua = 20.03;%bei 15kg
219     mLast = 15.408;
220 
221 elseif wahl == 37
222     dateiName = 'rauf20kg20V.txt';
223     verschiebung=6;
224     Ua = 19.86;%bei 20kg
225     mLast = 20.561;
226 
227 elseif wahl == 38
228     dateiName = 'runter20kg20V.txt';
229     verschiebung=6;
230     Ua = 20.04;%bei 20kg
231     mLast = 20.561;
232 
233 elseif wahl == 39
234     dateiName = 'rauf25kg20V.txt';
235     verschiebung=6;
236     Ua = 19.83;%bei 25kg
237     mLast = 25.512;
238 
239 elseif wahl == 40
240     dateiName = 'runter25kg20V.txt';
241     verschiebung=6;
242     Ua = 20.04;%bei 25kg
243     mLast = 25.512;
244 
245 elseif wahl == 41
246     dateiName = 'rauf30kg20V.txt';
247     verschiebung=6;
248     Ua = 19.78;%bei 30kg
249     mLast = 30.413;
250 
251 elseif wahl == 42
252     dateiName = 'runter30kg20V.txt';
253     verschiebung=6;
254     Ua = 20.05;%bei 30kg
255     mLast = 30.413;
256 
257 else
258     error('Versuch existiert nicht (Versuche zwischen 1 & 42)');
259 end
```

```
260
261 [eingang, ausgang, zeit] = kompaktesDatenlesen(dateiName,verschiebung);
262
263 end
```

B.4. Einlesen der Daten aus Textdatei

```
1
2  function [eingang, ausgang, zeit2] ...
3      = kompaktesDatenlesen(DateiName,verschiebung)
4
5  anzahlSignale = 8;
6  bits = 3*anzahlSignale; % 8...uebertragene Signale
7  startbit = uint64('A');
8  stopbit = uint64('0');
9  %% Auslesen
10     % Initialisierung und Datei oeffnen
11 toadd = 0;
12 data = [];
13 n = 0; % n: Anzahl der schlechten Zeilen
14 fid = fopen(DateiName);
15 % nicht mehr als 5 aufeinanderfolgende "Fehlerzeilen" (oder Ende)
16 while n < 5
17     toadd = textscan(fid, '%f%f%f%f%f%f%f%f %*[^\n]','Delimiter',{':',';'});
18     if length(toadd{1}) > length(toadd{2})
19         toadd{2}(end+1) = 0;
20     end
21     if length(toadd{2}) > length(toadd{3})
22         toadd{3}(end+1) = 0;
23     end
24     if length(toadd{3}) > length(toadd{4})
25         toadd{4}(end+1) = 0;
26     end
27     if length(toadd{4}) > length(toadd{5})
28         toadd{5}(end+1) = 0;
29     end
30     if length(toadd{5}) > length(toadd{6})
31         toadd{6}(end+1) = 0;
32     end
33     if length(toadd{6}) > length(toadd{7})
34         toadd{7}(end+1) = 0;
35     end
```

```
36      if length(toadd{7}) > length(toadd{8})
37          toadd{8}(end+1) = 0;
38      end
39
40      if isempty(toadd{1})
41          fgetl(fid); % schlechte Zeile, eine ueberspringen
42          n = n + 1;
43      else
44          data = [data; toadd{1:8}]; % gute Zeile, Daten hinzufuegen
45          n = 0;
46
47      end
48  end
49  fclose(fid);
50
51  %% Alles hintereinander in einer Wurst schreiben und die NAN's weg lassen
52  k = 0;
53  for i=1:length(data)/2
54      for j = 1:length(data(1,:))
55          if isnan(data(i*2,j)) %NAN's weg lassen
56              k = k+1;
57          else
58              wurst(i+(i-1)*(length(data(1,:))-1)+j-1-k) = data(i*2,j);
59              zeitwurst(i+(i-1)*(length(data(1,:))-1)+j-1-k) = data(2*i-1,j);
60          end
61      end
62  end
63
64  %% Aus der Wurst wieder eine Matrix machen, und das in Korrekter Reihenfolge!
65  j = 1;
66  k = 1;
67  for i = 1:length(wurst)
68      if wurst(i) == 65
69          k = 1;
70          j = j+1;
71          matr(j,k) = wurst(i);
72          zeit(j,k) = zeitwurst(i);
73      else
74          k = k+1;
75          matr(j,k) = wurst(i);
76          zeit(j,k) = zeitwurst(i);
77      end
78  end
79
```

```
k=0;
for i = 2:length(matr)
    if matr(i,8) ~= 48 || (matr(i,6) ~= 0 && matr(i,6) ~= 1)...
            || ~isnan(zeit(i,4)) || isnan(zeit(i,3))
        k = k+1;
        a = 1;
    else
        matr1(i-k,1:8) = matr(i,1:8);
        zeit1(i-k,1:8) = zeit(i,1:8);
        a = 0;
    end
    if a == 0
        if i < length(zeit)
            if abs(zeit(i+1,3)-zeit(i,3))>0.5
                k = k+1;
            else
                matr1(i-k,1:8) = matr(i,1:8);
                zeit1(i-k,1:8) = zeit(i,1:8);
            end
        else
            if abs(zeit(i-1,3)-zeit(i,3))>0.5
                k = k+1;
            else
                matr1(i-k,1:8) = matr(i,1:8);
                zeit1(i-k,1:8) = zeit(i,1:8);
            end
        end
    end
end

for i = 2:length(matr1)
    bin1 = dec2bin(matr1(i,2));
    bin2 = dec2bin(matr1(i,3));
    bin3 = dec2bin(matr1(i,4));
    bin4 = dec2bin(matr1(i,5));
    temp = zeros(1,8-length(bin1));
    temp = dec2bin(temp)';
    temp1 = zeros(1,8-length(bin2));
    temp1 = dec2bin(temp1)';
    temp2 = zeros(1,8-length(bin3));
    temp2 = dec2bin(temp2)';
    bin3 = [bin4 temp2 bin3 temp1 bin2 temp bin1];
    Signal_neu(i-1) = bin2dec(bin3);
end
```

```
124 % Die Richtung so umrechnen, damit man sie besser verwenden kann
125 for i= 1:length(matr1)
126     if matr1(i,6) == 0
127         matr1(i,6) = -1;
128     else
129         matr1(i,6) =1;
130     end
131 end
132
133 %%fuer die Zeitachse in sec umrechnen
134 min1 = [];
135 min2 = [];
136 if zeit1(round(length(zeit1)/2)+1,3)-zeit1(round(length(zeit1)/2),3) <1
137     diff = zeit1(round(length(zeit1)/2)+1,3)-zeit1(round(length(zeit1)/2),3);
138 else
139     diff = zeit1(round(length(zeit1)/2),3)-zeit1(round(length(zeit1)/2)+1,3);
140 end
141
142 for i = 2:length(zeit1)-verschiebung
143
144     if isempty(min1)
145         min1 = zeit1(i,3);
146     end
147     if isempty(min2)
148         min2 = 0;
149     end
150     if zeit1(i,3) <= 0.02
151         if abs(zeit1(i+1,3)-zeit1(i,3))>=1
152             min1 = min1 +diff;
153         else
154             min1 = min1 +abs(zeit1(i+1,3)-zeit1(i,3));
155         end
156     else
157         if abs(zeit1(i+1,3)-zeit1(i,3))>=1
158             min1 = min1 +diff;
159         else
160             min1 = min1 +abs(zeit1(i+1,3)-zeit1(i,3));
161         end
162     end
163      if min1 +(zeit1(i+1,3)-zeit1(i,3))*2 >= 60
164         min1 = 0;
165         min2 = min2 +1;
166     end
167     zeit2(i-1) = zeit1(i+verschiebung,3)+min2*60;
```

```
168 end
169
170 eingang = -matr1(2:end,7).*matr1(2:end,6);
171 ausgang = Signal_neu./10000-1000;
172 for i = 1:length(ausgang)
173     if ausgang(i) >= 1e5
174         ausgang(i) = ausgang(i)-2^32/10000+1;
175     end
176 end
177
178 eingang = eingang(1:end-verschiebung);
179 ausgang = ausgang(1:end-verschiebung);
180 end
```

B.5. Optimierung mittels kleinster Fehlerquadrate

```
1  function paropt = optimierung1Finish(eingang, ausgang,...
2      zeit, mz, Ua, start, Anz_Versuche, abschnitt)
3
4  %Anfangswerte
5  ia0=0;
6  om0=0;
7  phi0=0.00;
8
9  %Anfangswerte der zu optimierenden Gren
10 para(1) = start(1); %J0
11 para(2) = start(2); %k
12 para(3) = start(3); %dr
13
14 %Obere und Untere Grenzen festlegen
15 lb = [1e-9 1e-9 0];
16 ub = [0.001 0.1 0.1];
17
18 for i = 1:length(zeit)-1
19     if zeit(i+1)-zeit(i)<0
20         disp('Fehler')
21     end
22 end
23
24 initx = [ia0 phi0 om0];
25 options = saoptimset('Display','iter','TolFun',1e-7,'TimeLimit',100000,...
26     'PlotFcn',{@saplotbestx,@saplotbestf,@saplotx,@saplotf},...
```

```
27      'ReannealInterval',1,'MaxFunEvals',100000,'MaxIter',100000);
28
29  funktion_opt = @(para)minimize1Finish(para, zeit, initx, zeit, initx,...
30      ausgang, eingang, mz, Ua, Anz_Versuche, abschnitt);
31
32  paropt = simulannealbnd(funktion_opt,para,lb,ub,options);
```

B.6. Minimierung

```
1
2  function lsq=minimize1Finish(para, t1, x, t, initx, ausgang, eingang,...
3      mz, Ua, Anz_Versuche, abschnitt)
4
5  options = odeset('RelTol',0.001,'AbsTol',0.001);
6  [t1,x] = ode15s(@vdp1Finish,t,initx,options,para,eingang,...
7      mz,Ua,t,Anz_Versuche,abschnitt);
8
9  figure(2); %Darstellung der Funktion mit den aktuellen Werten
10 plot(t1,x(:,2)); hold on; plot(t1,ausgang,'r'); hold off;
11 xlabel('Zeit in s');
12 ylabel('Position in rad');
13 title('Vergleich Position gemessen und simuliert');
14 legend('position simuliert','position gemessen');
15
16 lsq = mean((x(:,2)' - ausgang).^2);
```

B.7. Lösen der Differentialgleichungen

```
1  function dxdt = vdp1Finish(t,x,para,eingang,mz1,...
2      Ua1,zeit,Anz_Versuche,abschnitt)
3
4  Ra = 3.175;
5  La = 1.1778e-3;
6  km = 0.0344585;
7
8  m0 = 2.1;
9  g = 9.81;
10
11 J0 = para(1);
12 k = para(2);
```

```
13 dr = para(3);
14
15 %Eingangsgre ermitteln
16 a=find(t<=zeit);
17 p=sign(eingang(a(1)));
18
19
20
21
22 %Masse mz und Spannung Ua den Versuchen zuordnen
23 for i = 1:Anz_Versuche
24     if i == 1 && i~= Anz_Versuche
25         %Erster Versuch
26         if t >= zeit(abschnitt(i)+1) && t <= zeit(abschnitt(i+1)+1)
27             mz(i) = mz1(abschnitt(i)+1)+m0;
28             Ua(i) = Ua1(abschnitt(i)+1);
29             j = i;
30         end
31     elseif i == Anz_Versuche && i ~= 1 && t >= zeit(abschnitt(i))...
32             && t <= zeit(abschnitt(i+1)) %Letzter Versuch
33         mz(i) = mz1(abschnitt(i)+1)+m0;
34         Ua(i) = Ua1(abschnitt(i)+1);
35         j = i;
36     elseif Anz_Versuche == 1 && t >= zeit(abschnitt(i)+1)...
37             && t <= zeit(abschnitt(i+1))
38         mz(i) = mz1(abschnitt(i)+1)+m0;
39         Ua(i) = Ua1(abschnitt(i)+1);
40         j = i;
41         %nicht erster und nicht letzer Versuch
42     elseif t >= zeit(abschnitt(i)) && t <= zeit(abschnitt(i+1)+1)
43         mz(i) = mz1(abschnitt(i)+1)+m0;
44         Ua(i) = Ua1(abschnitt(i)+1);
45         j = i;
46     end
47     if t>= max(zeit)
48         mz(i) = mz1(abschnitt(i)+1)+m0;
49         Ua(i) = Ua1(abschnitt(i)+1);
50         j = Anz_Versuche;
51     end
52 end
53
54 dxdt = zeros(3,1);
55
56 dxdt(1) = Ua(j)/La*p-Ra/La*x(1)-x(3)*km/La ;
```

```
dxdt(2) = x(3) ;
dxdt(3) = x(1)*km*1/(J0+k^2.*mz(j))-k*g.*mz(j)/(J0+k^2.*mz(j))-...
    dr/(J0+k^2.*mz(j))*x(3);

if p == 0 % Position halten bei Stillstand (Haftreibung)
    dxdt(1) = 0;
    dxdt(2) = 0;
    dxdt(3) = 0;
end
```

B.8. Anpassen der Beschleunigungs- und Abbremszeit

```
function y  = fcn(u)

Ta = 0.001;
steigung = 380;%528.4706;
verzoegerung = 100;
bremsfaktor = 10;
sstime1 = 0.6;

persistent zaehler_steigen zaehler_sinken
persistent temp temp2 temp3
persistent begrenzung temp4
persistent sstime bremszustand bremszaehler
persistent zustand1 zustand2 zustand3 alt

neu = u(1);
ueberlauf = abs(u(2));
ist_pos1 = u(3);
ist_pos2 = u(4);
soll_pos = u(5);

if isempty(zustand1)
    zustand1 = 0;
end

if isempty(zustand2)
    zustand2 = 0;
end

if isempty(zustand3)
    zustand3 = 0;
```

```
31 end
32
33 if isempty(alt)
34     alt = 0;
35 end
36
37 if isempty(bremszaehler)
38     bremszaehler = 0;
39 end
40
41 if isempty(bremszustand)
42     bremszustand = 0;
43 end
44
45 if isempty(sstime)
46     sstime = 0.6;
47 end
48
49 if isempty(zaehler_steigen)
50     zaehler_steigen = 0;
51 end
52
53 if isempty(temp)
54     temp = 0;
55 end
56
57 if isempty(zaehler_sinken)
58     zaehler_sinken = 0;
59 end
60
61 if isempty(temp2)
62     temp2 = 0;
63 end
64
65 if isempty(temp3)
66     temp3 = 0;
67 end
68
69 if isempty(begrenzung)
70     begrenzung = 1;
71 end
72
73
74 if isempty(temp4)
```

```
75      temp4 = 0;
76  end
77
78
79  %% Begrenzung
80  if (ueberlauf > 0 && temp4 == 0 && begrenzung > 0)
81      temp4=verzoegerung;
82      begrenzung = begrenzung *0.9;
83  elseif (temp4 > 0)
84      temp4 = temp4 - 1;
85  end
86
87  %% Steigende Flanke erkennen (anfahren)
88  if neu == 1 && alt == 0  && zustand1 == 0 && zustand2 == 0 && zustand3 == 0
89      zaehler_steigen = 1;
90  end
91
92  if zaehler_steigen == 1
93      zustand1=1;
94      if temp < sstime/Ta;
95          temp = temp + 1;
96      else
97          zaehler_steigen = 0;
98          temp = 0;
99          zustand1 = 0;
100     end
101 end
102
103
104 %% Sinkende Flanke erkennen (abbremsen)
105 if alt == 1 && neu == 0 && zustand1 == 0 && zustand2 == 0
106     zaehler_sinken = 1;
107     zustand3 = 0;
108 end
109
110 if zaehler_sinken == 1
111     zustand2 = 1;
112     if temp2 < sstime/Ta;
113         temp2 = temp2 + 1;
114     else
115         zaehler_sinken = 0;
116         temp2 = 0;
117         zustand2 = 0;
118     end
```

```
119 end
120
121
122 %% Geschwindigkeit halten
123 if alt == 1 && neu == 1
124     temp3 = 1;
125 end
126
127 if temp3 == 1 && zaehler_steigen == 0 && zaehler_sinken == 0
128     zustand3 = 1;
129     temp3 = 0;
130 else
131     zustand3 = 0;
132 end
133
134 %% Zu schnelles abbremsen erkennen
135 if (abs(ist_pos1 - soll_pos) >bremsfaktor || ...
136         abs(ist_pos2 - soll_pos) >bremsfaktor) &&...
137         zustand2 == 1 && bremszustand == 0 && sstime < 1.5
138     bremszustand = 1;
139     bremszaehler = 100;
140     sstime = sstime + 0.1;
141 end
142
143 if bremszustand == 1 && bremszaehler > 0
144     bremszaehler = bremszaehler - 1;
145 else
146     bremszustand = 0;
147 end
148
149 alt=neu;
150 %% Ausgabe
151 y = steigung*begrenzung/sstime*(temp)*Ta*zustand1...
152     +begrenzung*steigung*zustand3...
153     +(steigung*begrenzung-steigung*begrenzung/sstime*temp2*Ta)*zustand2;
```

B.9. Anpassen der Beschleunigungs- und Abbremszeit

```
1 function y  = fcn(u)
2
3 Ta = 0.001;
4 steigung = 380;%528.4706;
```

```
5  verzoegerung = 100;
6  bremsfaktor = 15;
7  sstime = 0.6;
8
9  persistent zaehler_steigen zaehler_sinken
10 persistent temp temp2 temp3
11 persistent begrenzung temp4
12 persistent bremszustand bremszaehler
13 persistent zustand1 zustand2 zustand3 alt
14 neu = u(1);
15 ueberlauf = abs(u(2));
16
17 if isempty(zustand1)
18     zustand1 = 0;
19 end
20
21 if isempty(zustand2)
22     zustand2 = 0;
23 end
24
25 if isempty(zustand3)
26     zustand3 = 0;
27 end
28
29 if isempty(alt)
30     alt = 0;
31 end
32
33 if isempty(bremszaehler)
34     bremszaehler = 0;
35 end
36
37 if isempty(bremszustand)
38     bremszustand = 0;
39 end
40
41 if isempty(zaehler_steigen)
42     zaehler_steigen = 0;
43 end
44
45 if isempty(temp)
46     temp = 0;
47 end
48
```

```
49 if isempty(zaehler_sinken)
50     zaehler_sinken = 0;
51 end
52 
53 if isempty(temp2)
54     temp2 = 0;
55 end
56 
57 if isempty(temp3)
58     temp3 = 0;
59 end
60 
61 if isempty(begrenzung)
62     begrenzung = 1;
63 end
64 
65 
66 if isempty(temp4)
67     temp4 = 0;
68 end
69 
70 
71 %% Begrenzung
72 if (ueberlauf > 0 && temp4 == 0 && begrenzung > 0)
73     temp4=verzoegerung;
74     begrenzung = begrenzung *0.9;
75 elseif (temp4 > 0)
76     temp4 = temp4 - 1;
77 end
78 
79 %% Steigende Flanke erkennen (anfahren)
80 if neu == 1 && alt == 0  && zustand1 == 0 && zustand2 == 0 && zustand3 == 0
81     zaehler_steigen = 1;
82 end
83 
84 if zaehler_steigen == 1
85     zustand1=1;
86     if temp < sstime/Ta;
87         temp = temp + 1;
88     else
89         zaehler_steigen = 0;
90         temp = 0;
91         zustand1 = 0;
92     end
```

```
93  end
94
95
96  %% Sinkende Flanke erkennen (abbremsen)
97  if alt == 1 && neu == 0 && zustand1 == 0 && zustand2 == 0
98      zaehler_sinken = 1;
99      zustand3 = 0;
100 end
101
102 if zaehler_sinken == 1
103     zustand2 = 1;
104     if temp2 < sstime/Ta;
105         temp2 = temp2 + 1;
106     else
107         zaehler_sinken = 0;
108         temp2 = 0;
109         zustand2 = 0;
110     end
111 end
112
113
114 %% Geschwindigkeit halten
115 if alt == 1 && neu == 1
116     temp3 = 1;
117 end
118
119 if temp3 == 1 && zaehler_steigen == 0 && zaehler_sinken == 0
120     zustand3 = 1;
121     temp3 = 0;
122 else
123     zustand3 = 0;
124 end
125
126
127 %% bestimmung der Sollgre
128 alt=neu;
129
130 y =   begrenzung*steigung*(3*(temp*Ta/sstime)^2-2*(temp*Ta/sstime)^3)*zustand1...
131      +begrenzung*steigung*zustand3...
132      +(steigung*begrenzung- begrenzung*steigung*(3*(temp2*Ta/sstime)^2-2*(temp2*Ta/
```

Literatur

Albach, Manfred (2011). *Grundlagen der Elektrotechnik 2 - Periodische und nicht periodische Signalformen*. 2.Auflage. Pearson (siehe S. 18).

Arduino (2016a). *Arduino Due Documentation*. URL: https://www.arduino.cc/en/Main/ArduinoBoardDue (siehe S. 38, 80).

Arduino (2016b). *Arduino Due Guide*. URL: https://www.arduino.cc/en/Guide/ArduinoDue (siehe S. 40).

Michel, Manfred (2011). *Leistungselektronik - Einführung in Schaltungen und deren Verhalten*. 5.Auflage. Springer (siehe S. 6, 7).

Plaßmann, Wilfried und Detlef Schulz (2013). *Handbuch Elektrotechnik - Grundlagen und Anwendungen für Elektrotechniker*. 6.Auflage. Springer (siehe S. 17).

Pololu. *Pololu Dual VNH5019 Motor Driver Shield User's Guide*. URL: https://www.pololu.com/docs/0J49 (siehe S. 38).

Rothfuß, Ralf, Joachim Rudolph und Michael Zeitz (1997). »Flachheit: ein neuer Zugang zur Steuerung und Regelung nichtlinearer Systeme«. In: *at-Automatisierungstechnik* 45.11, S. 517–525 (siehe S. 30, 35).

Svaricek, Ferdinand (2006). »Nulldynamik linearer und nichtlinearer Systeme: Definitionen, Eigenschaften und Anwendungen (Zero Dynamics of Linear and Nonlinear Systems: Definitions, Properties and Applications)«. In: *at-Automatisierungstechnik* 54.7/2006, S. 310–322.