Technische Universität Dresden
Fakultät Elektrotechnik und Informationstechnik
Institut fürr Regelungs- und Steuerungstheorie
Prof. Dr.-Ing. habil. Dipl.-Math. K. Röbenack

RST

Wintersemester 2019/20

Anleitung für das Praktikum Regelungstechnik

Versuch V21:
Positionsregelung eines Schwebekörpers

Versuchsziel: Einstellen eines PID-Reglers nach verschiedenen Methoden am Beispiel der Positionsregelung eines Schwebekörpers. Experimentelle Bestimmung der Werte von physikalischen Parametern, Identifikation der Parameter von Übertragungsfunktionen, nichtlineare Steuerung.

1. Einführung

Dieser Praktikumsversuch beschäftigt sich mit der Positionsregelung eines Schwebekörpers. Der entsprechende Versuchsstand besteht aus einer vertikal stehenden Plexiglasröhre, in der sich eine Styroporkugel befindet, vergl. Abbildung 1. Mittels eines am unteren Ende der Röhre angebrachten Lüfters wird ein nach oben gerichteter Luftstrom eingebracht, welcher die Kugel bei genügender Stärke emporhebt. Ein am oberen Ende der Röhre befindlicher Ultraschallsensor wird zur Bestimmung der vertikalen Position der Kugel in der Röhre verwendet. Ziel ist es, mit Hilfe eines Reglers und einer Steuerung den Luftstrom so zu manipulieren, dass eine festgelegte Position des Balls eingehalten wird bzw. die Ballposition einer vorgegebenen Trajektorie folgt. Darüber hinaus kann der bereitgestellte Luftstrom durch ein unterhalb des Lüfters angebrachtes Blech gestört werden, sodass das Störverhalten der Regelung untersucht und auf den Zielkonflikt zwischen Führungs- und Störverhalten eingegangen werden kann.

Zur experimentellen wie modellbasierten Bestimmung der Reglerparameter wird auf die in der Praktikumseinführung V0 beschriebenen Methoden zurückgegriffen, und zwar im einzelnen: Handeinstellung eines PID-Reglers (siehe dort Abschnitt 6.3), experimentelles Verfahren nach Ziegler und Nichols (siehe dort Abschnitt 6.4.1), Betragsoptimum/ Symmetrisches Optimum (siehe dort Abschnitt 6.5).

Im Rahmen der Versuchsvorbereitung und -durchführung wird die Programmiersprache Python (Version 3.5) verwendet, um die Auswertung der experimentell gewonnenen Daten zur Parameteridentifikation zu ermöglichen. Die Installation von Python-Programmpaketen auf

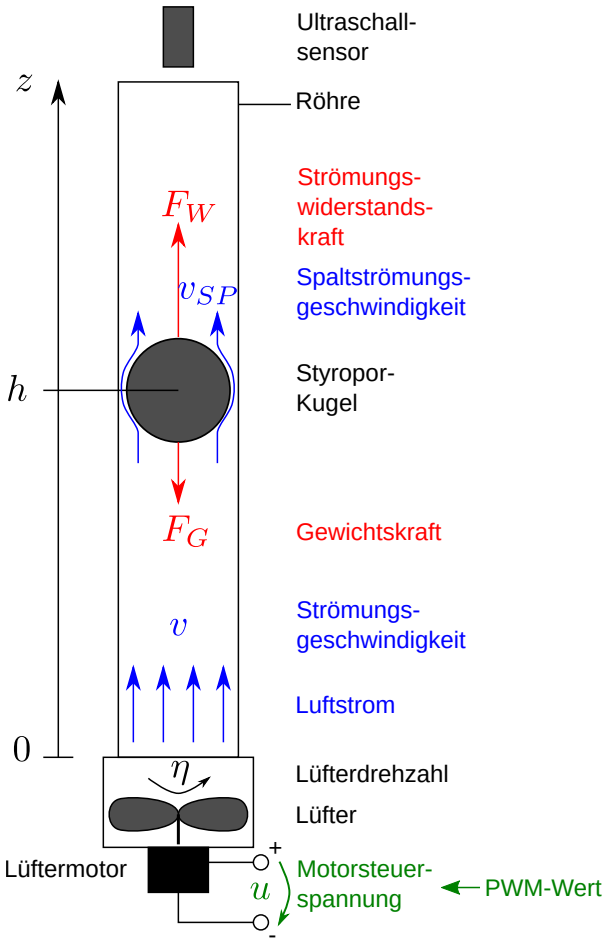

Abbildung 1: Skizze des Schwebekörperversuchsstandes.

Ihren privaten Rechnern ist dazu *nicht* erforderlich, da die entsprechenden Rechnungen webbasiert bzw. auf den im Praktikum verwendeten Rechnern erfolgen. Alle hierzu erforderlichen Kenntnisse werden in der vorliegenden Praktikumsanleitung vermittelt.

2. Modellbildung

Aus der Betrachtung der Skizze in Abbildung 1 wird unmittelbar klar, welches das Modell des Systems in zwei Teilmodelle zerfällt: Zum einen ein Modell, dass das dynamische Verhalten der Ballposition h in Abhängigkeit von der Strömungsgeschwindigkeit v der Luft beschreibt, zum anderen ein Modell, welches das dynamische Verhalten der Strömungsgeschwindigkeit v in Abhängigkeit von der Lüftermotorspannung u – repäsentiert durch einen PWM[1]-Wert –

[1] Pulsweitenmodulation

widerspiegelt. Dabei wird davon ausgegangen, dass das Verhalten des Balls keinen Einfluss auf die vom Lüfter bereitgestellte Luftströmung hat. Diese Annahme ist bei nicht zu geringen Ballpositionen h gerechtfertigt.

2.1. Modell des Teilsystems "Ball"

Auf den Ball wirken im wesentlichen zwei Kräfte [1]: Die Gewichtskraft F_g und die Strömungswiderstandskraft F_w. Letztere ist dafür verantwortlich, dass der Ball emporgehoben wird, sofern diese die Gewichtskraft überschreitet[2]. Die Gewichtskraft berechnet sich mit der Masse m und der Erdbeschleunigung g zu

$$F_\mathrm{g} = mg. \tag{1}$$

Die Strömungswiderstandskraft setzt sich aus einer Komponente zusammen, die aus der Reibung der vorbeiströmenden Luft mit der Kugel resultiert und einer, die eine Folge der Druckdifferenz zwischen Ober- und Unterseite der Kugel ist. Letztere wird als Druckwiderstandskraft bezeichnet. Im hier vorliegenden Fall eines so genannten "voluminösen Körpers" kann die Reibungskraft vernachlässigt werden. Die Strömungswiderstandskraft entspricht damit der Druckwiderstandskraft und berechnet sich zu

$$F_\mathrm{w} = \frac{1}{2} c_\mathrm{w} \varrho_\mathrm{L} A_\mathrm{B} \tilde{v}^2 \tag{2}$$

mit dem Widerstandsbeiwert c_w, der Luftdichte ϱ_L, der Ballquerschnittsfläche A_B und der relativen Geschwindigkeit zwischen Kugel und dem sie umströmenden Medium $\tilde{v}$. Diese relative Geschwindigkeit setzt sich zusammen aus der Geschwindigkeit v_SP des vom Lüfter erzeugten Luftstroms (siehe Abbildung 1) und der Geschwindigkeit $\dot{h}_\mathrm{SP}$ des durch die sich bewegenden Kugel erzeugten Luftstroms, jeweils im Luftspalt:

$$\tilde{v} = v_\mathrm{SP} - \dot{h}_\mathrm{SP}.$$

Geht man idealisierend davon aus, dass die Dichte der Luft im Rohr und im Luftspalt identisch ist, so berechnet sich aufgrund der daraus resultierenden Konstanz des Volumenstroms v_SP in Abhängigkeit von der Strömungsgeschwindigkeit v im Rohr zu

$$v_\mathrm{SP} = \frac{A_\mathrm{R}}{A_\mathrm{SP}} v \tag{3}$$

[2] Desweiteren wirkt die statische Auftriebskraft, resultierend aus dem statischen Auftrieb, den der Ball durch die verdrängte Luft erfährt (Archimedisches Prinzip) und die dynamische Auftriebskraft, die dadurch entsteht, dass die Luft an unterschiedlichen Stellen des Balls unterschiedlich schnell entlangströmt. Die statische Auftriebskraft ist gegenüber F_g und F_w vernachlässigbar, die dynamische Auftriebskraft spielt aufgrund der symmetrischen Führung des Balls in der Röhre ebenfalls keine nennenswerte Rolle. Beide werden daher vernachlässigt.

mit der Innenrohrquerschnittsfläche A_R und der Luftspaltquerschnittsfläche $A_\mathsf{SP} = A_\mathsf{R} - A_\mathsf{B}$. Die Geschwindigkeit des durch die Bewegung des Balls erzeugten Luftstroms berechnet sich analog zu

$$\dot{h}_\mathsf{SP} = \frac{A_\mathsf{B}}{A_\mathsf{SP}} \dot{h} \tag{4}$$

mit der vertikalen Bewegungsgeschwindigkeit $\dot{h}$ des Balls. Gl. (2) lässt sich unter Verwendung von Gl. (3) und (4) wie folgt notieren:

$$F_\mathsf{w} = \frac{1}{2} c_\mathsf{w} \varrho_\mathsf{L} A_\mathsf{B} \left(\frac{A_\mathsf{R} v - A_\mathsf{B} \dot{h}}{A_\mathsf{SP}} \right)^2 . \tag{5}$$

Der Volumenstrom $A_\mathsf{R} v$ des Lüfters ist eine recht abstrakte Größe. Besser greifbar ist die Drehzahl $\bar{\eta}$ des Lüfters. Der Zusammenhang zwischen beiden ist proportional mit dem Proportionalitätsfaktor k_V, sodass man anstelle von Gl. (5) auch schreiben kann:

$$F_\mathsf{w} = \frac{1}{2} c_\mathsf{w} \varrho_\mathsf{L} A_\mathsf{B} \left(\frac{k_\mathsf{V} \bar{\eta} - A_\mathsf{B} \dot{h}}{A_\mathsf{SP}} \right)^2 . \tag{6}$$

Ein zunächst problematisch erscheinender Parameter in dieser Formel ist der Widerstandsbeiwert c_w. Dieser hängt nichtlinear von der so genannten *Reynoldszahl* Re ab, vergl. Abbildung 2. Die dimensionslose Reynoldszahl beschreibt das Verhältnis von auf einen Körper wirkenden Druck- und Reibungskräften: Je größer die Reynoldszahl ist, desto stärker ist der Einfluss der Druckkräfte. Die Reynoldszahl berechnet sich in Abhängigkeit von der Strömungsgeschwindigkeit v, dem Kugeldurchmesser d_B und der kinematischen Viskosität ν von Luft zu

$$Re = \frac{v_\mathsf{SP} d_\mathsf{B}}{\nu}. \tag{7}$$

Um nun abschätzen zu können, in welchem Bereich sich der Widerstandsbeiwert für den vorliegenden Versuchsstand bewegt, ist die Reynoldszahl in Abhängigkeit von der vom Lüfter bereitgestellten Strömungsgeschwindigkeit zu berechnen. Dem Datenblatt des Lüfters (vergl. Anhang A, Tabelle 2) sind die Unter- und Obergrenzen $\dot{V}_\mathsf{min}$ und $\dot{V}_\mathsf{max}$ des Volumenstroms zu entnehmen. Dann muss die Reynoldszahl im Bereich

$$\frac{V_\mathsf{min} d_\mathsf{B}}{A_\mathsf{SP} \nu} \leq Re \leq \frac{V_\mathsf{max} d_\mathsf{B}}{A_\mathsf{SP} \nu} \tag{8}$$

liegen, wobei A_SP wiederum die Spaltquerschnittsfläche zwischen Kugel und Rohr ist. Man erhält ungefähr $Re \in [87000, 261624]$. Abbildung 2 kann man nun entnehmen, dass der Widerstandsbeiwert auf diesem Intervall annähernd konstant ist: $c_\mathsf{w} \approx 0.4$. Führt man nun noch den Parameter

$$k_\mathsf{L} := \frac{1}{2} c_\mathsf{w} \varrho_\mathsf{L} A_\mathsf{B}$$

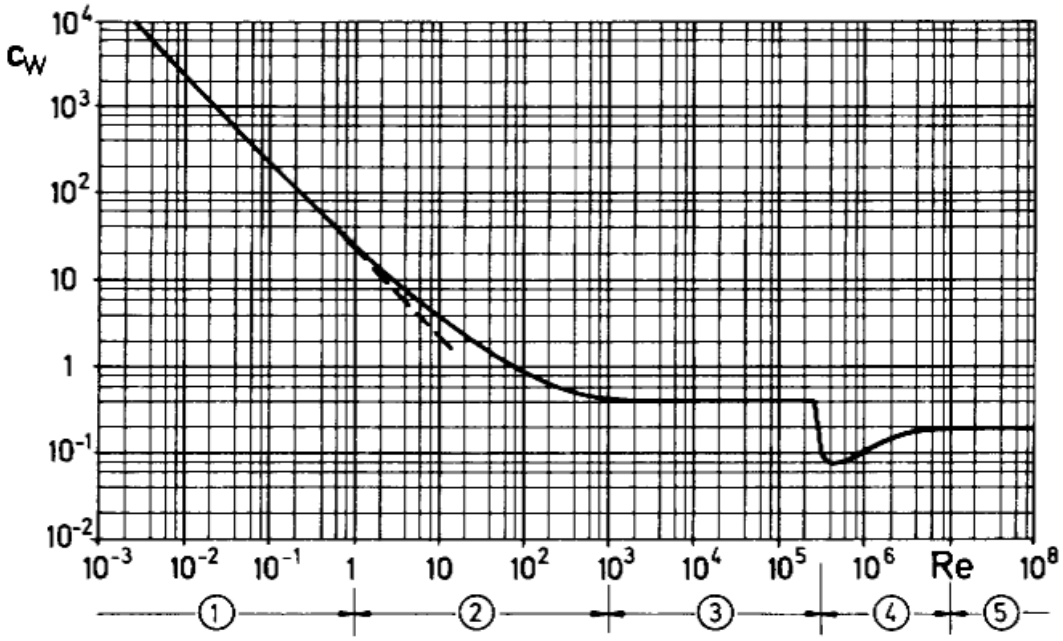

Abbildung 2: Widerstandsbeiwert c_w in Abhängigkeit von der Reynoldszahl für eine umströmte Kugel nach [1].

ein, ergibt sich als endgültige Gleichung für die Strömungswiderstandskraft

$$F_\mathrm{w} = k_\mathrm{L} \left(\frac{k_\mathrm{V}\bar{\eta} - A_\mathrm{B}\dot{h}}{A_\mathrm{SP}} \right)^2, \tag{9}$$

sodass sich damit und Gl. (1) die vertikale Kräftebilanz am Ball aufstellen lässt:

$$m\ddot{h} = k_\mathrm{L} \left(\frac{k_\mathrm{V}\bar{\eta} - A_\mathrm{B}\dot{h}}{A_\mathrm{SP}} \right)^2 - mg, \tag{10}$$

woraus die Differenzialgleichung zur Beschreibung der Dynamik der Ballposition h folgt:

$$\ddot{h} = \frac{k_\mathrm{L}}{m} \left(\frac{k_\mathrm{V}\bar{\eta} - A_\mathrm{B}\dot{h}}{A_\mathrm{SP}} \right)^2 - g. \tag{11}$$

2.2. Modellgleichungen des Teilsystems “Lüfter”

Am Versuchsstand erzeugt ein handelsüblicher PC-Lüfter mit Pulsweitenmodulation (PWM)-Ansteuerung und Tachosignal den erforderlichen Luftstrom. Wegen der PWM-Ansteuerung

gilt für die Stellgröße u_{PWM} des Motors:

$$u_{\mathrm{PWM}} \in \{0, 1, \ldots, 255\}. \tag{12}$$

Zu beachten ist, dass der Lüfter für $u_{\mathrm{PWM}} = 0$ bereits eine Grunddrehzahl η_0 aufweist. Das Teilmodell des Lüfters soll nun den dynamischen Zusammenhang zwischen dem am Lüftermotor angelegten PWM-Wert u_{PWM} und der Differenzdrehzahl

$$\eta := \bar{\eta} - \eta_0 \tag{13}$$

beschreiben. Am einfachsten ist es im hier vorliegenden Fall für das Modell PT_1-Verhalten mit der Zeitkonstanten T_{M} und der Verstärkung k_{M} anzunehmen:

$$T_{\mathrm{M}}\dot{\eta} + \eta = k_{\mathrm{M}} u_{\mathrm{PWM}}. \tag{14}$$

Die Parameter T_{M} und k_{M} in dieser Gleichung sind experimentell zu bestimmen. Bei einem PWM-Wert von 0 stellt sich die Differenzdrehzahl 0 ein, so dass die Absolutdrehzahl $\bar{\eta}$ dann der Grunddrehzahl η_0 entspricht.

2.3. Gesamtmodell

Das Gesamtmodell des Systems ergibt sich aus den Gln. (11) und (14) unter Berücksichtigung von (13) zu

$$\dot{\eta} = -\frac{1}{T_{\mathrm{M}}}\eta + \frac{k_{\mathrm{M}}}{T_{\mathrm{M}}} u_{\mathrm{PWM}} \tag{15a}$$

$$\ddot{h} = \frac{k_{\mathrm{L}}}{m}\left(\frac{k_{\mathrm{V}}(\eta + \eta_0) - A_{\mathrm{B}}\dot{h}}{A_{\mathrm{SP}}}\right)^2 - g. \tag{15b}$$

Bekannte Parameter: A_B, A_{SP}, m, g

Unbekannte und damit zu bestimmende Parameter: T_{M}, k_{M}, k_{V}, k_{L}, η_0.

3. Parameteridentifikation

Wie im vorangegangen Abschnitt deutlich geworden ist, müssen eine Reihe von Parametern experimentell bestimmt werden, sofern man modellbasierte Steuerungs- und Regelungsstrategien anwenden möchte. Dies kann zum einen durch Ausnutzung der Gleichungsstrukturen

in (15) verbunden mit „geschicktem Experimentieren“ und zum anderen durch die Anwendung der Methode der kleinsten Fehlerquadrate bzw. durch die Minimierung eines aus Mess- und Simulationsdaten gebildeten Kostenfunktionals geschehen. Das Vorgehen wird in den nachfolgenden Abschnitten erläutert.

3.1. Bestimmung des Parameters k_L

Der Parameter k_L beschreibt das Verhältnis zwischen Luftspaltgeschwindigkeit v_SP und Strömungswiderstandskraft F_w (vergl. Gl. (9)). Eine Inspektion von Gl. (15b) zeigt, dass dieser sehr einfach bestimmt werden kann, wenn man die Kugel bei ausgeschaltetem Lüfter ($\bar{\eta} = 0$) einfach in der Röhre herunterfallen lässt. Bekanntermaßen stellt sich dann schnell eine konstante Geschwindigkeit ein ($\ddot{h} = 0$), sodass sich der Parameter unmittelbar aus den Messdaten berechnen lässt, wenn man $\dot{h}$ im Bereich konstanter Fallgeschwindigkeit bestimmt und Gl. (15a) entsprechend umgestellt hat. Dann gilt:

$$k_\mathsf{L} = mg \left(\frac{A_\mathsf{SP}}{A_\mathsf{B} \dot{h}_\mathsf{exp}} \right)^2 \tag{16}$$

mit der experimentell bestimmten Fallgeschwindigkeit $\dot{h}_\mathsf{exp}$.

3.2. Bestimmung des Parameters k_V

Für die Bestimmung des Parameters k_V, der das Verhältnis von Lüfterumdrehungszahl und erzeugten Volumenstrom darstellt, kann ebenfalls Gl. (15b) zusammen mit dem in Abschnitt 3.1 bestimmten Parameter k_L herangezogen werden. Dazu gestaltet man das Experiment so, dass sich der Ball bei konstanter Absolut-Lüfterdrehzahl $\bar{\eta}$ mit konstanter Geschwindigkeit $\dot{h}$ (d.h. $\ddot{h} = 0$) nach oben bewegt. Dann ergibt sich der Parameter k_V wie folgt:

$$k_\mathsf{V} = \frac{\sqrt{\frac{mg}{k_\mathsf{L}}} A_\mathsf{SP} + A_\mathsf{B} \dot{h}_\mathsf{exp}}{\bar{\eta}_\mathsf{exp}} \tag{17}$$

mit den experimentell bestimmten konstanten Werten $\dot{h}_\mathsf{exp}$ und $\bar{\eta}_\mathsf{exp}$ für die Ballgeschwindigkeit bzw. die absolute Lüfterdrehzahl.

3.3. Bestimmung der Parameter k_M und T_M des Motors

Da das Verhalten des Motors als PT_1-Glied modelliert wurde, lassen sich die Parameter prinzipiell aus der Sprungantwort der Differenzdrehzahl η grafisch bestimmen. Hier sollen

zwei alternative Ansätze benutzt und miteinander verglichen werden, da die Parameter als Grundlage für die nichtlineare Steuerung sehr genau ermittelt werden müssen.

3.3.1. Bestimmung mit Hilfe der Methode der kleinsten Fehlerquadrate (MKQ)

Als Grundlage der Parameterbestimmung mit der Methode der kleinsten Fehlerquadrate dient ein Experiment, bei dem die Verläufe des Lüfter-PWM-Wertes u_{PWM}, der Differenzdrehzahl η sowie deren Ableitung $\dot{\eta}$ aufgezeichnet werden. Das weitere Vorgehen basiert darauf, dass Gl. (14) in jedem Abtastschritt k erfüllt sein muss:

$$\dot{\eta}[k] = -\frac{1}{T_{\mathrm{M}}}\eta[k] + \frac{k_{\mathrm{M}}}{T}u_{\mathrm{PWM}}[k]. \tag{18}$$

Hat man nun n Messwerte aller Größen vorliegen und fasst diese im Vektor $\mathbf{y}$ und der Matrix Φ wie folgt zusammen:

$$\mathbf{y} = \begin{pmatrix} \dot{\eta}[0] \\ \dot{\eta}[1] \\ \vdots \\ \dot{\eta}[n-1] \end{pmatrix} \qquad \Phi = \begin{pmatrix} \eta[0] & u_{\mathrm{PWM}}[0] \\ \eta[1] & u_{\mathrm{PWM}}[1] \\ \vdots & \vdots \\ \eta[n-1] & u_{\mathrm{PWM}}[n-1] \end{pmatrix},$$

dann muss entsprechend Gleichung (18) gelten:

$$\begin{pmatrix} \dot{\eta}[0] \\ \dot{\eta}[1] \\ \vdots \\ \dot{\eta}[n-1] \end{pmatrix} = \begin{pmatrix} \eta[0] & u_{\mathrm{PWM}}[0] \\ \eta[1] & u_{\mathrm{PWM}}[1] \\ \vdots & \vdots \\ \eta[n-1] & u_{\mathrm{PWM}}[n-1] \end{pmatrix} \cdot \underbrace{\begin{pmatrix} -\frac{1}{T_{\mathrm{M}}} \\ \frac{k_{\mathrm{M}}}{T_{\mathrm{M}}} \end{pmatrix}}_{\mathbf{a}}, \tag{19}$$

in Kurzform also $\mathbf{y} = \Phi\mathbf{a}$. Unter der Annahme von „vielen“ Messwerten (d.h $n > 2$) handelt es sich bei (19) um ein überbestimmtes lineares Gleichungssystem. Die Koeffizientenmatrix Φ hat mehr Zeilen als Spalten und lässt sich daher nicht invertieren. Im Allgemeinen existiert in diesem Fall keine *exakte* Lösung, da Messungenauigkeiten zu Widersprüchen in den Gleichungen führen. Gesucht ist demnach der Parametervektor $\mathbf{a}$, sodass der summierte (quadratische) Fehler (linke Seite minus rechte Seite) minimal wird. Dazu multipliziert man (19) von links mit der transponierten Matrix Φ^T und erhält:

$$\Phi^T\mathbf{y} = \Phi^T\Phi\mathbf{a}.$$

Da $\Phi^T\Phi$ quadratisch ist, lässt sie sich im Falle ihrer Regularität invertieren und man erhält:

$$(\Phi^T\Phi)^{-1}\Phi^T\mathbf{y} = \mathbf{a}.$$

Die Matrix $(\Phi^T\Phi)^{-1}$ wird auch als *Pseudoinverse* der Matrix Φ bezeichnet. Damit erhält man den gesuchten Parametervektor $\mathbf{a} = (a_0, a_1)^T$ und es gilt:

$$T_\mathsf{M} = -\frac{1}{a_0} \tag{20a}$$

$$k_\mathsf{M} = -\frac{a_1}{a_0}. \tag{20b}$$

Der Nachteil der Methode ist, dass es erforderlich ist, die Ableitungen der Drehzahl numerisch zu bestimmen. In diesem Praktikumsversuch liegt die Ableitung in den Versuchsdaten bereits vor.

3.3.2. Bestimmung über die Minimierung eines Kostenfunktionals

Alternativ zur MKQ-Methode lassen sich die Lüfterparameter auch mit Hilfe eines Algorithmus bestimmen, der ein bestimmtes Kostenfunktional minimiert. Die Bestimmung erfolgt dabei nicht durch explizites Lösen einer Gleichung wie beim MKQ-Verfahren, sondern durch eine numerische Iteration, wobei die Schätzwerte $\hat{T}_\mathsf{M}$ und $\hat{k}_\mathsf{M}$ für die Parameter des Lüftermodells (14) solange variiert werden, bis das Kostenfunktional ein Minimum erreicht. Ein weitverbreitetes Verfahren ist hierfür das Downhill-Simplex-Verfahren nach Nelder & Mead [3, 5]. Hierzu definiert man ein Kostenfunktional J, das die Differenz zwischen dem gemessenen Verlauf $\eta[k]$, $k = 0, \ldots, n-1$ der Lüfterdifferenzdrehzahl und dem mit den Schätzparametern $\hat{T}_\mathsf{M}$ und $\hat{k}_\mathsf{M}$ simulierten Verlauf $\eta_\mathsf{sim}[k]$ bewertet. Man setzt an:

$$J = \sum_{k=0}^{n-1} \big(\eta[k] - \eta_\mathsf{sim}[k]\big)^2. \tag{21}$$

Dem Algorithmus ist also eine Funktion zu übergeben, welche in Abhängigkeit der Lüfterparameter T_M und k_M sowie des gemessenen Stellgrößenverlaufes $u[k], k = 0, \ldots, n-1$

1. den zugehörigen Verlauf der Lüfterdifferenzdrehzahl simuliert und
2. aus diesem simulierten Verlauf sowie den *Messdaten* der Lüfterdifferenzdrehzahl das Kostenfunktional (21) berechnet.

Der Vorteil des Verfahrens besteht darin, dass keine Ableitung der Drehzahl benötigt wird. Nachteilig sind die längere Rechenzeit (insbesondere wenn – wie hier – kein expliziter Gradient des Kostenfunktionals angegeben werden kann) und die Gefahr, in einem lokalen Minimum hängen zu bleiben.

4. Nichtlineare Steuerung

Im Rahmen des Versuches wird sich zeigen, dass eine Regelung alleine gut geeignet für die Stabilisierung eines Arbeitspunktes ist, deren Leistungsfähigkeit für eine präzise Folgeregelung (z.B. bei Arbeitspunktwechseln) aber nicht ausreicht, insbesondere dann, wenn schnelle Übergänge über einen großen Betriebsbereich gefordert werden. In einem solchen Fall kann der Einsatz einer zusätzlichen Steuerung hilfreich sein. Dabei wird aus der Kenntnis der Solltrajektorie $t \mapsto h_{\text{ref}}(t)$ für die Ballposition und des Modells (15) der erforderliche Verlauf $t \mapsto u_{\text{PWM,ref}}(t)$ für den Lüfter-PWM-Wert a priori berechnet. Dieser wird dann zusammen mit dem Reglerausgang auf die Eingang der Strecke gegeben, vergl. Abbildung 3. Die Steuerung sorgt dann dafür, dass der Regler nicht mehr alleine für den Übergangsvorgang, sondern nur noch für die Ausregelung der Abweichungen verantwortlich ist (sog. *Folgeregler*). Diese Abweichungen ergeben sich aus Störungen, die auf das System einwirken, und Ungenauigkeiten im für den Steuerungsentwurf verwendeten Modell.

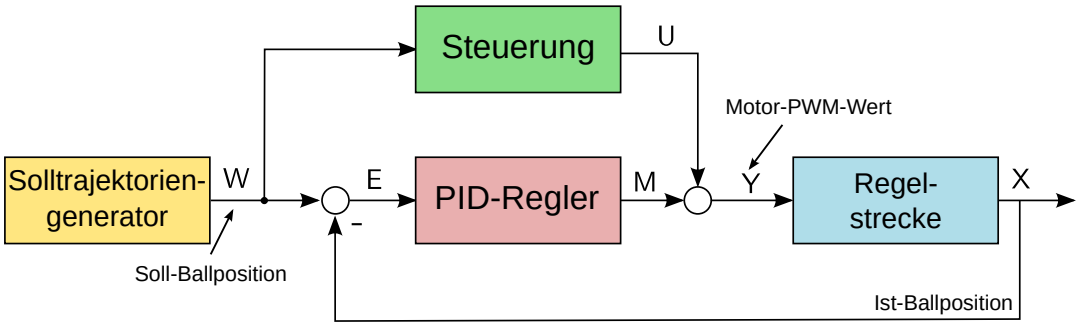

Abbildung 3: Signalflussplan für einen geschlossenen Regelkreis mit zusätzlicher Steuerung.

4.1. Steuerungsentwurf

Für die vorliegende Aufgabe wird eine *nichtlineare* Steuerung entworfen, damit auch besonders große Arbeitspunktwechsel mit genügender Genauigkeit bewältigt werden können, was bei einer auf dem linearisierten Modell (Vorbereitungsaufgabe 7.2) basierenden (linearen) Steuerung (vergl. Praktikumsversuch V1) nicht der Fall wäre. Hierzu geht man von Gl. (15b) aus und löst diese nach η auf:

$$\eta = \frac{1}{k_{\text{V}}} \left(A_{\text{SP}} \sqrt{\frac{m}{k_{\text{L}}}(\ddot{h} + g)} + A_{\text{B}} \dot{h} \right) - \eta_0 =: f(\ddot{h}, \dot{h}). \tag{22}$$

Für den PWM-Wert der Lüftermotorspannung erhält man aus Gl. (15a):

$$u_{\text{PWM}} = \frac{T_{\text{M}}}{k_{\text{M}}} \dot{\eta} + \frac{1}{k_{\text{M}}} \eta. \tag{23}$$

Setzt man in diese Gleichung den Ausdruck für η nach Gl. (22) bzw. dessen Zeitableitung ein, so erhält man ein Stellgesetz für den PWM-Wert der Lüftermotorspannung, das nur von $\dot{h}$, $\ddot{h}$ und $h^{(3)}$ abhängt. Die Zeitableitung von (22) ergibt sich zu

$$\begin{aligned}\dot{\eta} &= \frac{\partial f}{\partial \ddot{h}} h^{(3)} + \frac{\partial f}{\partial \dot{h}} \ddot{h} \\ &= \frac{m A_{\mathsf{SP}}}{2 k_{\mathsf{V}} k_{\mathsf{L}}} \left(\frac{m}{k_{\mathsf{L}}} (\ddot{h} + g) \right)^{-\frac{1}{2}} h^{(3)} + \frac{A_{\mathsf{B}}}{k_{\mathsf{V}}} \ddot{h}, \end{aligned} \tag{24}$$

so dass man als Ausdruck für die Steuerung erhält:

$$\begin{aligned} u_{\mathsf{PWM}} = \frac{T_{\mathsf{M}}}{k_{\mathsf{M}}} \left[\frac{m A_{\mathsf{SP}}}{2 k_{\mathsf{V}} k_{\mathsf{L}}} \left(\frac{m}{k_{\mathsf{L}}} (\ddot{h} + g) \right)^{-\frac{1}{2}} h^{(3)} + \frac{A_{\mathsf{B}}}{k_{\mathsf{V}}} \ddot{h} \right] + \\ \frac{1}{k_{\mathsf{M}}} \left[\left(\frac{A_{\mathsf{SP}}}{k_{\mathsf{V}}} \sqrt{\frac{m}{k_{\mathsf{L}}} (\ddot{h} + g)} + \frac{A_{\mathsf{B}}}{k_{\mathsf{V}}} \dot{h} \right) - \eta_0 \right]. \end{aligned} \tag{25}$$

Setzt man für $h^{(3)}$, $\ddot{h}$ und $\dot{h}$ die sich aus dem Wunschverlauf $t \to h_{\mathsf{ref}}(t)$ ergebenden Verläufe ein, so liefert Gl. (25) den – laut Modell – erforderlichen Verlauf $t \mapsto u_{\mathsf{PWM,ref}}(t)$ der Stellgröße u_{PWM}.

Nebenbemerkung: Wie ersichtlich, lassen sich aus dem Verlauf $h(t)$ der Ballposition und dessen Zeitableitungen die Verläufe der übrigen zeitveränderlichen Größen – nämlich $\eta(t)$ (Gl. (22)), $\dot{\eta}(t)$ (Gl. (24)) und $u(t)$ (Gl. (25)) berechnen, ohne dabei Integrale lösen zu müssen. (Nichtlineare) SISO-Systeme, die diese Eigenschaft aufweisen, werden als *flach* bezeichnet und die Größe, von der alle anderen Größen integrallos abhängen, als *flacher Ausgang*[3], hier also die Ballposition h. Die Theorie der flachen Systeme liefert mächtige Hilfsmittel zum nichtlinearen Steuerungs- und Regelungsentwurf, die mittlerweile auch in der industriellen Praxis ein breites Anwendungsfeld gefunden haben. Näheres dazu wird in der Vorlesung *Flachheitsbasierte Folgeregelung* im neunten Fachsemester vermittelt.

4.2. Wahl der Solltrajektorie

Die Gl. (25) berechnet aus der Solltrajektorie $t \mapsto h_{\mathsf{ref}}(t)$ für die Ballposition die erforderliche Steuertrajektorie $t \mapsto u_{\mathsf{PWM,ref}}(t)$. Damit diese existiert, muss die Funktion, die die Solltrajektorie beschreibt, dreimal differenzierbar sein. Damit die Steuertrajektorie darüberhinaus nicht springt, ist es erforderlich, dass auch die dritte Ableitung stetig ist. Es ist also zu fordern, dass h_{ref} dreimal stetig differenzierbar ist. Die Solltrajektorie ist entsprechend zu wählen. Im

[3] Im Falle von MIMO-Systemen besteht der flache Ausgang aus mehreren Komponenten. Dabei muss zusätzlich gefordert werden, dass diese Komponenten unabhängig voneinander vorgebbar sind.

vorliegenden Praktikumsversuch kommen zwei verschiedene Solltrajektorien zum Einsatz: Eine für einen Arbeitspunktwechsel, also den Übergang zwischen zwei Ruhelagen, und eine für die Dauerschwingung des Balls um eine bestimmte Höhe.

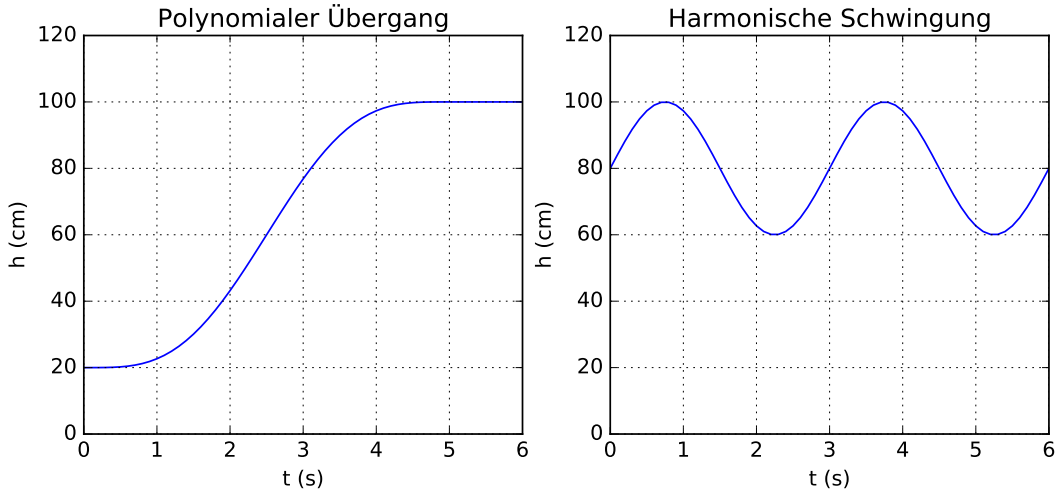

Abbildung 4: Beispiele für Solltrajektorien: Übergang zwischen zwei Ruhelagen auf Basis eines Polynoms (links), harmonische Schwingung (rechts).

Für den Wechsel von einem Arbeitspunkt h_0 zu einem anderen Arbeitspunkt h_1 innerhalb der Zeit τ wird h_{ref} zur Berechnung der Steuerung nach Gl. (25) wie folgt angesetzt:

$$h_{\text{ref}}(t) = \begin{cases} h_0 & \text{für} \quad t < 0 \\ \sum_{i=0}^{7} c_i t^i & \text{für} \quad 0 \leq t \leq \tau \\ h_1 & \text{für} \quad t > \tau \end{cases} . \tag{26}$$

Die acht Koeffizienten $c_0, \ldots, c_7$ ergeben sich aus der Forderung, dass h_{ref} auch an den Stellen $t = 0$ und $t = \tau$ dreimal stetig differenzierbar ist, dort also gelten muss: $h_{\text{ref}}(0) = h_0, \dot{h}_{\text{ref}}(0) = 0, \ddot{h}_{\text{ref}}(0) = 0, h^{(3)}_{\text{ref}}(0) = 0$ sowie $h_{\text{ref}}(\tau) = h_1, \dot{h}_{\text{ref}}(\tau) = 0, \ddot{h}_{\text{ref}}(\tau) = 0, h^{(3)}_{\text{ref}}(\tau) = 0$, also 8 Bedingungen erfüllt sein müssen.

Für eine Schwingung der Ballposition um die Höhe h_0 mit einer Amplitude A und der Periodendauer τ ist folgender Ansatz sinnvoll:

$$h_{\text{ref}}(t) = h_0 + A \sin\left(\frac{2\pi}{\tau} t\right) . \tag{27}$$

Diese Trajektorie ist unendlich oft stetig differenzierbar.

Die Berechnung der Parameter für die Solltrajektorie erfolgt in diesem Praktikumsversuch automatisch entsprechend der von Ihnen gemachten Vorgaben.

5. Parameteridentifikation mittels Python

In diesem Abschnitt wird beschrieben, wie Sie mit Hilfe der Programmiersprache Python die Parameter des PT_1-Gliedes, welches zur Modellierung der Motordynamik angesetzt wird, unter Verwendung der in Abschnitt 3.3 beschriebenen Verfahren identifizieren. Dabei geht es nicht darum, Python umfassend zu erlernen, sondern einige wenige Hilfsmittel gezielt einzusetzen, um schnell und ingenieurmäßig zum Ziel zu kommen.

5.1. Bereitstellung und Verwendung von Python

Um Python bei der Versuchsvorbereitung verwenden zu können, gibt es drei verschiedene Möglichkeiten:

1. Die einfachste Möglichkeit besteht darin, Python interaktiv über das ILMRT-Portal[4] des Instituts für Regelungs- und Steuerungstheorie mit Jupyter Notebook zu nutzen. Die hierzu erforderlichen Zugangsdaten finden Sie Ihrer Gruppe zugeordnet auf der OPAL Lernplattform. Um diese Möglichkeit nutzen zu können, muss Ihr Rechner mit dem TU-Netz verbunden sein. Nutzen Sie hierzu ggf. VPN[5].
2. Sie installieren sich Python (Version 3.5) lokal auf Ihrem Rechner und nutzen Jupyter Notebook.
3. Sie installieren sich Python (Version 3.5) lokal auf Ihrem Rechner und entwerfen eine Python-Datei, die sie im Python-Interpreter aufrufen.

Wenn Sie sich Python lokal installieren möchten, so finden Sie Hinweise dazu auf unserer Website im Bereich „Studium“ unter „Python in der Lehre“ (https://tu-dresden.de/ing/elektrotechnik/rst/studium/python-in-der-lehre).

5.2. Aufbau der Python-Quellcode-Datei

Der Quellcode zur Umsetzung der Parameteridentifikation kann entweder direkt in einer Datei erstellt und dann im Rahmen einer existierenden Python-Installation im Python-Interpreter aufgerufen oder aber interaktiv in Jupyter Notebook entwickelt werden (s.o.). In jedem Fall besteht der Quellcode aus den folgenden Abschnitten:

1. Import benötigter Python-Module.

[4] ILMRT – Interaktive Lehrmaterialien in der Regelungstechnik

[5] Zugang erfolgt mit Ihrem ZIH-Benutzernamen als benutzername@tu-dresden.de, weitere Informationen unter https://tu-dresden.de/zih/dienste/service-katalog/arbeitsumgebung/zugang_datennetz/vpn

2. Einlesen der aufgenommenen Messdaten aus der im Experiment erzeugten csv-Datei.
3. Festlegung des Datenbereiches, auf dem die Identifikation durchgeführt werden soll,
4. Auswahl der benötigten Messgrößen, Umrechnung dieser in SI-Einheiten und Definition benötigter Hilfsgrößen (z.B. der Grunddrehzahl η_0)
5. Definition einer Funktion, die die rechte Seite der Dgl. (14) realisiert. Diese Funktion wird benötigt, um den Verlauf der Lüfterdrehzahl in Abhängigkeit von den gefundenen Parametern und des Stellsignalverlaufs mit Hilfe eines numerischen Integrators berechnen zu können.
6. Definition einer Funktion, die das Kostenfunktional (21) realisiert.
7. Konstruktion der Matrizen und Vektoren zur Parameterbestimmung mit Hilfe der Methode der kleinsten Fehlerquadrate nach Abschnitt 3.3.1, Lösung des Gleichungssystems (19) und Simulation des PT_1-Gliedes mit den identifizierten Parametern.
8. Aufruf einer Minimierungsroutine, die unter Verwendung des definierten Kostenfunktionals (21) die Streckenparameter berechnet. Simulation eines PT_1-Gliedes mit den identifizierten Parametern.
9. Darstellung der Ergebnisse in Form eines grafischen Vergleichs der aufgenommenen Messdaten für die Lüfterdrehzahl mit den beiden simulierten Verläufen.

Für das Verständnis der nachfolgenden Ausführungen und die eigene Entwicklungsarbeit sind zwei Datentypen in Python besonders wichtig:

Liste
: Dieser Datentyp wird nativ von Python bereitgestellt. Eine Liste ist eine Sammlung von Objekten *gleichen oder unterschiedlichen* Typs. Eine Liste von drei Objekten `a`, `b` und `c` (z.B. Zahlen) wird mit eckigen Klammern angelegt: `x = [a, b, c]`. Der Zugriff auf die einzelnen Elemente der Liste kann über eine nullbasierte Indizierung mittels eckiger Klammern erfolgen. Das erste Element einer Liste `x` erhält man dementsprechend mit `x[0]`. Auf das letzte Element der Liste kann mit `x[-1]` zugegriffen werden. Die Syntax `x[i:j]` führt dazu, dass eine Liste mit den Elementen `i` bis `j-1` der Liste `x` zurückgeliefert wird. Möchte man auf alle Elemente ab einschließlich dem `i`.ten Element der Liste `x` zugreifen, so lautet die Syntax: `x[i:]`. Die Liste kann auch wieder in ihre einzelnen Bestandteile zerlegt („entpackt“) werden: Für das obige Beispiel könnte man schreiben `d, e, f = x` (hier würde `d` den Inhalt von `a` enthalten, `e` den von `b` usw.).

Array
: Dieser Datentyp wird vom Modul `NumPy` (siehe Quellcodeabschnitt 1 unten) bereitgestellt, das am Beginn der Pythondatei über das Kommando `import numpy as np` importiert wird. Alle von `NumPy` bereitgestellten Methoden werden dann nachfolgend mit dem Präfix `np.` aufgerufen. Bei einem Array handelt es sich um eine Liste, deren

Elemente alle vom *gleichen numerischen* Typ sind, und mit der bestimmte Rechenoperationen durchgeführt werden können, bspw. die *elementweise* Multiplikation zweier Arrays `A` und `B` mittels `A*B`. Ein *eindimensionales* Array kann wie folgt angelegt werden: `x = np.array([1, 2, 3])`. Dabei ist nicht festgelegt, ob es sich um einen Zeilen- oder einen Spaltenvektor handelt. Ein zweidimensionales Array kann auf unterschiedliche Weise angelegt werden: `x=np.array([[1, 2, 3],[4, 5, 6]])` erzeugt das 2×3-Array

$$x = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \tag{28a}$$

`y=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])` hingegen das 3×3-Array

$$y = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}. \tag{28b}$$

Zwei eindimensionale Arrays können zu einem mehrdimensionalen Array verknüpft werden: Der Befehl `np.column_stack(z)` stellt die in der Liste `z` stehenden eindimensionalen Arrays spaltenweise nebeneinander (behandelt die Arrays also als Spaltenvektoren), wohingegen der Befehl `np.row_stack(z)` die in der Liste `z` stehenden Arrays zeilenweise untereinander stellt (behandelt die Arrays also als Zeilenvektoren).

Zweidimensionale Arrays können entsprechend den Regeln der Matrixmultiplikation mit Hilfe des Operators `@` multipliziert werden, sofern die Dimensionen stimmen. Im obigen Beispiel (28) ergibt sich das Produkt xy der Matrizen x und y über den Befehl `x@y`, wohingegen `y@x` einen Fehler ausgibt. Die Transponierte A^T der Matrix A erhält man über den Operator `T`, zu schreiben ist: `A.T`. Die Inverse einer quadratischen Matrix A ergibt sich mit dem Befehl `np.linalg.inv(A)`.

Der indizierte Zugriff auf Arrays erfolgt wie bei der Liste, z.B. liefert `x[:,2]` alle Elemente der dritten Spalte des Arrays `x` als eindimensionales Array zurück.

Nachfolgend ist der Inhalt einer Vorlage gegeben, die als Ausgangspunkt für die eigenen Entwicklungen im Rahmen der Versuchsvorbereitung verwendet werden kann. Anhand dieser Vorlage werden die einzelnen Schritte zur Parameteridentifikation erläutert. Die Datei steht unter dem Namen `v21_paraident_template.py` auch auf der OPAL Lernplattform zum Download für Python 3.5 bereit bzw. findet sich im ILMRT-Portal des Instituts.

```
1  # -*- coding: utf-8 -*-
2  ########################## Abschnitt 1 ###################################
3  import numpy as np
4  import scipy.integrate as sci
5  import scipy.optimize as sco
6  import matplotlib.pyplot as plt
7
8  # Wenn True werden nur u und eta über den Tastschritten angezeigt
9  plot_raw_data = True
10
```

```
11
12  ############################ Abschnitt 2 ###################################
13
14  # Daten laden, erste Zeile auslassen, da dort Spaltennamen stehen
15  Data = np.loadtxt(XXX, skiprows=1)
16
17
18  ############################ Abschnitt 3 ###################################
19
20  # Wenn plot_raw_data == True werden u und etaAbs über den Abtastschritten
21  # geplottet. Das Skript wird anschließend beendet
22  if plot_raw_data:
23
24      u = Data[:,6]
25      etaAbs = Data[:,7]
26
27      plt.figure()
28      plt.plot(u, hold=True)
29      plt.plot(etaAbs)
30      plt.legend(['u (PWM)', 'eta'])
31      plt.xlabel('Index')
32      plt.ylabel('u bzw. eta in U/min')
33      plt.show()
34
35      quit()
36
37
38  ############################ Abschnitt 4 ###################################
39
40  # Daten zurechtschneiden
41  iStart = XXX
42  iStop =  XXX
43  Data = Data[iStart:iStop,:]
44
45  # Alle Größen in SI-Einheiten laden [t] = s, [eta] = s^{-1}!
46  t = XXX
47  u = XXX             # PWM-Wert der Motorspannung
48  etaAbs = XXX        # Lüfterdrehzahl in s^{-1}
49  etaDot = XXX        # Zeitableitung der Lüfterdrehzahl in s^{-2}
50
51  # Hilfsgrößen
52  etaAbs0 = XXX       # Lüfterdrehungszahl für u = 0
53  eta = XXX           # Differenz-Lüfterdrehzahl bezogen auf etaAbs0
54
55
56  ############################ Abschnitt 5 ###################################
57
58  # Systemdynamik (rechte Seite Dgl. PT1-Glied)
59  def pt1_sys(x, t, T_para, K_para, t_data, u_data):
60
61      # u an der Stelle t im durch t_data und u_data definierten Verlauf
62      u = u_data[max(0, len(t_data[t_data<=t])-1)]
63
64      xDot = XXX  # hier rechte Seite Dgl. implementieren
65      return xDot
66
67
68  ############################ Abschnitt 6 ###################################
69
70  # Kostenfunktional für Simplex-Algorithmus
71  def cost_functional_pt1(Para, t_data, u_data, eta_data):
72      T, K = Para
73
```

```
74      # Simulation System mit Parametern T und K für den Eingangssignalverlauf
75      # t_data, u_data, [:,0] sorgt dafür, dass etaSim ein 1-dim. Array ist
76      etaSim = sci.odeint(XXX)[:,0]
77
78      # Berechnung Kostenfunktional
79      J = np.sum(XXX)
80      return J
81
82
83  ############################ Abschnitt 7 ###################################
84  # Parameteridentifikation (MKQ)
85
86  Phi = np.column_stack([XXX, XXX])
87  y = XXX
88  a = XXX      # Nutzen Sie np.linalg.inv, @, .T (siehe Anleitung)
89
90  # Parameter ausgeben
91  T_MKQ = XXX
92  km_MKQ = XXX
93  print('MKQ                  : T = %f ms, km = %f 1/s' % (T_MKQ*1000, km_MKQ))
94
95  # Simulation des Systems mit den identifizierten Parametern
96  etaSim_MKQ = sci.odeint(XXX)
97
98
99  ############################ Abschnitt 8 ###################################
100 # Parameteridentifikation (Simplex-Algorithmus)
101
102 TInit = 0.5
103 KInit = 5
104 xMin = sco.fmin(XXX)
105
106 # Parameter ausgeben
107 T_Simplex, km_Simplex = xMin
108 print('Simplex (Nelder-Mead): T = %f ms, km = %f 1/s' % (T_Simplex*1000, km_Simplex))
109
110 # Simulation des Systems mit den identifizierten Parametern
111 etaSim_Simplex = sci.odeint(XXX)
112
113
114 ############################ Abschnitt 9 ###################################
115 # Ergebnis darstellen
116
117 plt.figure()
118 plt.plot(t, etaAbs*60, t, (etaAbs0+etaSim_MKQ)*60, '-+', t, (etaAbs0+etaSim_Simplex)
        *60, '-x', markersize=8, markevery=5)
119 plt.ylabel('Lüfterdrehzahl eta (U/min)')
120 plt.xlabel('Zeit t (s)')
121 plt.title('Vergleich Messdaten und Identifikation')
122 plt.legend(['Experiment', 'Identifikation (MKQ)', 'Identifikation (Simplex)'], loc='
        lower right')
123 plt.grid()
124
125 plt.show()
```

Quellcodeabschnitt 1 – Import benötigter Python-Module

Damit benötigte Funktionen zur Verfügung stehen, müssen am Kopf der Datei bzw. am Anfang des Jupyter Notebooks einige Module eingebunden werden: Das Modul `numpy` stellt Funktionalität für das Arbeiten mit Arrays zur Verfügung (s.o.). Für die durchzuführenden

Untersuchungen ist es bspw. sinnvoll, die aufgenommenen Messdaten in Arrays zu halten. Alle Befehle des `numpy`-Moduls werden in der Datei mit dem Präfix `np.` aufgerufen. Das Modul `scipy.optimize` stellt eine Routine mit dem Namen `fmin` zur Verfügung, die für eine gegebene Funktion $x \mapsto f(x)$ das Argument x so bestimmt, dass $f(x)$ minimal ist. Auf Details wird weiter unten eingegangen. Das Modul `scipy.integrate` stellt eine Funktion `odeint` bereit, mit deren Hilfe die Lösung $t \mapsto x(t)$ der Dgl. $\dot{x} = f(x,t)$ für einen Anfangswert $x(0) = x_0$ numerisch bestimmt werden kann. Der Aufruf erfolgt hier über die Präfixe `sci` bzw. `sco`. Schließlich kann die grafische Auswertung mit Hilfe des Moduls `pyplot` aus der `matplotlib`-Bibliothek mit dem Präfix `plt` erfolgen.

Sie können in diesem Abschnitt die Variable `plot_raw_data` auf `True` setzen, damit das Skript nur die Verläufe der Lüfterdrehzahl und des PWM-Wertes darstellt und sonst nichts macht. Dies hilft Ihnen ggf. zu bestimmen, welchen Bereich der Daten Sie überhaupt zur Parameteridentifikation verwenden möchten.

Quellcodeabschnitt 2 – Laden der Datei mit den Messdaten

In diesem Abschnitt werden die Versuchsdaten aus der csv-Datei, die die Messdaten enthält, geladen. Hierzu stellt das NumPy-Modul die Funktion `loadtext(file_name, skiprows=n)` zur Verfügung, welche die Daten aus der Datei mit dem Namen `file_name` einliest. Der Parameter `n` im Keyword-Argument `skiprows` ist durch die Anzahl der Zeilen zu ersetzen, die nicht eingelesen werden sollen (z.B. Berücksichtigung der Kopfzeile).

Quellcodeabschnitt 3 – Darstellung der Rohdaten

Hier brauchen Sie nichts zu verändern. In diesem Abschnitt werden lediglich die Verläufe der Lüftergeschwindigkeit und des PWM-Wertes dargestellt zur Bestimmung der Variablen `iStart` und `iStop` im nächsten Abschnitt.

Quellcodeabschnitt 4 – Aufbereitung der Messdaten

Legen Sie hier über die Variablen `iStart` und `iStop` fest, welchen Bereich der Daten Sie für die Identifikation verwenden wollen (siehe Hinweise in Quellcodeabschnitt 1). Anschließend sind die erforderlichen Messdaten für die Zeit, den PWM-Wert, die Absolutlüfterdrehzahl und deren Ableitung entsprechend der Spezifikationen im Anhang B in *SI-Einheiten* einzulesen (ggf. umrechnen!). Außerdem werden die Grunddrehzahl η_0 für $u_{\text{PWM}} = 0$ und die Differenzdrehzahl $\eta = \bar{\eta} - \eta_0$ benötigt.

Quellcodeabschnitt 5 – Implementation der rechten Seite der Differentialgleichung (15a)

Die in diesem Abschnitt definierte Funktion wird später bei der numerischen Integration der Dgl. (15a) benötigt. Sie implementiert die rechte Seite der Dgl. und ist wie folgt zu deklarieren:

```
1 def pt1_sys(x, t, T_para, K_para, t_data, u_data):
```

Alle Zeilen des Funktionscodes müssen in Python mit Tabulator[6] eingerückt sein, damit der Interpreter diese als zur Funktionsdefinition zugehörig einordnen kann. Die Parameter bedeuten im einzelnen:

1. `x`: der aktuelle Wert von x beim Funktionsaufruf
2. `t`: der aktuelle Zeitpunkt des Funktionsaufrufs
3. `T_para`, `K_para`: die Parameter T_M und k_M der Dgl. (15a)
4. `t_data`: ein eindimensionales Array, das die Zeitwerte der aufgenommenen Messung enthält
5. `u_data`: ein eindimensionales Array, das die im Versuch zu den in `t_data` gespeicherten Zeitpunkten aufgenommen PWM-Stellwerte enthält.

Im Code dieser Funktion sind folgende Operationen durchzuführen (die Zeilennummern beziehen sich auf das ab Seite 15 beginnende Listing):

1. Zeile 62: Hier wird der zum Zeitpunkt `t` am Motor anliegende PWM-Wert berechnet. Da nicht sichergestellt ist, dass der Integrator die Funktion zu genau den Zeitpunkten aufruft, an denen gemessen wurde (also zu den Zeitpunkten, die in `t_data` stehen), ist es erforderlich, den Wert für u zum Zeitpunkt `t` durch ein Halteglied 0-ter Ordnung zu berechnen. Diese Funktion ist bereits für Sie implementiert.
2. Zeile 64: Mit den zur Verfügung stehenden Parametern `T_Para` und `K_Para` sowie dem aktuellen Stellwert `u` sowie dem aktuellen Zustand `x` wird die rechte Seite der Dgl. (15a) implementiert.
3. Zeile 65: Der aktuelle Wert der rechten Seite der Dgl. wird an den die Funktion aufrufenden Integrator zurückgegeben.

Quellcodeabschnitt 6 – Implementation des Kostenfunktionals

In Quellcodeabschnitt 6 wird eine Funktion definiert, die das Kostenfunktional (21) in Abhängigkeit von den Modellparametern T_M und k_M berechnet. Diese Funktion wird später von der übergeordneten Minimierungsroutine wiederholt mit variierenden Parametern aufgerufen, bis ein Minimum gefunden ist (siehe Quellcodeabschnitt 8). Die Funktion ist wie folgt zu deklarieren:

```
1 def cost_functional_pt1(Para, t_data, u_data, eta_data):
```

[6] Genauer: mit vier Leerzeichen, geschieht in den meisten Entwicklungsumgebungen automatisch beim Drücken der Tabulator-Taste.

Die Parameter bedeuten im einzelnen:

1. `Para`: eine Liste mit zwei Komponenten: `Para[0]` enthält die Zeitkonstante T_M, `Para[1]` die Verstärkung k_M.
2. `t_data`: ein eindimensionales Array, das die Zeitwerte der aufgenommenen Messung enthält
3. `u_data`: ein eindimensionales Array, das die im Versuch zu den in `t_data` gespeicherten Zeitpunkten aufgenommen PWM-Werte enthält.
4. `eta_data`: ein eindimensionales Array, das die im Versuch zu den in `t_data` gespeicherten Zeitpunkten aufgenommen Lüfterdifferenzdrehzahlwerte in s^{-1} enthält.

Im Code dieser Funktion sind folgende Operationen durchzuführen:

1. Zeile 72: Aus dem der Funktion übergebenen Array `Para` werden die beiden Streckenparameter T_M und k_M entpackt.
2. Zeile 76: Hier wird der Verlauf der Lüfterdifferenzdrehzahl basierend auf den Parameterwerten `T` und `K` simuliert. Dies geschieht durch numerische Integration mit Hilfe der Funktion `odeint` aus SciPy:

   `sci.odeint(func, x0, t, args=(a, b, c, d))`
   Integriert die in `func(x, t, a, b, c, d)` implementierte Funktion mit den festen Parameterwerten `a, b, c, d` ausgehend vom Anfangswert `x0` und liefert die Lösung zu den im Array `t` gespeicherten Zeitpunkten als ein Array zurück, das soviele Zeilen hat wie Elemente in `t` enthalten sind und soviele Spalten hat, wie Elemente im Anfangszustand `x0` vorhanden sind.
3. Zeile 79: An dieser Stelle erfolgt der „Vergleich“ der soeben simulativ bestimmten Daten mit den messtechnisch bestimmten. Dies erfolgt mittels des Kostenfunktionals (21). Das Integral kann mit der Funktion `sum` aus dem Modul NumPy (Präfix hier: `np`) berechnet werden:

   `np.sum(y)`
   Berechnet die Summe der in dem Array `y` gespeicherten Werte.

   Beachten Sie, dass die `n`-te Potenz x^n einer Variablen `x` wie folgt berechnet werden kann: `x**n`.
4. Zeile 80: Der Wert des Kostenfunktionals wird zurückgegeben (hier an die aufrufende Minimierungsroutine).

Quellcodeabschnitt 7 – Parameterbestimmung nach der MKQ

In diesem Abschnitt werden die Parameter des Lüftermodells mit der in Abschnitt 3.3.1 beschriebenen Methode der kleinsten Fehlerquadrate berechnet:

1. Zeile 86-88: Aufbau der Matrix Φ, des Vektors $\mathbf{y}$ und Berechnung des Lösungsvektors $\mathbf{a}$ entsprechend Gl. (19) und (20) (für Konstruktionshinweise sowie Rechenoperationen und Inversion/ Transponieren von Matrizen siehe Hinweis zu Arrays auf Seite 14).
2. Zeile 91+92: Berechnung der Parameter aus den Komponenten des Lösungsvektors `a`.
3. Zeile 96: Hier ist der Verlauf der Lüfterdifferenzdrehzahl für die ermittelten Lüfterparameter in Abhängigkeit vom messtechnisch erfassten Verlauf des PWM-Wertes in `u` zu den Zeitpunkten der Messung in `t` zu bestimmen, um ihn später mit den Messdaten grafisch vergleichen zu können. Siehe hierzu die Ausführungen zu `odeint` in Quellcodeabschnitt 6.

Quellcodeabschnitt 8 – Parameterbestimmung durch Minimierung eines Kostenfunktionals

In diesem Quellcodeabschnitt erfolgt die Bestimmung der Streckenparameter nach dem in Abschnitt 3.3.2 beschriebenen numerischen Optimierungsverfahren:

1. Zeile 102+103: Initialisierungswerte, die der Minimierungsroutine im nächsten Schritt zu übergeben sind.
2. Zeile 104: Aufruf der eigentlichen Optimierungsroutine. Diese wird aus dem Modul `scipy.optimize` bereitgestellt:

   `sco.fmin(func, IniVals, args=(a, b))`
   Berechnet unter Verwendung des so genannten Downhill-Simplex-Algorithmus [3, 5] für eine Funktion `func(x, a, b)` mit den festen Parametern `a` und `b` den Wert des variablen Arguments `x` so, dass die Funktion für dieses Argument ein Minimum zurückliefert. In der Liste `IniVals` sind die Anfangswerte für die Suche anzugeben. `IniVals` muss so viele Werte enthalten wie im Argument `x` erwartet werden. Im vorliegenden Fall wäre `IniVals = [TInit, KInit]`. Selbstverständlich können über `args` auch mehr als zwei feste Parameter übergeben werden, z.B. `(a, b, c, d, e)`.
3. Zeile 107+108: Ausgabe der Ergebnisse.
4. Zeile 111: Wie in Quellcodeabschnitt 7 soll auch hier eine Simulation des Lüfterverhaltens für die gemessenen Stellgrößenwerte mit den ermittelten Parametern erfolgen.

Quellcodeabschnitt 9 – Darstellung der Ergebnisse

Die in den Quellcodeabschnitten 7 und 8 simulierten Verläufe für die Lüfterdrehzahl werden in diesem Abschnitt zusammen mit den messtechnisch erfassten grafisch dargestellt, um die Ergebnisse beurteilen zu können. Sofern Sie im Template keine Anpassungen der Variablennamen vorgenommen haben, brauchen Sie hier nichts zu ändern.

6. Kontrollfragen

Wichtiger Hinweis: Beachten Sie bitte, dass Sie auch die Antworten auf die allgemeinen Kontrollfragen in der Praktikumseinführung [4] beherrschen! Beachten Sie weiterhin, dass im Eingangstest ggf. Regler mit Hilfe derjenigen Methoden zu dimensionieren sind, die in den Vorbereitungsaufgaben in Abschnitt 7 verwendet werden (eine Formelsammlung wird zur Verfügung gestellt, ein prüfungszugelassener Taschenrechner kann verwendet werden)!

6.1 Betrachten Sie das Gleichungssystem (15): Wie bzw. durch welche Experimente können die Parameter k_L, k_V, k_M und T_M bestimmt werden?

6.2 Für ein lineares zeitinvariantes Übertragungsglied liegen die dessen Dynamik beschreibende Differenzialgleichung und Messdaten für die Ausgangsgröße des Übertragungsglieds sowie deren Ableitungen vor. Geben Sie die Vektoren/ Matrizen und Gleichungen an, die zur Bestimmung der Parameter des Übertragungsglieds nach der MKQ-Methode erforderlich sind (vergl. Abschnitt 3.3.1).

6.3 Gegeben sei eine Gleichung zur Berechnung der Steuerung für ein System, z.B. wie in (25). Leiten Sie aus der Gleichung die Anforderungen ab, die an die Solltrajektorie zu stellen sind.

6.4 Vergewissern Sie sich, dass Sie Aufgaben folgenden Typs lösen können:

Gegeben sei das mathematische Modell eines Systems in Form der Dgl.

$$\ddot{y} = \dot{y} - y^3 + x \qquad\qquad \dot{x} = ax + u.$$

Das System ist flach mit dem flachen Ausgang y. Berechnen Sie einen Ausdruck zur Steuerung des Systems entlang einer vorgegebenen Referenztrajektorie $t \mapsto y_{Ref}(t)$ mittels der Eingangsgröße u.

Lösung hier:

$$u = y^{(3)}_\mathrm{Ref} - \ddot{y}_\mathrm{Ref} + 3y^2_\mathrm{Ref}\dot{y}_\mathrm{Ref} - a(\ddot{y}_\mathrm{Ref} - \dot{y}_\mathrm{Ref} + y^3_\mathrm{Ref})$$

7. Versuchsvorbereitung

Tipp: Teilen Sie sich bei der Vorbereitung auf: Eine Person könnten bspw. separat das Pythonskript vorbereiten. Bereiten Sie desweiteren ein DIN A4 Blatt vor, auf das Sie alle bestimmten Parameter sowie Zeitkonstanten und Verstärkungen der Übertragungsfunktion notieren können.

7.1 Bestimmen Sie allgemein die Ruhelagen des Systemmodells (15) (also ohne konkrete Zahlenwerte für die Parameter). Was fällt Ihnen bezüglich der Ballposition auf? Bereiten Sie Ihre Ergebnisse dabei so auf, dass Sie nach Bestimmung der physikalischen Parameter im Versuch die konkreten Zahlenwerte für die Ruhelagen sofort und übersichtlich berechnen können. Geben Sie den für das Erreichen der Ruhelage erforderlichen PWM-Wert an (vergl. Gl. (12)).

7.2 Linearisieren Sie das Modell (15) um eine geeignete, in der vorangegangenen Teilaufgabe bestimmten Ruhelagen.

7.3 Geben Sie ausgehend von dem linearisierten Modell die Übertragungsfunktion $G_U^H(s)$ an. Formen Sie die Übertragungsfunktion dann so um, dass in Zähler und Nenner nur „Standard-Terme“ wie K, sT, $1+sT$, $1+2dTs+T^2s^2$, etc. vorkommen. Bereiten Sie Ihre Ergebnisse wiederum so auf, dass Sie die Parameter in den Standard-Termen unmittelbar berechnen können, sobald Sie die physikalischen Parameter des Modells experimentell bestimmt haben. Wenn Sie möchten, können Sie die im OPAL bereitliegende Pythondatei `v21_calc_para.py` bzw. die im ILMRT-Portal bereitliegende Jupyter-Notebook-Datei `v21_calc_para.ipynb` verwenden und geeignet anpassen, um die Parameter dann im Versuch schnell berechnen zu können (die Konstruktionsparameter aus Anhang A sind in diesen Dateien bereits eingetragen).

7.4 Bestimmen Sie die Pole und Nullstellen der ermittelten Übertragungsfunktion. Ist die Regelstrecke stabil?

7.5 Machen Sie sich mit der experimentellen Bestimmung der Reglerparameter nach Ziegler und Nichols vertraut (Praktikumseinführung V0, Abschnitt 6.4.1) und notieren Sie sich die nötigen Schritte.

7.6 Machen Sie sich mit der händischen Einstellung eines Regelkreises mit PID-Regler vertraut (Praktikumseinführung V0, Abschnitt 6.3) und notieren Sie sich die nötigen Schritte.

7.7 Wählen Sie aus den Einstellregeln *Betrags- und Symmetrisches Optimum* (Praktikumseinführung V0, Abschnitt 6.5) diejenigen Verfahren aus, die prinzipiell für die in Vorbereitungsaufgabe 7.3 bestimmte Übertragungsfunktion geeignet sind. Begründen Sie Ihre Wahl. Gestalten Sie Ihre Vorbereitung so, dass Sie die Reglerparameter mit

dem gewählten Verfahren unmittelbar berechnen können, sobald sie die „Standard-Terme“ in Vorbereitungsaufgabe 7.3 bestimmt haben (Sie können dazu ebenfalls die Python-Dateien aus Vorbereitungsaufgabe 7.3 nutzen).

7.8 Bereiten Sie ein Python-Script vor, das aus den eingelesenen Messdaten für die Zeit t, die Lüfterdrehzahl $\bar{\eta}$ und deren Ableitung sowie des PWM-Wertes u_{PWM} die Parameter T_{M} und k_{M} des Lüftermodells berechnet. Das Format der Messdatendatei ist in Anhang B definiert. Sie können als Ausgangspunkt die in Abschnitt 5 beschriebene Vorlage verwenden.

Zum Test Ihres Skriptes können Sie die Datei `v21_testdaten.csv` nutzen, die das gleiche Format wie die Datei im Versuch hat. Diese Datei steht im OPAL zum Download bereit. Für diese Datei wurden die Parameter $T = 500\,\text{ms}$ und $k_{\text{M}} = 0.1\,\text{s}^{-1}$ verwendet. Zur Orientierung finden Sie das Ergebnis der Identifikation in Abbildung 5.

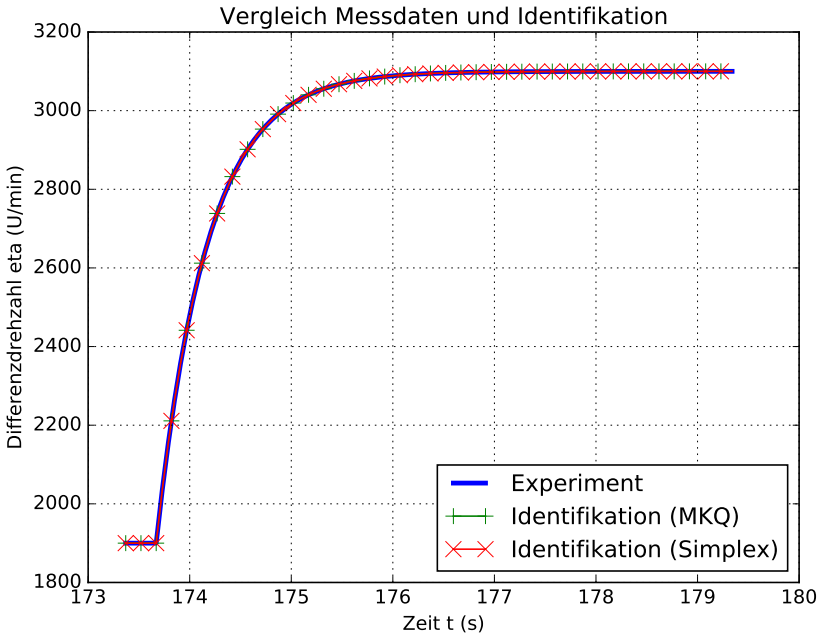

Abbildung 5: Identifikationsergebnis für das Beispiel aus Vorbereitungsaufgabe 7.8.

7.9 Lesen Sie sich die Aufgaben zur Versuchsführung in Abschnitt 8 durch.

8. Versuchsdurchführung

8.1 Stellen Sie die Lüftergeschwindigkeit von Hand so ein, dass die Position des Balls auf 60, auf 80 und auf 100 cm verbleibt. Notieren Sie sich die ermittelten PWM-Werte und die zugehörigen Lüfterdrehzahlen. Was fällt Ihnen auf?

8.2 Führen Sie entsprechend Abschnitt 3 die Experimente zur Datenaufnahme für die Bestimmung der Parameter η_0, k_L, k_V, k_M und T_M durch. Sobald alle Daten erfasst wurden, bestimmt eine Person der Praktikumsgruppe die Parameter, während die anderen Gruppenmitglieder mit den Aufgaben 8.3 und 8.4 fortfahren. Von den aerodynamischen Parametern müssen Sie nur k_L (Aufgabe 8.2.1) oder k_V (Aufgabe 8.2.2) bestimmen. Welches von beiden, sagt Ihnen Ihr Betreuer und versorgt Sie mit dem Wert des nicht zu bestimmenden Parameters für die weiteren Rechnungen.

8.2.1 Datenaufnahme zur Bestimmung von k_L nach Abschnitt 3.1: Stellen Sie die Lüftergeschwindigkeit manuell so ein, dass der Ball bis ans obere Ende des Rohres gehoben wird. Arretieren Sie den Ball in dieser Position mit der vorgesehenen Vorrichtung und schalten Sie den Lüfter dann aus (Stecker Spannungsversorgung ziehen, $u_\mathrm{PWM} = 0$ reicht nicht!). Lösen Sie die Arretierung, sobald der Lüfter steht. Drucken Sie sodann nur den relevanten Verlauf der Ballposition aus.

8.2.2 Datenaufnahme zur Bestimmung von k_V nach Abschnitt 3.2: Sorgen Sie dafür, dass der Ball am unteren Ende des Rohres liegt. Geben Sie dann einen Sprung des u_PWM-Wertes von 0 auf 200 vor. Stoppen Sie die Datenaufzeichnung, sobald der Ball am oberen Ende des Rohres angekommen ist. Notieren Sie sich die Lüfterdrehzahl, bei der der Ball mit konstanter Geschwindigkeit steigt (Zoom ins Diagramm). Drucken Sie nur den relevanten Verlauf der Ballposition aus.

8.2.3 Datenaufnahme zur Bestimmung von T_M und k_M: Registrieren Sie die Antwort der Lüfterdrehzahl auf einen Sprung von u_PWM von 0 auf 200. Stoppen Sie die Datenaufzeichnung, sobald der Ball am oberen Ende des Rohres angekommen ist. Notieren Sie sich die Werte für die Lüfterdrehzahl für $u_\mathrm{PWM} = 0$ und $u_\mathrm{PWM} = 200$, wobei im letzteren Fall der Wert zu nehmen ist, *bevor* der Ball das obere Ende des Rohres erreicht hat. Exportieren Sie die Daten in einer csv-Datei. Drucken Sie sodann nur die relevanten Verläufe für die Lüfterdrehzahl und den u_PWM-Wert aus.

8.2.4 Bestimmen Sie aus den aufgenommen Daten die Modellparameter sowie die Parameter der in Vorbereitungsaufgabe 7.3 hergeleiteten Übertragungsfunktion.

Stellen Sie nachfolgend einen geeigneten Arbeitspunkt für den Lüfter-PWM-Wert u_PWM ein (vergl. Durchführungsaufgabe 8.1). Setzen Sie dazu ggf. auch den I-Anteil der Regler zurück.

8.3 Bestimmen Sie für einen Führungsgrößensprung von 60 auf 80 cm Ballhöhe die Parameter für einen PID-Regler nach dem experimentellen Verfahren von Ziegler und Nichols (vergl. Vorbereitungsaufgabe 7.5). Dokumentieren Sie das Vorgehen.

Nehmen Sie anschließend die Führungs- und die Störsprungantwort des geschlossenen Regelkreises auf.

8.4 Optimieren Sie von Hand die PID-Reglerparameter für einen Führungsgrößensprung von 60 auf 80 cm Ballhöhe. Sie können von den Parametern ausgehen, die Sie nach Ziegler und Nichols bestimmt haben oder nach dem Verfahren aus Vorbeitungsaufgabe 7.6 vorgehen.

Nehmen Sie anschließend die Führungs- und die Störsprungantwort des geschlossenen Regelkreises auf.

8.5 Stellen Sie den Regler nach dem in Vorbereitungsaufgabe 7.7 gewählten Verfahren unter Nutzung der in Aufgabe 8.2.4 bestimmten Parameter ein und nehmen Sie die Führungs- und die Störsprungantwort des geschlossenen Regelkreises auf.

Nachfolgend verwenden Sie bitte die Reglerparameter, die auf Basis der bislang gewonnenen Experimentalergebnisse Ihrer Meinung nach das beste Führungsverhalten erzeugen. Geben Sie für die Steuerung die in Aufgabe 8.2.4 bestimmten Parameter ein.

8.6 Stellen Sie als Solltrajektorie für die Ballposition einen polynomialen Übergang von 0 auf 100 cm innerhalb von 5 s ein und nehmen Sie das Systemverhalten mit dieser Solltrajektorie für folgende Konfigurationen auf:

- Regelung alleine mit Arbeitspunkt $u_{\mathrm{PWM}} = 0$,
- Regelung alleine mit Arbeitspunkt $u_{\mathrm{PWM}} \neq 0$,
- Steuerung alleine mit Arbeitspunkt $u_{\mathrm{PWM}} = 0$,
- Regelung und Steuerung mit Arbeitspunkt $u_{\mathrm{PWM}} = 0$.

8.7 Wählen Sie als Solltrajektorie für die Ballposition eine harmonische Schwingung mit einer Amplitude von 20 cm, einem Offset von 80 cm und einer Periodendauer von 5 s. Nehmen Sie das Systemverhalten für folgende Konfigurationen auf:

- Regelung alleine mit Arbeitspunkt $u_{\mathrm{PWM}} = 0$,
- Regelung alleine mit Arbeitspunkt $u_{\mathrm{PWM}} \neq 0$,
- Regelung und Steuerung mit Arbeitspunkt $u_{\mathrm{PWM}} = 0$.

A. Kennwerte und Parameter

Beschreibung	**Parameter**	Wert
Ballquerschnittsfläche	A_B	$2{,}8274 \cdot 10^{-3}\,\mathrm{m}^2$
Rohrquerschnittsfläche	A_R	$3{,}2573 \cdot 10^{-3}\,\mathrm{m}^2$
Luftspaltquerschnittsfläche	A_SP	$0{,}4299 \cdot 10^{-3}\,\mathrm{m}^2$
Balldurchmesser	d_B	$60 \cdot 10^{-3}\,\mathrm{m}$
Rohrinnendurchmesser	d_R	$64{,}4 \cdot 10^{-3}\,\mathrm{m}$
Erdbeschleunigung	g	$9{,}81\,\frac{\mathrm{m}}{\mathrm{s}^2}$
Masse des Balls	m	$2{,}8 \cdot 10^{-3}\,\mathrm{kg}$

Tabelle 1: Parameter des Versuchsstandes.

Daten	**Wert**
Abmaße	$70 \times 70 \times 20\,\mathrm{mm}$
Betriebsspannung	$12\,\mathrm{V}$
Strom	$0{,}45\,\mathrm{A}$
Anschluss	4-poliger Stecker mit
	Tachosignal und PWM-Steuereingang
Lüfterdrehzahl	2000 - 6000 $\frac{1}{\mathrm{min}}$
Volumenstrom	34 - 102 $\frac{\mathrm{m}^3}{\mathrm{h}}$
Geräuschpegel	20 - 40 dB

Tabelle 2: Parameter des Lüfters vom Typ FOXCONN PVA070E12L.

B. Spezifikation der Messdatendatei

Das Computerprogrammm zur Steuerung des Praktikumsversuches erlaubt es, die Ergebnisse eines Experiments als csv-Datei zu exportieren. In dieser Datei werden die Messdaten in Spalten durch Tabulator getrennt gespeichert, d.h. jede Spalte enthält den Verlauf einer Messgröße. Die Namen der Messgrößen finden sich in der ersten Zeile der Datei, ebenfalls durch Tabulator getrennt. Tabelle 3 listet die aufgenommen Daten, deren Positionen in der Datei sowie die Einheiten auf, in der die Werte gespeichert sind. Für die Weiterverarbeitung sind ggf. Umrechnung in SI-Einheiten erforderlich.

Spalte	ID	Bezeichnung	Einheit
0	time	Zeit	s
1	position	Ballposition	cm
2	dot_position	Geschwindigkeit Ball	cm/s
3	ddot_position	Beschleunigung Ball	cm/s^2
4	desired_position	Soll-Ballposition	cm
5	error	Regelabweichung Ballposition	cm
6	fan_pwm	PWM-Wert Lüfterspannung	–
7	fan_speed	Lüfterdrehzahl	min^{-1}
8	dot_fan_speed	1. Zeitableitung Lüfterdrehzahl	min^{-1}/s
9	ddot_fan_speed	2. Zeitableitung Lüfterdrehzahl	min^{-1}/s^2
10	P	P-Anteil Regler	–
11	I	I-Anteil Regler	–
12	D	D-Anteil Regler	–

Tabelle 3: Struktur der Messdatendatei.

Literatur

[1] L. Böswirth und S. Bschorer. *Technische Strömungslehre: Lehr- und Übungsbuch*. Springer-Verlag, 2010.

[2] C. Burggraf. *Konzeption und Aufbau eines Versuchsstandes zur vertikalen Positionsregelung eines Balls in einem luftdurchströmten Rohr*. Studienarbeit. TU Dresden, 2015.

[3] J.A. Nelder und R. Mead. „A simplex method for function minimization“. In: *The Computer Journal* 7 (1965), S. 308–313.

[4] J. Winkler, C. Knoll, K. Röbenack, C. John und C. Grüning. *Einführung in das Regelungstechnische Praktikum: Grundbegriffe – Hilfsmittel – Kontrollfragen – Gerätetechnik (Fassung Wintersemester 2019/20)*. Erhältlich im OPAL-Dokumentenordner des Praktikums. 2019/20.

[5] M.H. Wright. „Direct Search Methods: Once Scorned, Now Respectable“. In: *Proc. of the 1995 Dundee Biennial Conf. in Numerical Analysis*. Hrsg. von D.F. Griffiths und G.A. Watson. Harlow, UK: Addison Wesley Longman, 1996, S. 191–208.

Autoren/ Korrektoren der Anleitung:
Dr.-Ing. J. Winkler, Dr.-Ing. C. Knoll, Dipl.-Ing. C. Burggraf

Ausgabe:
Oktober 2019