Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Felix Christoph Tiede

Automatisierte Identifikation und Reglersynthese für Regelstrecken der Gebäudeautomation

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Felix Christoph Tiede

Automatisierte Identifikation und Reglersynthese für Regelstrecken der Gebäudeautomation

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Michael Erhard
Zweitgutachter : Prof. Dr.-Ing. Karl-Ragmar Riemschneider

Abgegeben am 7. Juni 2018

Felix Christoph Tiede

Title of the Bachelorthesis

Automated System Identification and Controller Synthesis for Controlled Systems in Buildung Automation Applications

Keywords

system identification, controller synthesis, BACnet, building automation, ARX-systems

Abstract

This thesis describes the development of a software for system identification and controller synthesis of controlled systems in building automation. The identification process is conducted using an ARX system model and a “Least Squares Fitting“ method. Excitation and recording of sample time-series is being performed using the BACnet network protocol. The performance of the implemented procedures is evaluated using a choice of three controlled systems.

Felix Christoph Tiede

Titel der Arbeit

Automatisierte Identifikation und Reglersynthese für Regelstrecken der Gebäudeautomation

Stichworte

Systemidentifikation, Reglersynthese, BACnet, Gebäudeautomation, ARX-Systeme

Kurzzusammenfassung

Diese Arbeit beschreibt die Entwicklung eines Programms zur automatisierten Anregung, Analyse und Reglerdimensionierung für Strecken der Gebäudeautomation. Die Systemidentifikation erfolgt im Zeitbereich mit einem „Least Squares Fitting“ für ein ARX-Systemmodell. Die Kommunikation erfolgt über das BACnet-Netzwerkprotokoll. Die Funktionsfähigkeit des implementierten Verfahrens wird anhand von Beispielmessungen an drei realen Systemen untersucht und diskutiert.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Danksagung

Ich möchte mich hiermit bei allen Personen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben.

Dabei möchte ich mich zunächst bei Prof. Dr. Michael Erhard bedanken, der mich mit großem Engagement und Zeitaufwand betreut und beraten hat. Zusätzlich ermöglichte er mir, in den Laboren der HAW-Hamburg eine Vielzahl von Messversuchen durchzuführen.

Außerdem möchte ich mich bei Benjamin Diers bedanken, der auch nach langen Arbeitstagen stets bereit war mir Rückmeldungen zu meiner Arbeit zu geben und sich fachlich mit mir auszutauschen.

Nicht zuletzt gilt großer Dank meiner Familie, insbesondere Annette Kaiser-Tiede, Simon Tiede und Ralph Tiede. Alle drei haben großes Interesse an meiner Arbeit gezeigt und mich stets unterstützt und motiviert.

Schlussendlich möchte ich mich bei meinem derzeitigen Arbeitgeber, der Firma Siemens AG, bedanken. Von dort habe ich Hardware zum Testen zur Verfügung gestellt bekommen, und ich wurde teilweise freigestellt, um die vorliegende Arbeit produzieren zu können. Auch bei einem Kunden der Siemens AG Building Technologies möchte ich mich bedanken. Dieser möchte namentlich nicht erwähnt werden, hat es mir jedoch ermöglicht, die Ergebnisse dieser Arbeit an realen Anlagen in „echten“ Gebäuden zu testen.

1 Einführung

In modernen Gebäuden kommt eine große Vielfalt von Mess-, Steuerungs- und Regelungstechnik (MSR) zum Einsatz. Um Komfort und Energie- sowie Kosteneffizienz in einem Gebäude zu maximieren, sind teilweise komplexe technische Anlagen notwendig. Von den Primäranlagen der Heizungs-, Lüftungs- und Klimatechnik bis zur Einzelraumregelung können heute eine Vielzahl an Regelstrecken angetroffen werden. Diese erstrecken sich über die verschiedenen Gewerke und werden häufig mit digitalen Gebäudeautomationscontrollern geregelt. In einem einzigen Gebäude können so schnell einige hundert Regelkreise zusammenkommen.

In der Praxis ist das Bestimmen der Reglerparameter für alle diese Regelstrecken mit erheblichen Aufwand verbunden. Besonders der Zeitaufwand durch einen qualifizierten Techniker mit regelungstechnischer Ausbildung ist hier ein Kostentreiber. Aufgrund des großen Kostendrucks in der Gebäudeautomation wird häufig auf eine genaue Bestimmung der Reglerparameter verzichtet. Es werden teilweise Standardparameter aus Software-Bibliotheken für bestimmte Anlagen oder von ähnlichen Anlagen übernommen. Diese Parameter werden also bei vielen Anlagen nicht genau auf die tatsächlich vorliegende Regelstrecke angepasst. Im Betrieb der Anlagen führt dies mitunter zu unerwünschtem Verhalten der Regelstrecken. Es kommt zum Schwingen der Regelgrößen oder die Zeitkonstanten der geregelten Anlage werden größer als gewünscht und technisch möglich.

Ziel dieser Arbeit ist es, die Bestimmung von Reglerparametern weitestgehend zu automatisieren. Zu diesem Zweck soll eine Software entwickelt werden. Diese soll in der Lage sein mit den eingesetzten Automationscontrollern (siehe Abschnitt 2.1.1) zu kommunizieren, die angeschlossenen Regelkreise zu analysieren und anschließend eine Empfehlung für geeignete Reglerparameter aussprechen.

2 Analyse der Ausgangssituation

Um eine geeignete Lösung für das eingangs beschriebene Problem zu finden, ist es notwendig zunächst die Ausgangssituation zu betrachten. Auf dieser Analyse beruht die weitere Arbeit, da hier wichtige Vorbedingungen für die zu treffenden Entscheidungen definiert werden.

2.1 Rahmenbedingungen

Zunächst sollen die Rahmenbedingungen für das in dieser Arbeit erstellte Produkt dargelegt werden. Diese haben erheblichen Einfluss auf die getroffenen Designentscheidungen bei der entwickelten Software. Im Folgenden werden sie als Thesen dargelegt.

Anwenderzielgruppe für das erstellte Reglersyntheseprogramm

Das im Rahmen dieser Arbeit entwickelte Programm zur Bestimmung von Reglerparametern richtet sich an qualifizierte Techniker eines MSR-Betriebs. Die Anwender haben eine grundlegende regelungstechnische Ausbildung erhalten. Sie sind z.B. in der Lage einzuschätzen, ob eine Strecke mit I-Verhalten für $t \rightarrow \infty$ vorliegt, und können die Ergebnisse des Syntheseverfahrens beurteilen.

2.1.1 Verwendete Hardware und Software

In Abbildung 2.1 ist der technische Systemkontext des zu erstellenden Programms dargestellt. Dort sind die wesentlichen Hard- und Software Rahmenbedingungen dargestellt, auf die das Programm angepasst werden muss. Das zu erstellende Programm soll auf den Dienstleistungsnotebooks des MSR-Betriebs lauffähig sein. Als Betriebssystem kommt dabei Windows 7 zum Einsatz. Es wird angestrebt, möglichst wenige externe Softwarepakete- und Schnittstellen zu nutzen, um zukünftig Kompatibilitätsprobleme zu vermeiden.

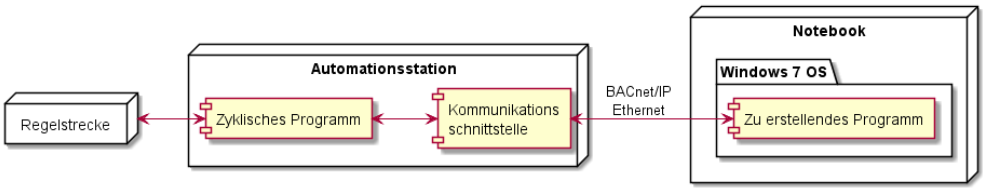

Abbildung 2.1: Systemkontext des zu erstellenden Programms

Die Entwicklung der Software wird mit Automationsstationen der Desigo^{TM} PXC..E-D Reihe der Firma Siemens AG durchgeführt. Es handelt sich dabei um ethernetfähige Automationscontroller für den Einsatz in der Gebäudeautomation. Diese Produktreihe ist hauptsächlich für den Einsatz in den Primärgewerken vorgesehen (Siemens Schweiz AG, 2015a, S.122). Allerdings können diese Controller auch in den Sekundärgewerken (z.B. Licht-, Heizung- und Lüftungsregelung) der Raumautomation eingesetzt werden.

Schnittstellen vom Programm zur Regelstrecke

Die Schnittstelle zwischen der PC-Software und dem Automationscontroller bildet das Netzwerkprotokoll BACnet/IP, das hier über Ethernet übertragen wird. BACnet ist ein offenes Kommunikationsprotokoll (Wosnitza und Hilgerst, 2012, S. 398), eine Kommunikation mit anderen Produkten (auch anderer Produktlinien des gleichen Herstellers) ist damit prinzipiell möglich. Um den Umfang dieser Arbeit zu beschränken, wird das erstellte Programm nicht im Zusammenspiel mit Controllern anderer Hersteller oder Produktreihen getestet. Die Nutzung der offenen Netzwerkschnittstelle ermöglicht dies jedoch. In Siemens Produkten bestehende proprietäre Erweiterungen des BACnet-Protokolls werden nach Möglichkeit nicht genutzt, um eine möglichst große Kompatibilität des Programms mit anderen Produkten zu gewährleisten.

Die Software auf der Automationscontroller lässt sich, wie in Abbildung 2.1 dargestellt, in zwei Hauptkomponenten unterteilen: Das zyklische Programm („D-MAP Programm“) und eine parallel betriebene Kommunikationsschnittstelle. Das zyklische Programm berechnet im Betrieb der Automationscontroller die regulären MSR-Funktionen. Ein- und Ausgangswerte werden zu Beginn eines Zyklus gelesen und zum Ende geschrieben. Dazwischen erfolgt keine Aktualisierung der Ein- und Ausgangsgrößen im Automationscontroller (Siemens Schweiz AG, 2015a, S. 258-267).

Die Zykluszeit stellt einen entscheidenden Faktor in der sinnvoll zu erreichenden Abtastzeit dar. Da die Kommunikationsschnittstelle unabhängig vom eigentlichen MSR-Prozess

läuft, ist ein Abfragen der Prozessgrößen im Speicher des Automationscontrollers in kürzeren Intervallen als dessen Zykluszeit möglich. Dieses Vorgehen ist allerdings nicht sinnvoll, da keine Wertänderungen auftreten und eine höhere als die tatsächliche Abtastfrequenz suggeriert würde.

In den zur Entwicklung verwendeten Desigo™ PXC..E-D Controllern ist eine minimale Zykluszeit einstellbar. Diese liegt standardmäßig bei 250ms. Liegt die tatsächliche Ausführungszeit unter dieser Schwelle, wird die Ausführung des zyklischen Programms bis zum Ablauf der Zeit pausiert (Siemens Schweiz AG, 2015a, S. 258). Theoretisch ist eine geringere minimale Zykluszeit (bis minimal 100ms) in der Software parametrierbar. Eine Änderung dieser Einstellung ist allerdings in der Praxis unüblich.

Bei aufwändiger Programmierung des zyklischen MSR-Programms ist es denkbar, dass die Zykluszeiten höher als die minimale Zykluszeit sind. In den untersuchten Controllern ist zusätzlich eine maximale Zykluszeit einstellbar. Bei den untersuchten Controller liegt diese Zeit bei 1000ms. Ist sie überschritten, wird nach einem Zyklus schnellstmöglich der nächste Zyklus abgearbeitet (Siemens Schweiz AG, 2015a, S. 258). Das bedeutet, dass die externe Kommunikation ggf. eingeschränkt wird, um die Ausführung des eigentlichen MSR-Programms sicherzustellen.

Die tatsächliche Zykluszeit lässt sich bei den untersuchten Automationscontrollern über ein proprietäres Eigenschaftsfeld des entsprechenden BACnet-Objekts auslesen. Wie oben bereits erwähnt, wird aus Kompatibilitätsgründen allerdings ein Verzicht auf proprietäre Funktionen angestrebt. Aus diesem Grunde und zur Vereinfachung des zu lösenden Problems wird im Folgenden von einer konstanten Zykluszeit von

$$T_{cylce} = 250ms \tag{2.1}$$

ausgegangen. Daraus resultiert, dass im Folgenden auch eine maximale Abtastfrequenz von

$$f_{A,max} = 4Hz \tag{2.2}$$

gewählt werden darf. Ansonsten kommt es durch Überabtastung dazu, dass mehrmals der gleiche Wert eingelesen wird, ohne dass eine weitere Sensorabfrage stattgefunden hat.

Priorität der Schreibzugriffe

Beim Schreiben eines Wertes auf ein BACnet Object muss stets eine Priorität $1 \ldots 16$ ausgewiesen werden, mit der der Schreibbefehl ausgegeben werden soll. Ein kleiner

Zahlenwert steht dabei für eine hohe Priorität. Der zu schreibende Wert wird dann in der sogenannten Prioritätsmatrix eingetragen. Als aktueller Ausgangswert wird aus den aktiven Einträgen in der Prioritätsmatrix stets der mit der höchsten Priorität ausgewählt (Siemens Schweiz AG, 2015a, S. 269).

Priorität	Bedeutung
1	Sicherheitswert (Personenschutz, lokal)
2	Sicherheitswert (Personenschutz, übergeordnet)
3	In Desigo nicht benutzt
4	Kritischer Wert (Anlagenschutz, lokal)
5	Kritischer Wert (Anlagenschutz, übergeordnet)
6	Minimale Ein-/Ausschaltzeit
7	Bedienwert (lokal)
8	Bedienwert (übergeordnet)
9-13	In Desigo nicht benutzt
14	Programmwert (übergeordnet)
15	Programmwert (lokal)
16	Programmwert (allgemeine Befehle)
17	Vorgabewert (Default)

Tabelle 2.1: Struktur der Prioritätsmatrix in Desigo^{TM} 6.0 (Siemens Schweiz AG, 2015a, S.271-273, gekürzt)

Tabelle 2.1 zeigt die im Desigo^{TM} Gebäudeautomationssystem vorgesehene Verwendung der verschiedenen Prioritäten. Einige dieser Verwendungen sind durch den BACnet-Standard definiert (siehe Kranz (2013), S. 319), andere sind für die genannte Produktlinie so definiert worden. Für diese Arbeit wird sich an den in Tabelle 2.1 dargestellten Verwendungen orientiert.

Das zu erstellende Reglersyntheseprogramm soll in der Lage sein die Regelstrecke selbsttätig anzuregen, um eine breitbandige Erregung der Regelstrecke sicher zu ermöglichen (siehe Abschnitt 3.1). Aus diesem Grunde müssen die erforderlichen Schreibzugriffe mit einer höheren Priorität als der des regulären Programmwertes des lokal ausgeführten zyklischen Programms (Priorität 16) vorgenommen werden. Andererseits darf auch keine Gefährdung von Personen und Sachwerten eintreten. Zum Schutz dieser sind die Prioritäten 1,2,4 und 5 vorgesehen.

Ein Beispiel für den Anlagenschutz stellt der Frostschutz für Lüftungsanlagen da. Nähert sich die Temperatur im Zuluftkanal dem Gefrierpunkt, wird die Anlage automatisch abgeschaltet, die Außenluftklappen werden geschlossen und Heizventile werden maximal geöffnet.

Priorität 6 ist für die Einhaltung von parametrierten minimalen Ein-/Ausschaltzeiten von Ausgängen vorgesehen. Mit dieser Priorität wird der aktuelle Zustand weiterhin gehalten, falls eine Umschaltung vor Ablauf der Mindestzeit angefordert wurde. Auch dies kann unter Umständen eine Sicherheitsfunktion zum Anlagenschutz darstellen. Für das zu erstellende Programm wird entschieden die Priorität 8 zu nutzen. Diese Priorität ist laut BACnet Standard für einen manuellen Eingriff in die Steuerung und Regelung vorgesehen (Kranz, 2013, S. 319).

2.1.2 Organisatorische Rahmenbedingungen

Beim Entwurf des Programms gab es weitere beschränkende Rahmenbedingungen zu beachten. So soll die Parameterbestimmung für ein bestehendes Programm im Automationscontroller erfolgen. D.h. es können zwar Parameter angepasst werden, die bestehende Programmstruktur soll aber erhalten bleiben. Dies ist damit begründet, dass das Programm für bereits in Betrieb befindliche Anlagen eingesetzt werden soll. Ziel ist es, die Güte der Regelung mit geringem Zeitaufwand zu verbessern. Weitreichende Programmänderungen lassen sich nicht so einfach automatisieren und erfordern eine komplexere Qualitätssicherung. Außerdem soll die Komplexität des MSR-Programms erhalten bleiben, um die Auslastung der Automationscontroller konstant zu halten. Bei den verwendeten Automationscontrollern bedeutet dies, dass z.B. lediglich ein sogenannter „PID“-Reglerbaustein zur Verfügung steht. (Tatsächlich handelt es sich aus Gründen der technischen Realisierbarkeit (Unbehauen, 2007, S. 251) um einen PID-T_{1R} Regler). In diesem Baustein lassen sich die Reglerparameter Reglerverstärkung K_P, Nachstellzeit T_N und Vorhaltezeit T_V über BACnet direkt parametrieren. Der integrale und differentiale Anteil des Reglerbausteins lässt sich jeweils durch das Nullsetzen der entsprechenden Zeitkonstante im laufenden Betrieb deaktivieren. Andere als die daraus resultierenden Reglertypen sollen hier nicht betrachtet werden.

Aus der Bedingung, dass das Programm nicht geändert werden darf, resultiert auch, dass kein adaptives Verfahren zur Identifikation eingesetzt werden kann. Bei einem solchen Verfahren erfolgt im Betrieb eine laufende Optimierung und Adaption der Reglerparameter (Bohn und Unbehauen, 2016, S. 207ff). Außerdem sind solche Verfahren auf den eingesetzten Controllern bereits als Firmware-Bausteine verfügbar (Siemens Schweiz AG, 2015a, S. 10/11). Bei Bedarf können diese durch Änderung des Programms auf dem Controller genutzt werden.

Diese Arbeit beschäftigt sich hingegen mit den Regelstrecken, bei denen der reguläre „PID“- Reglerbaustein eingesetzt wird. Ein Online-Verfahren ließe sich hier zwar auch über die BACnet Schnittstelle realisieren, allerdings würde dies dauerhaft zu einer hohen

Netzwerkauslastung führen. Außerdem müsste das zu erstellende Programm dann dauerhaft ausgeführt werden. Es wird daher stattdessen eine einmalige, zeitlich begrenzte Offline-Identifikation des Systems gewählt. D.h., es werden zunächst Ein- und Ausgangssignale aufgezeichnet und erst anschließend eine Identifikation durchgeführt (Bohn und Unbehauen, 2016, S.11). Da einige Strecken zeitvariante Parameter besitzen (z.B. witterungsbedingt), muss die Identifikation ggf. zu verschiedenen Zeitpunkten wiederholt werden. Es wird im Folgenden davon ausgegangen, dass die Streckenparameter für die Dauer der Identifikation und Auswertung zeitinvariant sind.

2.2 Art der Regelstrecken

In dieser Arbeit wird sich auf den Einsatzbereich der untersuchten Desigo^{TM} PXC..E-D Controller fokussiert. Dort sind diverse Regelstrecken der Heizungs-, Lüftungs- und Klima (HLK)-Technik sowie der Gebäudeautomation zu finden. Dort vorkommende Strecken sind in der Regel „gutmütig“, d.h., sie weisen PT_n Verhalten mit großer Dämpfung auf. Ein PXC..E-D Controller muss zu Wartungszwecken (z.B. Firmwareupdate) regelmäßig gestoppt und urgelöscht werden können. Dabei gehen anstehende Ausgangswerte verloren. Bei Strecken mit integralem Verhalten für $t \rightarrow \infty$ wird die Störung der Strecke durch ein unkontrolliertes Eingangssignal während dieser Zeit integriert. Ein PXC..E-D Controller eignet sich daher eher nicht für die Regelung dieser Strecken und wird auch selten dafür eingesetzt. Diese Strecken sollen daher hier auch nicht untersucht werden.

Liegen in der Praxis kompliziertere Regelstrecken vor, werden diese oft durch unterlagerte Regelkreise an den Automationscontroller angebunden. Dann wird vom übergeordneten Controller lediglich die Führungsgröße vorgegeben. Das ist etwa bei Anlagen zur Kälteerzeugung oder bei Heizkesseln der Fall. Die interne Steuerung kann dort durch ein technisches System des Herstellers erfolgen, während der Gebäudeautomationscontroller z.B. lediglich eine Soll-Temperatur für das Heiz-/Kühlmedium vorgibt.

Zeitkonstanten

Die maximal möglichen Zeitkonstanten werden durch die Zykluszeit der Controller und die resultierende Abtastfrequenz beschränkt. Um das Abtasttheorem (siehe Frey und Bossert (2008) S. 236) nicht zu verletzen, darf die maximal auftretende Frequenz nicht

höher als die Hälfte der Abtastfrequenz liegen. Als Richtwert nennen Bohn und Unbehauen (2016) (siehe S. 504) unter Berufung auf einige Referenzen eine Abtastfrequenz von

$$f_s = 10 \cdot f_b \tag{2.3}$$

f_b gibt dabei die obere Grenze der Bandbreite des zu untersuchenden Systems an. Eine strikte Begrenzung der Bandbreite ist in der Regel nicht vorhanden. Es liegt vielmehr ein stetiger Verlauf des Spektrums vor. Als Kenngröße wird hier also die 3-dB-Bandbreite gewählt. Diese gibt die Frequenz an, bei der eine Dämpfung von $\approx -3,01 dB$ erreicht wird. Bei einem PT_1-Glied handelt es sich dabei um den Kehrwert der Zeitkonstante T_1 (Bohn und Unbehauen, 2016, S. 505). Mit einer maximalen Abtastrate $f_{A,max} = 4Hz$ (Gleichung 2.2) ergibt sich damit die schnellste sinnvolle Streckenzeitkonstante

$$\begin{aligned} T_{\text{min}} &= 10 \cdot \frac{1}{\frac{f_{A,\text{maxx}}}{2}} = 10 \cdot \frac{1}{\frac{4\,\text{sek}^{-1}}{2}} \\ &= \underline{\underline{5\,\text{sek}}} \end{aligned} \tag{2.4}$$

Bei schnelleren Strecken sollte versucht werden, die Strecke vor der Regelung zu verlangsamen. Dazu kann beispielsweise eine Tiefpassfilterung der Regelgröße vorgenommen werden.

Eine obere Grenze der anzutreffenden Zeitkonstanten zu finden, ist nicht ohne Weiteres möglich, da ein sehr breites Anwendungsspektrum vorliegt. Bei übergeordneten Reglungen sind sehr große Zeitkonstanten anzutreffen. In der Firmware der Controller ist etwa ein Programm zur prädiktiven Regelung von Heizkreisen über die Vorgabe der Vorlauftemperatur enthalten. Dieser Baustein adaptiert unter anderem Zeitkonstanten. Einer der voreingestellten Startwerte dazu liegt bei einer Zeitkonstante von $T_{Bldg} = 50h$ (Siemens Schweiz AG, 2015b, S. 64). Dies liefert einen Eindruck über die große Bandbreite möglicher Zeitkonstanten.

Eine Einschränkung der Zeitkonstanten ist für dieses Projekt notwendig, um ein Einsatzgebiet zu definieren, in dem später Tests durchgeführt werden. Dazu wird vom Autor eine Grenze von

$$T_{\text{max}} = 30\text{min} \tag{2.5}$$

gewählt. In dem dadurch festgelegten Bereich sollten etwa typische Heizungsregelungen für Einzelräume noch vertreten sein.

3 Systemidentifikation der Regelstrecke

In diesem Kapitel werden die theoretischen Aspekte der Systemidentifikation einer Regelstrecke untersucht. Die praktischen Aspekte der Implementierung der hier gezeigten Verfahren erfolgt dann in Abschnitt 5.4.

3.1 Anregung der Regelstrecke

Um einen geeigneten Regler für die zu regelnde Strecke auszuwählen, ist es notwendig, die Dynamik des Systems zu untersuchen. Es ist offensichtlich, dass dazu eine gewisse Anregung des Systems notwendig ist, da im stationären Zustand keine Untersuchung des dynamischen Verhaltens des Systems möglich ist. Es ist daher sinnvoll die Regelstrecke während der Identifikation explizit anzuregen. An eine mögliche Anregung der Regelstrecke werden dabei die folgenden Anforderungen gestellt:

I **Fortwährende Erregung**
Um ein lineares System n-ter Ordnung zu identifizieren, muss dieses während der Identifikation mit mindestens $2n$ Frequenzanteilen im Eingangssignal (Stellsignal) angeregt werden (Bohn und Unbehauen, 2016, S. 534). Ist die Bedingung der *fortwährenden Erregung* nicht erfüllt, liefert die Identifikation keine zuverlässigen Werte oder ist nicht lösbar (Bohn und Unbehauen, 2016, S. 69). Bei dem in den folgenden Abschnitten beschriebenen „Least Square Fitting“ wird dabei die Datenmatrix M singulär, und es kann kein Parametervektor bestimmt werden. Da die Modellordnung n nun zunächst unbekannt ist, ist es sinnvoll eine möglichst breitbandige Anregung des Systems durchzuführen, um später auch auf höhere Modellordnungen testen zu können.

II **Anregung um den Arbeitspunkt**
Zur Identifikation des Systems wird im Folgenden ein lineares Systemmodell angesetzt (siehe Abschnitt 3.2). Ein lineares System liegt genau dann vor, wenn es

das Superpositionsprinzip erfüllt. D.h.: „Bei einem linearen System ist die Antwort auf eine Linearkombination von Eingangssignalen gleich der entsprechenden Linearkombination der einzelnen Systemantworten“ (Frey und Bossert, 2008, S. 4). Diese Eigenschaft wird in der Praxis wahrscheinlich von den wenigsten Systemen erfüllt. Dies lässt sich an einem einfachen Beispiel verdeutlichen: Stellt sich etwa an einem Heizkreis zur Raumheizung beim Einstellen einer Ventilöffnung von $y = 10\%$ im Raum eine Temperatur von $20°C$ ein, so müsste sich bei einer Ventilöffnung etwa von $y = 80\%$ $(= 8 \cdot 10\%)$ eine Raumtemperatur von $160°C$ $(= 8 \cdot 20°C)$ einstellen. Das wird offensichtlich nicht der Fall sein.
Aus diesem Beispiel lässt sich leicht schließen, dass es hier sinnvoll ist, eine Linearisierung um einen bestimmten Arbeitspunkt vorzunehmen und im Anschluss für Regel- und Stellgröße mit den Differenzsignalen zu diesem Arbeitspunkt zu arbeiten. Dieser Arbeitspunkt sollte einen typischen Betriebszustand der Anlage widerspiegeln. Für die Anregung der Regelstrecke bedeutet dies, dass es möglich sein muss, einen „Offset“ einstellen zu können, der das System in einen typischen Betriebszustand versetzt. Bei der Identifikation wird dann das System gesucht, dass das Streckenverhalten in hinreichend kleiner Umgebung um diesen Arbeitspunkt beschreibt.

III **Begrenzte Amplitude**
Die Amplitude der Streckenanregung sollte aus zwei Gründen begrenzt werden. Zum einen liegt, wie oben beschrieben, kein lineares Verhalten der Regelstrecke vor. Bei zu großer Abweichung vom Arbeitspunkt lässt sich das System nicht mehr hinreichend gut durch ein lineares Systemmodell beschreiben. Ein extremes Beispiel dazu stellt das Überschreiten des Stellbereichs der Stellgröße dar. Wird etwa bei einer Ventilansteuerung $y = 0 \ldots 100\%$ für die Anregung ein Wert von 150% ausgegeben, wird dieser vom Stellglied nicht umgesetzt werden können. Wie groß die Abweichung um den Arbeitspunkt sein darf, hängt hier wieder von der konkreten Anwendung ab. Dabei spielt etwa die Ausprägung der Nichtlinearität und der typische Arbeitsbereich des Reglers eine Rolle.
Zum anderen kann es sein, dass etwa die Regelgröße auch während der Identifikation einen bestimmten Bereich nicht verlassen darf. Das kann aus komfort-, sicherheitstechnischen und/oder sonstigen Gründen der Fall sein. Auch hier ist es sinnvoll, die Amplitude der Anregung zu begrenzen. Diese Forderung für die Regelgröße umzusetzen, ist nicht trivial, da bei einer unbekannten Strecke keine Erkenntnis vorliegt, wie sich eine Änderung der Stellgröße in eine Änderung der Regelgröße umsetzen wird.

3.1.1 Pseudo-Random-Binary-Sequence-Signale

Sogenannte Pseudo-Random-Binary-Sequence-Signale (PRBS-Signale) erfüllen die oben geforderten Anforderungen: Sie besitzen ein sehr breites Spektrum, sie nehmen lediglich zwei definierte Amplitudenwerte an und lassen sich durch das Hinzufügen eines Offsets um den Arbeitspunkt legen. Der Hauptvorteil der PRBS-Signale gegenüber Gleitsinus, Multisinus und anderen Signalen liegt in der vollen und gleichmäßigen Ausnutzung des möglichen Spektrums bis zur Nyquist-Frequenz (Bohn und Unbehauen, 2016, S. 543), d.h. bis zur halben Abtastrate .

Die Bezeichnung „Pseudo-Random“ (pseudo-zufällig) für das Signal kann missverständlich aufgenommen werden. Es handelt sich nicht etwa um eine stochastisches Signal, sondern um ein deterministisches Signal, das eine fest definierte Abfolge der beiden Amplitudenwerte aufweist.
Die PRBS-Signale werden jeweils über ein sogenanntes *primitives Polynom* über den Galois-Körper definiert (Bohn und Unbehauen, 2016, S. 543). Wie daraus das entsprechende Signal generiert wird, ist in Abschnitt 5.3 dargestellt.

In Abbildung 3.1 wird ein PRBS-Signal („PRBS“) mit einem Zufallssignal verglichen. Das „Zufallssignal“ entspricht dabei dem ersten Signal, ist jedoch um einen Abtastschritt gekürzt. Zusätzlich ist in der Abbildung das gleiche PRBS-Singal mit doppelter Periodendauer dargestellt. Dort ist zu sehen, dass das Spektrum des Zufallssignals zufällig verteilt ist, während die PRBS-Signale (betrachtet über ganzzahlige Vielfache der Periodendauer) eine gleichmäßige Anregung bieten. Für allgemein zufällige Signale ist es nicht möglich, das Signalspektrum vorherzusagen. Wie zu erwarten, liefert das PRBS-Signal mit der doppelten Signallänge im Wesentlichen das gleiche Spektrum wie bei einfacher Periodenlänge. Die durch das längere Signal zusätzlich abgetasteten Frequenzen (siehe auch Abschnitt 3.4) nehmen dabei den Wert „0“ an. Um nun später unterschiedlich aufgelöste Spektren zur Anregung nutzen zu können, können die PRBS-Signale also weder gekürzt, noch durch periodische Wiederholung verlängert werden. Aus diesem Grunde werden in Abschnitt 5.3 PRBS-Signale unterschiedlicher Periodendauern implementiert.

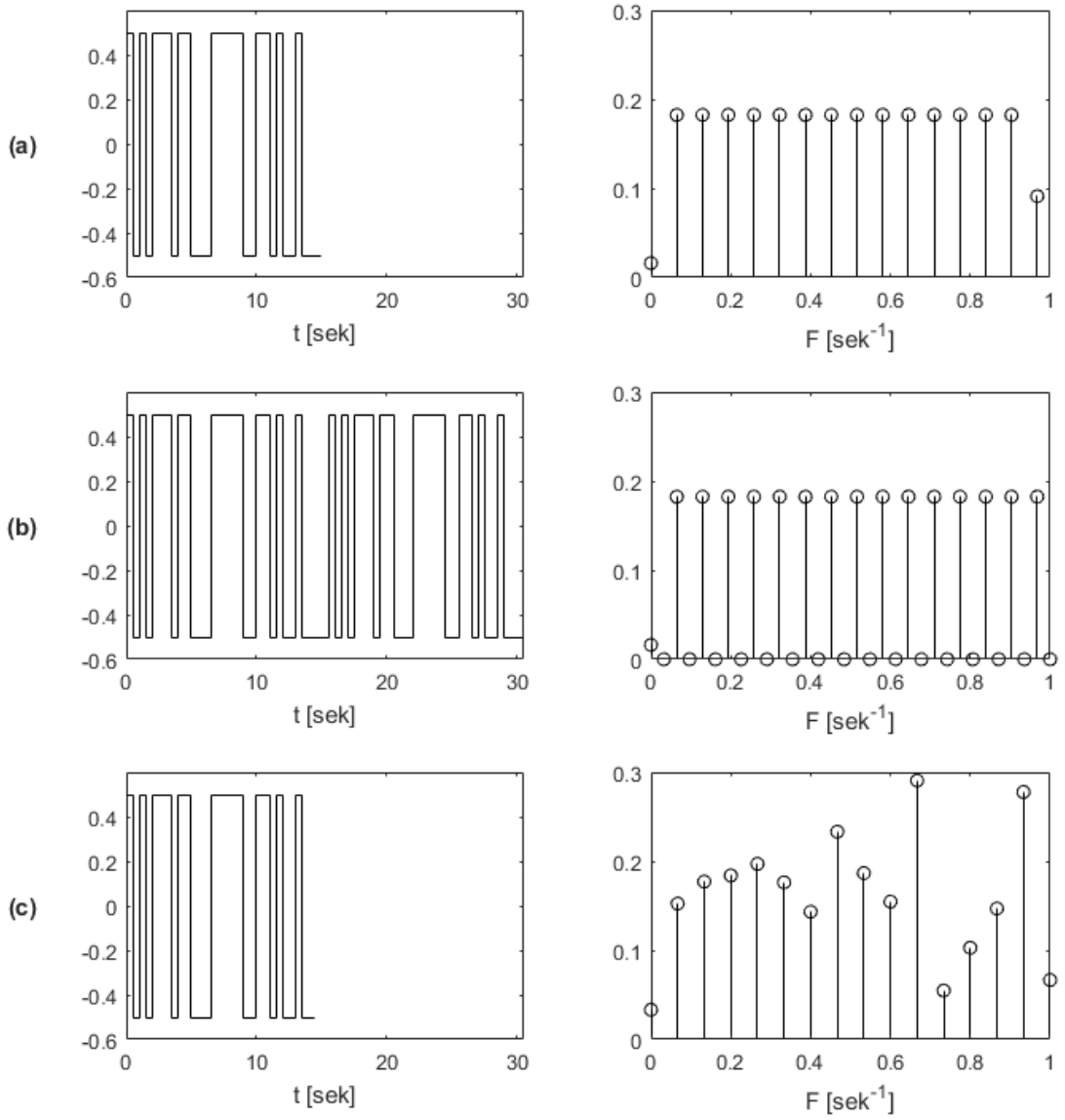

Abbildung 3.1: Qualitatives Zeitsignal (links) und qualitatives einseitiges diskretes Amplitudenspektrum (rechts) (a) eines PRBS-Signals mit einer Periodendauer von 31 Abtastschritten („PRBS“); (b) des gleichen PRBS-Signals über zwei Periodendauern („PRBS2“); (c) eines „Zufallssignals“

3.1.2 Ungeregelte Streckensprungantwort

Bei einer ungeregelten Streckensprungantwort wird auf die Stellgröße der Strecke y ein Sprung gegeben und anschließend die Antwort der Strecke analysiert. Diese Sprungantwort wird oft genutzt, um das dynamische Verhalten eines Systems zu beschreiben. Sie liegt hier direkt vor und eignet sich z.B. auch für eine schnelle Plausibilitätsüberprüfung der ermittelten Zeitkonstanten, da es möglich ist, eine dominierende Zeitkonstante aus dem Signalverlauf abzuschätzen. Dies ist etwa durch die Konstruktion einer Wendetangente in das Ausgangssignal möglich (Bohn und Unbehauen, 2016, S.16).

Diese Form der Anregung bietet allerdings auch erhebliche Nachteile. Zunächst kann es zeitaufwendig sein, bei einem ungeregelten System abzuwarten, bis sich ein bestimmter Arbeitspunkt eingestellt hat. Danach ist es bei unbekannter Strecke nicht möglich abzuschätzen, mit welcher Ausgangsamplitude das System auf eine Eingangsanregung reagieren wird. Es ist hier also schwierig, die Anforderungen II und III aus Abschnitt 3.1 zu erfüllen. Eventuell sind dafür mehrere Versuche notwendig.

Zusätzlich liegt bei einem Eingangsgrößensprung ein Großteil der Signalleistung bei einer Frequenz von $\omega = 0$, also im Gleichanteil des Signals, während höhere Frequenzen kaum vertreten sind (siehe Abbildung 3.3).

3.1.3 Geregelte Führungssprungantwort

Eine andere Möglichkeit die Regelstrecke anzuregen, ist es, die Strecke indirekt über einen bestehenden Regler anzuregen. Soll etwa eine bestehende Regelstrecke im laufenden Betrieb untersucht werden, so ist es möglich, einen Führungsgrößensprung auf die Strecke zu geben und dabei sowohl Regel- als auch Stellgröße zu beobachten, um dann Rückschlüsse auf das Verhalten der Regelstrecke zu ziehen. Ein Vorteil dieses Verfahrens liegt darin, dass genau festgelegt werden kann, auf welchen neuen Wert sich die Regelgröße während des Sprungs einpendeln wird. Außerdem wird durch diese Form der Anregung auch ein typischer Arbeitsbereich der Stellgröße ausgenutzt. Das Streckenmodell wird also für typische Wertebereiche von Stell- und Regelgröße gleichermaßen erstellt. Anforderungen II und III können damit leicht erfüllt werden, da zumindest die Endwerte der Regelgröße hier direkt eingestellt werden können. Voraussetzung dafür ist lediglich eine asymptotische Stabilität des bestehenden Regelkreises. Ein weiterer Vorteil dieser Anregung liegt darin, dass die Messzeit mitunter deutlich verkürzt werden kann, falls der Regelkreis durch den bestehenden Regler bereits schneller als die ungeregelte Streckensprungantwort ist.

Ein Nachteil dieser Form der Anregung liegt darin, dass es nicht möglich ist, eine allgemeine Aussage über das Spektrum des Stellsignales zu treffen, da dies stark von den eingestellten Reglerparametern abhängt. Ob Anforderung I aus Abschnitt 3.1 erfüllt ist, kann also nicht allgemein beantwortet werden.

3.1.4 Zwischenfazit zur Anregung

Nach den vorherigen Abschnitten mag zunächst eine Anregung mittels PRBS-Signalen sinnvoll erscheinen, da hier alle Anforderungen aus Abschnitt 3.1 erfüllt sind. Allerdings ist es hier notwendig, das anregende Signal gut abgestimmt auf die Regelstrecke zu parametrieren. Aus der Anregung mit einem sehr gleichmäßigen Spektrum ergibt sich zwangsläufig, dass auch ein (im Vergleich zu den Sprungantworten) relativ großer Teil der Signalleistung auf das obere Frequenzspektrum entfällt. Bei einer kleinen Abtastzeit kann es nun passieren, dass das System auch oberhalb der Bandbreite des Systems noch stark angeregt wird, während das System hier bereits stark dämpfend wirkt. Diese Frequenzen werden sich damit auch kaum im Ausgangssignal wiederfinden. Die Signalleistung wird also unter Umständen nicht optimal genutzt. Ein weiteres Problem stellt die feste Länge einer PRBS-Sequenz dar, da diese auch vor Beginn der Aufzeichnung der Signale zur Identifikation eingestellt werden muss. Im Gegensatz dazu, kann bei den Sprungantworten in der Regel das Ende des Einschwingvorgangs relativ leicht festgestellt werden, da die Änderung der Regelgröße mit Erreichen des neuen Endwerts langsam abnimmt.

Abbildung 3.2 zeigt zwei Beispiele von möglichen Sprungantworten. Die Aufnahmen stammen dabei von einer realen Regelstrecke. Es ist dort gut zu erkennen, dass bei der ungeregelten Sprungantwort nur ein kleiner Stellbereich genutzt wird, der jedoch eine relativ große Abweichung der Regelgröße verursacht. Außerdem ist deutlich zu sehen, dass sich die Regelgröße ab einer Zeit $t > 40\,\mathrm{sek}$ nur noch langsam ändert und (im Vergleich zu einer gedachten beschränkten Exponentialfunktion durch den Signalverlauf) ein Rauschen im Signal sichtbar wird. Es ist hier schwierig festzustellen, auf welchen Endwert sich die Regelstrecke einpendeln wird. Zusätzlich wirken sich langsame Drift-Effekte in der Strecke hier stark aus. Im Gegensatz dazu wird das System bei der geregelten Führungssprungantwort in kürzerer Zeit mit einer größeren Signalleistung angeregt und der Signalverlauf erscheint klarer.

Die Spektren der Stellgrößen aus Abbildung 3.2 sind in Abbildung 3.3 dargestellt. Dort ist gut zu sehen, dass die geregelte Sprungantwort ein deutlich breiteres Spektrum aufweist. Dies ist allerdings auch dem schwingenden Führungsverhalten geschuldet. Es

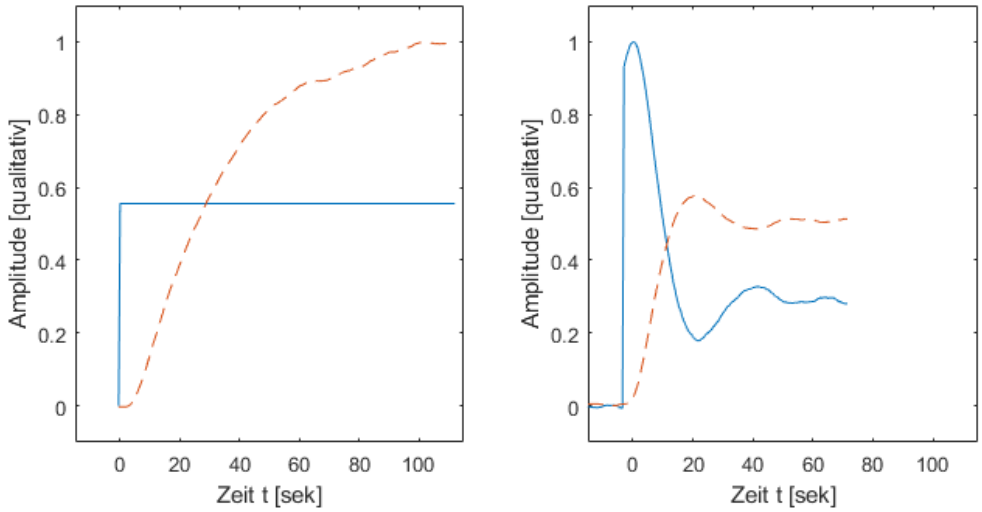

Abbildung 3.2: Ungeregelte Streckensprungantwort und geregelte Führungssprungantwort einer Regelstrecke (Beispiele). Mit Stellgröße (blau, auf einen Maximalwert von $y = 1$ normiert) und Regelgröße (rot, gestrichelt)

handelt sich lediglich um ein willkürliches Beispiel. Andere Reglereinstellungen führen zu anderen Ergebnissen.

Zusammenfassend lässt sich sagen, dass es sinnvoll ist, bei der Analyse einer laufenden Regelstrecke mit einer geregelten Sprungantwort zu beginnen, da hier eine geringe und vor allem eine vorhersehbare Beeinträchtigung des Betriebs auftritt. Bei ungünstigen Reglerwerten und zur Verifikation der Ergebnisse kann eine ungeregelte Streckensprungantwort gewählt werden. Bei gehobenen Anforderungen und falls eine explizite Untersuchung des oberen Frequenzbereichs erwünscht ist, kann eine PRBS-Anregung gewählt werden.

Da alle Formen der Anregung Vor- und Nachteile bieten, werden alle implementiert. Eine Darstellung der, mit den verschiedenen Verfahren erzielten, Syntheseergebnisse erfolgt in Kapitel 6.

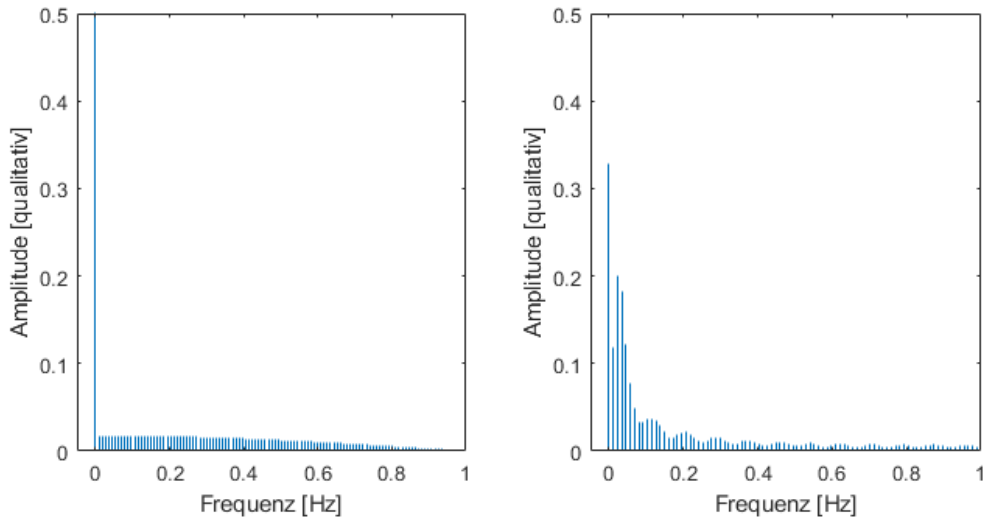

Abbildung 3.3: Spektrum der Stellgrößen aus Abbildung 3.2. Die Signale wurden jeweils vor der Fourieranalyse auf einen Maximalwert von $y_{i,max} = 1$ normiert

3.2 Wahl der Modellstruktur

Das zu erstellende Programm soll eine möglichst große Zahl an Systemen ohne systemspezifische Vorkenntnisse identifizieren können. Diese Forderung entspricht einem „Black Box“ Ansatz zur Identifikation, bei dem keine physikalischen Zusammenhänge des Systems und somit auch keine mathematische Beschreibung des Systems bekannt ist (Bohn und Unbehauen, 2016, S. 5). Prinzipiell kann das untersuchte System über verschiedene Modellarten eindeutig identifiziert werden.

Für die spätere Auslegung eines Reglers für die zu identifizierende Strecke ist es hier wünschenswert, das System durch eine Übertragungsfunktion zu beschreiben. Liegt die Übertragungsfunktion als gebrochen-rationale Funktion vor, lassen sich auch ihre Pole bestimmen. Die Kenntnis über die Pole bietet dann einige Vorteile: So lässt sich anhand der Pole später leicht die Stabilität des Systems untersuchen. Zusätzlich können bei einer späteren Reglerdimensionierung Zeitkonstanten (die aus den Polen berechnet werden) kompensiert werden. Schlussendlich liefert die Darstellung des Systems durch eine Übertragungsfunktion die Möglichkeit mit relativ wenig Aufwand Simulationen des Streckenverhaltens durchzuführen. Da die Identifikation auf einem Computer durchgeführt wird, liegen die Ein- und Ausgangssignale zeit-diskret abgetastet vor. Es bietet sich also an zunächst mit einem zeit-diskreten Modell zur Identifikation zu arbeiten.

Um eine Übertragungsfunktion aufstellen zu können, ist es notwendig die Struktur dieser festzulegen. Dazu beschreiben Bohn und Unbehauen (2016) das in Abbildung 3.4 dargestellte „allgemeine Regressionsmodell“. In diesem Modell wird davon ausgegangen, dass sich der Systemausgang $Y(z)$ aus einem deterministischen Anteil $\tilde{Y}_M(z)$, überlagert mit einem stochastischen Fehlersignal $R_M(z)$, zusammensetzt (Bohn und Unbehauen, 2016, S. 61). Die Terme $A(z^{-1})$ bis $E(z^{-1})$ stellen dabei z-Polynome dar.

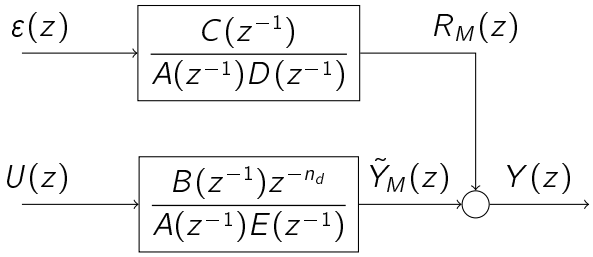

Abbildung 3.4: „Allgemeines Regressionsmodell“ (Bohn und Unbehauen, 2016, S. 64)

Je nach Form dieser Polynome lassen sich verschiedene Sonderfälle der allgemeinen Struktur ableiten (siehe Bohn und Unbehauen (2016), Tabelle 3.2).

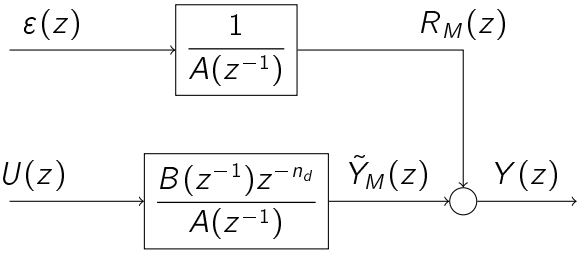

Abbildung 3.5: „Autoregressive model with exogenous input“ („ ARX“) Systemmodell abgeleitet aus „Allgemeinen Regressionsmodell“ nach (Bohn und Unbehauen, 2016, S. 64)

Einer dieser Sonderfälle ist das "ARX-Systemmodell" (siehe Abbildung 3.5). Dieses Modell beschreibt das zu untersuchende System durch die Störübertragungsfunktion

$$G_{\text{Stoer}}(z) = \frac{R_M(z)}{\varepsilon(z)} \tag{3.1}$$

$$= \frac{1}{A(z^{-1})} \tag{3.2}$$

$$= \frac{1}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} \tag{3.3}$$

und die Eingangsübertragungsfunktion

$$G_M(z) = \frac{\tilde{Y}_M(z)}{U(z)} \tag{3.4}$$

$$= \frac{B(z^{-1})}{A(z^{-1})} z^{-n_d} \tag{3.5}$$

$$= \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} \cdot z^{-n_d} \tag{3.6}$$

(Bohn und Unbehauen, 2016, S. 61-64) Der deterministische Teil des Systems wird also durch eine gebrochen-rationale Übertragungsfunktion der Ordnung n, sowie einer zusätzlichen Totzeit von n_d Abtastschritten beschrieben. Die Modellordnung n wird vorher nicht festgelegt, sondern während der Identifikation innerhalb eines bestimmten Bereichs ermittelt. Dieser Vorgang wird später beschrieben.

Zusätzlich zum deterministischen, vom Eingangssignal abhängigen Teil des Systemausgangs wird eine überlagerte stochastische Störgröße $R_M(z)$ angenommen. Im Modell besteht dieser Signalanteil aus einem band-begrenzten weißen Rauschen. Dieses wird durch die Filterung eines weißen Rauschens ε mit der Störübertragungsfunktion $G_{\mathrm{Stoer}}(z)$ beschrieben. Die Störübertragungsfunktion $G_{\mathrm{Stoer}}(z)$ stellt dabei einen Tiefpassfilter n-ter Ordnung mit denselben Polen und Zeitkonstanten wie die Eingangsübertragungsfunktion dar.

Die durch das Modell getroffenen Annahmen zur Beschaffenheit der Störgröße $R_M(z)$ sind später für die Rechtfertigung des Vorgehens zur Identifikation des Systems wichtig (siehe Abschnitt 3.3). Ein lineares System kann durch das in Abbildung 3.5 dargestellte Modell gut beschrieben werden. Wie in Kapitel 3.1 dargelegt, ist jedoch davon auszugehen, dass auch nichtlineare Systeme vorliegen. Die Identifikationsverfahren in Abschnitt 3.3 arbeiten jedoch mit linearen Systemmodellen. Es wird im Folgenden also ein lineares Modell ausgewählt, dass dem realen System im Arbeitspunkt möglichst genau angepasst wird.

3.3 Identifikation im Zeitbereich

Die Übertragungsfunktion des zur Identifizierung genutzten Modells lautet nach Gleichungen 3.4 und 3.6

$$G_M(z) = \frac{\tilde{Y}_M(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} \cdot z^{-n_d} \tag{3.7}$$

Durch Multiplizieren der obigen Gleichung mit den Zählern der Brüche ergibt sich folgende Gleichung:

$$\tilde{Y}_M(z) \cdot (1 + a_1 z^{-1} + \ldots + a_n z^{-n}) = U(z) \cdot (b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}) \cdot z^{-n_d} \tag{3.8}$$

Wird diese Gleichung in den Zeitbereich transformiert und ausmultipliziert, ergibt sich der Zusammenhang

$$\begin{aligned} \tilde{y}_M(k) + a_1 \tilde{y}_M(k-1) + \ldots + a_n \tilde{y}_M(k-n) = \\ b_0 u(k-0-n_d) + \ldots + b_n u(k-n-n_d) \end{aligned} \tag{3.9}$$

Der aktuelle Ausgangswert $\tilde{y}_M(k)$ ergibt sich damit zu:

$$\begin{aligned} \tilde{y}_M(k) = -a_1 \tilde{y}_M(k-1) - \ldots - a_n \tilde{y}_M(k-n) \\ + b_0 u(k-0-n_d) + \ldots + b_n u(k-n-n_d) \end{aligned} \tag{3.10}$$

Diese Darstellung entspricht dem deterministischen Teil des nach Abschnitt 3.2 gewählten Modells. Zur besseren Übersichtlichkeit können die Terme aus Gleichung 3.10 in Summendarstellung zusammengefasst werden.

$$\tilde{y}_M(k) = -\sum_{i=1}^{n} \left(a_i \tilde{y}_M(k-i) \right) + \sum_{i=0}^{n} \left(b_i u(k-n_d-i) \right) \tag{3.11}$$

Die benötigten Modellwerte $\tilde{y}_M(k-1) \ldots \tilde{y}_M(k-n)$ sind noch nicht bekannt. Zur Ermittlung der Parameter werden diese daher durch die gemessenen Werte der Regelgröße $y(k-1) \ldots y(k-n)$ ersetzt.

$$\tilde{y}_M(k) = -\sum_{i=1}^{n} \left(a_i y(k-i) \right) + \sum_{i=0}^{n} \left(b_i u(k-n_d-i) \right) \tag{3.12}$$

Damit ist der aktuelle Modell-Ausgangswert $\tilde{y}_M(k)$ von einer linearen Kombination der vorangegangenen gemessenen Ausgangswerte $y(k-1)\ldots y(k-n)$ und den Eingangswerten $u(k)\ldots u(k-n_d-n)$ abhängig.

Für die Identifikation wird nun der Parametervektor

$$\boldsymbol{p_M} = [a_1 \ldots a_n \quad b_0 \ldots b_n]^T \tag{3.13}$$

eingeführt (abgewandelt nach Bohn und Unbehauen (2016) S. 104; zusätzlicher Parameter b_0 wurde eingeführt). Aufgabe der Identifikation ist es nun, diesen Vektor so zu bestimmen, dass die Abweichung zwischen dem gemessenen Ausgangswert $y(k)$ und dem Modellausgang $\tilde{y}_M(k)$ möglichst gering ist. Die Abtastung des Eingangssignals $u(k)$ und Ausgangssignal $y(k)$ wird mit einem Index von $k = 0$ begonnen. Die Werte von $u(k)$ und $y(k)$ sind für Werte von $k < 0$ nicht definiert. Daraus folgt, dass sich Gleichung 3.12 erst vollständig aufstellen lässt, wenn $n_d + n$ Abtastwerte vorliegen. Dies liegt daran, dass für die zweite Summe die vergangenen Eingangssignale bis $u(k-n_d-n)$ benötigt werden.

Um das Ausgleichsproblem lösen zu können, sind mindestens so viele Gleichungen wie Parameter notwendig (Bohn und Unbehauen, 2016, S. 68). Der Parametervektor p_M hat $2n+1$ Elemente. Um diese Parameter eindeutig bestimmen zu können, sind also mindestens $2n+1$ Gleichungen der Art von Gleichung 3.12 erforderlich.

Da $n_d + n$ Abtastwerte notwendig sind, um die erste Gleichung aufzustellen, sind zur Bestimmung der $2n+1$ gesuchten Parameter also mindestens

$$\begin{aligned} N_{min} &= n_d + n + 2n + 1 \\ &= 3n + n_d + 1 \end{aligned} \tag{3.14}$$

Abtastwerte von Ein- und Ausgang notwendig. Um den Einfluss der stochastischen Störgröße $R_M(z)$ heraus zu filtern, ist es sinnvoll, deutlich mehr als N_{min} Messwerte aufzunehmen (Bohn und Unbehauen, 2016, S. 68), um dann eine lineare Regression ausführen zu können.

3.3.1 Direkte Lösung als lineares Ausgleichsproblem

Im Folgenden wird die Lösung des oben beschriebenen linearen Ausgleichsproblems frei nach (Bohn und Unbehauen, 2016, S. 67-75) dargestellt. Im Vergleich zur genannten Quelle wird hier ein leicht geänderter Parametervektor verwendet, der auch die Identifikation sprungfähiger Systeme ermöglicht. Dazu wird im Folgenden auch der Parameter b_0 geschätzt (in Bohn und Unbehauen (2016) S. 67-75 wurde dieser zu $b_0 = 0$ definiert). Außerdem wird hier eine zusätzliche zeitliche Verschiebung des Ausgangssignals um n_d Abtastschritte hinzugefügt.

Zur Lösung wird die „Methode der kleinsten Quadrate“ verwendet. Das Optimum der Parameterbestimmung wird also so definiert, dass durch die Wahl der Parameter die Summe der quadratischen Fehler zwischen dem Modellausgang $\tilde{y}_M(k)$ und dem gemessenen Ausgang $y(k)$ möglichst gering wird. Die einzelnen Fehler werden dabei durch den Residuenvektor

$$\boldsymbol{r} = \tilde{\boldsymbol{y}} - \boldsymbol{y} = \boldsymbol{M}\boldsymbol{p}_M - \boldsymbol{y} \tag{3.15}$$

bestimmt. In dieser Darstellung wird der Ausgangswert des deterministischen Systemmodells als eine Summe von Funktionen $f_1 \dots f_{2n+1}$ beschrieben, die jeweils multipliziert mit den Elementen des Parametervektors p_M den zugehörigen Modellwert $\tilde{y}_M(k)$ liefern. Die gemessenen Werte der Regelgröße fließen hier in den Vektor

$$\boldsymbol{y} = \begin{bmatrix} y(n+nd) \\ \vdots \\ y(N) \end{bmatrix} \tag{3.16}$$

ein. N gibt dabei die Anzahl der gemessenen Abtastwerte an. Die Datenmatrix $\boldsymbol{M}$ hat nun die folgende Form:

$$\boldsymbol{M} = \begin{bmatrix} f_1(t_1) & \dots & f_{2n+1}(t_1) \\ \vdots & & \vdots \\ f_1(t_m) & \dots & f_{2n+1}(t_m) \end{bmatrix} \tag{3.17}$$

(Ansorge und Oberle, 2000, S. 160)

Die Funktionen $f_1 \dots f_n$ stellen dabei Verschiebungen des negierten Ausgangssignals dar, die Funktionen $f_n \dots f_{2n+1}$ das aktuelle und Verschiebungen des Eingangssignals. Die Zeitpunkte $t_1 \dots t_m$ werden im Folgenden durch die in den Klammern angegebenen Abtastschritte repräsentiert.

$$M = \left[\begin{array}{ccc:ccc} -y(n+n_d) & \dots & -y(n_d) & u(n) & \dots & u(0) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -y(N-1) & \dots & -y(N-1-n) & u(N-n_d) & \dots & u(N-n_d-n) \end{array}\right] \tag{3.18}$$

Wie oben erwähnt, soll nun der Parametervektor $\boldsymbol{p}_M$ so gewählt werden, dass die Summe der quadratischen Residuen

$$SSE = \boldsymbol{r}^T \boldsymbol{r} \tag{3.19}$$

möglichst gering wird. Für diesen Fall gilt laut (Ansorge und Oberle, 2000, S. 162) die Bestimmungsgleichung

$$M^T M \boldsymbol{p}_M = M^T \boldsymbol{y} \tag{3.20}$$

Der offensichtliche Weg zur Auflösung nach dem Parametervektor $\boldsymbol{p}_M$ führt über die linksseitige Multiplikation der Matrix $[M^T M \boldsymbol{p}_M]^{-1}$. Allerdings ist dazu die Inversion der Matrix $[M^T M \boldsymbol{p}_M]$ notwendig. In (Bohn und Unbehauen, 2016, S. 69) wird die Bestimmung der Streckenparameter durch die Inversion einer ähnlichen Matrix gelöst. Es ergibt sich damit folgende Lösung der Identifikation

$$\boldsymbol{p}_M = [M^T M]^{-1} M^T - \boldsymbol{y} \tag{3.21}$$

In anderen Quellen wird ausdrücklich vor dieser Lösung gewarnt (Ansorge und Oberle, 2000, S. 162). Dazu wird angebracht, dass es für den Fall, dass das Gleichungssystem 3.20 schlecht konditioniert ist, zu Rundungsfehlern und numerischen Problemen bei der Identifikation führen kann. Als alternatives Verfahren wird dort eine QR-Zerlegung empfohlen und beschrieben. Um die Komplexität dieser Arbeit zu beschränken, wurde entschieden das lineare Ausgleichsproblem aus Gleichung 3.20 mit einem fertigen Algorithmus aus einer Softwarebibliothek zu lösen. Das genaue Vorgehen dazu ist in Abschnitt 5.4 beschrieben.

3.3.2 Bestimmung der Modellordnung und der Totzeit

Das oben beschriebene Verfahren zur Ermittlung einer Streckenübertragungsfunktion kann nur für gegebene Werte der Modellordnung n und der Totzeit n_d durchgeführt werden. Diese Parameter sind allerdings zunächst unbekannt. Um sie zu bestimmen, kann die Identifikation testweise mit unterschiedlichen Parametersätzen durchgeführt werden und anschließend aus diesen Tests ein passendes Systemmodell gewählt werden. In (Bohn und Unbehauen, 2016, Kapitel 4.2) sind dazu verschiedene Verfahren beschrieben.

Zur Bestimmung der Modellordnung n wird hier daraus der sogenannte „F-Test“ ausgewählt. Es handelt sich um einen statistischen Test, der zwei Systemmodelle mit einer unterschiedlichen Modellordnung n (und damit einer unterschiedlichen Parameterzahl) direkt miteinander vergleicht. Dazu wird die Nullhypothese aufgestellt, dass die zusätzlichen Parameter im Modell höherer Ordnung nicht signifikant sind. Die Ablehnung oder Annahme dieser Hypothese erfolgt bei einer frei wählbaren Irrtumswahrscheinlichkeit α nach einer einfach zu berechnenden Testgröße, die mit einem, durch eine F-Verteilung definierten, Schwellwert verglichen wird (Bohn und Unbehauen, 2016, S. 146-147). Da sich ein solches Verfahren gut algorithmisch auswerten lässt, wird es hier zur Strukturprüfung ausgewählt.

Die Testgröße lautet hierbei

$$f(\hat{n}_2, \hat{n}_1) = \frac{I_{v1} - I_{v2}}{I_{v2}} \cdot \frac{\tilde{N} - 2\hat{n}_2}{2(\hat{n}_2 - \hat{n}_1)} \tag{3.22}$$

Wobei $\hat{n}_2$, $\hat{n}_1$ die Modellordnungen der zu vergleichenden Systeme, $\tilde{N}$ die Anzahl der zur Modellbildung verwendeten Messwerte und I_{v1}, I_{v2} einem Gütefunktional der beiden Modelle entsprechen (Bohn und Unbehauen, 2016, S. 146).

Das Gütefunktional wird dabei durch die Hälfte der Summe der quadrierten Signalfehler $\varepsilon(k, n)$ definiert:

$$I_v(\hat{n}) = \frac{1}{2} \sum_{k=\hat{n}}^{N} \varepsilon^2(k, n) \tag{3.23}$$

Dabei soll der Signalfehler $\varepsilon(k, n)$ durch die Bildung der Differenz aus einem simulierten Ausgangssignalverlauf mit dem gemessenen Signalverlauf verglichen werden (Bohn und Unbehauen, 2016, S. 144/145).

Von diesem Verfahren wird hier leicht abgewichen, da die Implementierung einer „echten“ Ausgangssignalsimulation einen erheblichen Mehraufwand zur Folge hätte und die Komplexität dieser Arbeit deutlich erhöht hätte. Stattdessen wird zur Berechnung der Signalfehler Gleichung 3.15 herangezogen. Dort werden die Residuen der Modellbildung auf einfache Weise berechnet. Dazu werden lediglich die Datenmatrix $\boldsymbol{M}$ (Gleichung 3.18), der Parametervektor $\boldsymbol{p}_M$ (Gleichung 3.13) und die gemessenen Ausgangswerte $\boldsymbol{y}$ benötigt. Als Gütefunktional kann dann die Hälfte der Summe der quadratischen Fehler nach 3.19 genutzt werden:

$$I_v = \frac{1}{2} SSE = \frac{1}{2} \cdot \boldsymbol{r}^T \boldsymbol{r} \tag{3.24}$$

Der wesentliche Unterschied im Vergleich zu (Bohn und Unbehauen, 2016, S. 144/145) besteht nun darin, dass jeder Modellwert $\tilde{y}$ direkt von den vorherig gemessenen Signalwerten y und nicht den vorherigen Modellwerten $\tilde{y}$ abhängt. Die Abweichung vom Modell addiert sich also nicht über die einzelnen Abtastschritte. Es ist daher ein kleinerer Wert für I_v zu erwarten.

Da jedoch in Gleichung 3.22 im ersten Bruch das Verhältnis der Änderung von I_v zum Wert von I_{v2} gesetzt wird, wird davon ausgegangen, dass sich diese Abweichung schwach auf die Ergebnisse auswirkt. Wären z.B. beide Größen I_{v1}, I_{v2} um den gleichen Faktor kleiner, ließe sich dieser ausklammern, und die Testgröße $f(\hat{n}_2, \hat{n}_1)$ aus Gleichung 3.22 würde exakt den gleichen Wert wie ohne diesen Faktor annehmen.

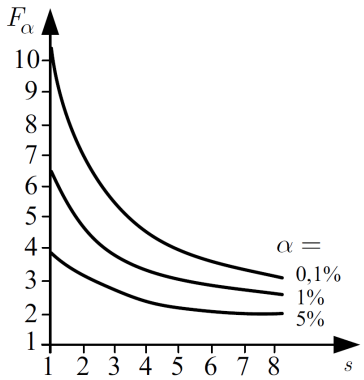

Abbildung 3.6: „Darstellung des Schwellwertes $F\alpha = F(s, \infty, \alpha)$ über s für verschiedene Irrtumswahrscheinlichkeiten α“ (Bohn und Unbehauen, 2016, Bild 4.1)

Ist nun die Testgröße $f(\hat{n}_2, \hat{n}_1)$ bestimmt, wird sie mit dem Schwellwert $F_{2\Delta n, \tilde{N}-2\hat{n}, \alpha}$ verglichen. Diese entspricht dem Wert, „der von einer $F(2\Delta n, \tilde{N} - 2\hat{n}_2)$-verteilten Zufalls-

größe gerade mit der Wahrscheinlichkeit α überschritten wird“ (Bohn und Unbehauen, 2016, S. 147). Beispiele für diese inverse F-Verteilung sind in Abbildung 3.6 dargestellt. Solange der Wert der Testgröße $f(\hat{n}_2, \hat{n}_1)$ größer als der Schwellwert $F_{2\Delta n, \tilde{N}-2\hat{n}, \alpha}$ ist, wird die eingangs erwähnte Nullhypothese verworfen. Damit sind die neuen Modellparameter signifikant (Bohn und Unbehauen, 2016, S. 147) und die Identifikation wird fortgesetzt.

Eine Bestätigung der Nullhypothese durch den obigen Test wird hier als Abbruchkriterium für die Identifikation genutzt. Liefert also das zweite System keine neuen signifikanten Parameter, so ist das vorherige System mit weniger, dafür signifikanten Parametern zu wählen. Mit diesem Vorgehen wird für jede Totzeit $n_d = 0 \ldots n_{d,max}$ das beste Systemmodell ermittelt. Die maximal getestete Totzeit $n_{d,max}$ wird hier auf einen Wert von 20% der Abtastwerte festgelegt. Diese Festlegung ist willkürlich, es scheint jedoch sinnvoll, dass eine Messung an der Regelstrecke mindestens dem Fünffachen der Totzeit entspricht.

Anschließend werden in einer Schleife die besten Systeme für jede Totzeit miteinander verglichen, um den besten Wert für die Totzeit n_d zu ermitteln. Auch hier wird bei ungleicher Systemordnung wieder der oben beschriebene F-Test angewendet, um das bessere System auszuwählen. Bei gleicher Systemordnung kann einfach das System mit dem geringeren Wert des Gütefunktionals I_v als das bessere ausgewählt werden.

3.4 Wahl der Messdauer und Abtastzeit

Die eingesetzten Automationscontroller und Algorithmen arbeiten zeit-diskret. Es liegen also nur diskret abgetastete Signale vor. Bei einem Signal der Länge N besteht das zugehörige diskrete Frequenzspektrum aus N äquidistanten Frequenzstellen

$$f = 0, \frac{f_s}{N} \ldots (N-1) \cdot \frac{f_s}{N} \tag{3.25}$$

(von Grünigen, 2014, S. 182). Dabei steht f_s für die Abtastfrequenz, also die reziproke Abtastzeit T_s. Bei der Frequenzstelle $f = 0$ handelt es sich um den Gleichanteil des Signals. Das untersuchte Spektrum (für Frequenzen $\neq 0$) reicht also von der kleinsten Frequenz

$$f_{\text{min}} = \frac{1}{T_s N} \tag{3.26}$$

bis zur maximalen Frequenz

$$f_{\text{max}} = (N-1) \cdot \frac{1}{T_s N} \tag{3.27}$$

und lässt sich über die Abtastzeit T_s und die Anzahl der Abtastungen N einstellen.

Wie unter Abschnitt 2.2 erwähnt, bietet es sich als Faustformel an, die Abtastfrequenz mindestens auf das Zehnfache der Systembandbreite zu legen. Außerdem wird die Totzeit n_d bei der obigen Systemidentifikation stets in ganzen Abtastschritten ermittelt. Um diese möglichst genau zu ermitteln, bietet sich eine hohe Abtastrate an. Gleichzeitig darf die Abtastzeit nicht so groß gewählt werden, dass sich die abgetasteten Messwerte stark ähneln, da das lineare Ausgleichsproblem der Systemidentifikation dann unter Umständen nicht mehr lösbar ist (Bohn und Unbehauen, 2016, S. 503).

Die Messdauer sollte zumindest so groß gewählt werden, dass das zu untersuchende System im unteren Frequenzbereich (kleiner der Systembandbreite) gut angeregt wird. Generell führt eine längere Messdauer auch zu einer größeren Datenmatrix $\boldsymbol{M}$ und somit auch zu einer besseren Mittlung der auftretenden Störungen. Damit steigt die Güte der Identifikation (Bohn und Unbehauen, 2016, S. 508).

Die Auswahl geeigneter Parameter für die Abtastzeit T_s und die Messdauer $T_{\text{mess}} = N * T_s$ muss je nach Strecke angepasst werden.

4 Reglersynthese

Für die durch das Systemmodell aus Kapitel 3 spezifizierte Regelstrecke soll nun ein Regler dimensioniert werden. Die Basis dazu bildet die folgende Übertragungsfunktion im Bildbereich

$$G_s(s) = e^{-T_t s} \cdot \frac{b_{cn} s^n \ldots b_{c1} s + b_0}{s^n \ldots a_{c1} s + a_{c0}} \tag{4.1}$$

(Der zum Erhalten dieser Übertragungsfunktion notwendige Übergang vom z- in den s-Bereich wird bei der Implementierung in Abschnitt 5.5.1 beschrieben. Hier soll die obige Übertragungsfunktion zunächst als gegeben angenommen werden.) Dabei handelt es sich um ein Systemmodell n-ter Ordnung mit einer Totzeit T_t. Auf den zur Regelung der Strecken verwendeten Automationscontrollern sind sogenannte „PID-Regler“ als Softwarebausteine vorhanden. Ein reiner PID-Regler ist technisch nicht realisierbar. Das folgt aus der Realisierbarkeitsbedingung, dass der Zählergrad nicht größer als der Nennergrad der Reglerübertragungsfunktion sein darf (Unbehauen, 2007, S. 251). Es wird jedoch im Folgenden davon ausgegangen, dass der Regler einem idealen, stetigen PID-Regler entspricht. In der Realität handelt es sich jedoch um einen zeit-diskreten PID-T_{1R}-Regler.

Des Weiteren hat sich der Autor entschlossen den D-Anteil des Reglers zu deaktivieren. Dazu wird im Regler die Vorhaltezeit $T_v = 0sek$ eingestellt. Damit wird auf die Möglichkeit verzichtet, einen zur Änderung der Regelgröße $y(t)$ proportionalen Anteil zur Stellgröße $u(t)$ zu addieren. Dieser Anteil ist vor allem zur schnellen Ausregelung von Störungen vorteilhaft. So wird bei einer Änderung der Regelgröße $y(t)$ sofort reagiert, bevor sich größere Abweichungen ergeben.

In der Gebäudetechnik ist dies oft nicht notwendig oder sogar unerwünscht. Das lässt sich an folgendem Beispiel veranschaulichen:

An einem kalten Tag wird bei einer Raumtemperaturregelung ein Fenster geöffnet. Infolgedessen sinkt die Raumtemperatur schnell ab. Bei vorhandenem D-Anteil würde sofort eine große Stellenergie aufgewendet werden, um diese Störung auszugleichen. Dieses Vorgehen führt möglicherweise zu einer schlechteren Energieeffizienz.

Bei einigen Anwendungen kann es sinnvoll sein, mit einem D-Anteil zu arbeiten. Allerdings soll das in diesem Bericht beschriebene Programm lediglich in der Lage sein für „gutmütige“ Strecken automatisiert Regler auszulegen. D.h., es wird, wie in Abschnitt 2.2 beschrieben, PT_n-Verhalten mit geringen Totzeiten und einer dominierenden Zeitkonstante angenommen. Bei komplexeren Strecken ist es dann immer noch möglich, die erhaltenen Identifikationsergebnisse zu nutzen, um dann eine manuelle Reglerdimensionierung durchzuführen.

Aus den oben genannten Gründen wird hier lediglich die Dimensionierung eines PI-Reglers mit der Übertragungsfunktion

$$G_R(s) = \frac{K_{PR}(1 + T_N s)}{T_N s} \tag{4.2}$$

diskutiert.

4.1 Anforderungen an das Syntheseergebnis

Um die Synthese eines Reglers durchzuführen, ist es zunächst sinnvoll die Anforderungen an das Syntheseergebnis zu definieren. Dies geschieht in diesem Abschnitt.

Stationäre Genauigkeit

Eine wichtige Forderung ist zunächst, dass der Regelkreis stationär genau werden soll. D.h. für die bleibende Regelabweichung

$$e_\infty = \lim_{t\to\infty} e(t) = \lim_{t\to\infty} (w(t) - y(t)) \tag{4.3}$$

wird eine Konvergenz zu $e_\infty = 0$ gefordert. Das bedeutet, dass die Regelgröße $y(t)$ nach einer unbestimmten Zeit den Wert der Führungsgröße $w(t)$ annehmen soll. Dies ist für sprungförmige Eingangsgrößen dann der Fall, wenn der offene Regelkreis

$$G_0(s) = G_R(s) \cdot G_S(s) \tag{4.4}$$

mindestens einfaches I-Verhalten aufweist. Es ist dabei unerheblich, ob es sich dabei um einen Führungs- oder Störgrößensprung der Hauptstörgröße handelt. Die Hauptstörgröße entspricht dabei der Summe der auftretenden Störungen umgerechnet auf den Systemausgang. In beiden Fällen gibt es keine bleibende Regelabweichung (Unbehauen,

2007, S. 123). Durch den Einsatz eines PI-Reglers ist ein I-Verhalten des Regelkreises sichergestellt und die Forderung nach stationärer Genauigkeit kann erfüllt werden.

Dynamisches Verhalten

Das dynamische Verhalten des Regelkreises beschreibt den zeitlichen Verlauf der Regelgröße. Im Folgenden wird zunächst davon ausgegangen, dass die Führungsübertragungsfunktion $G_W(s)$ nach der Reglersynthese ein dominierendes Polpaar besitzt. Das bedeutet, dass ein Polpaar maßgeblich das Verhalten der Regelstrecke bestimmt, während mögliche weitere Pole der Übertragungsfunktion so weit links in der s-Ebene liegen („so schnell sind“), dass sie keinen großen Einfluss auf das Zeitverhalten haben (vgl. Abbildung 4.1).

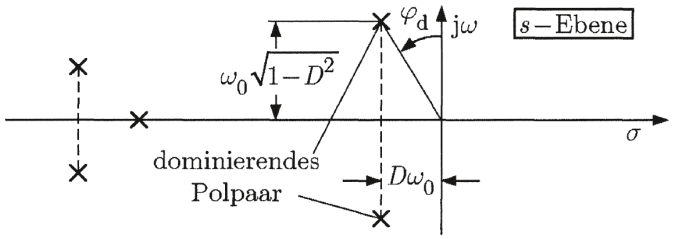

Abbildung 4.1: „Polstellenverteilung eines Übertragungsgliedes mit dominierendem Polpaar“ (Unbehauen, 2007, Bild 8.3.2.)

Das Verhalten der Regelstrecke kann dann in Näherung durch ein PT_2-Glied mit der Übertragungsfunktion

$$G_w(s) = \frac{\omega_0^2}{s^2 + 2D\omega_0 s + \omega_0^2} \tag{4.5}$$

beschrieben werden, wobei D den Dämpfungsgrad des Systems angibt (Unbehauen, 2007, S. 210).

Der Grad der Dämpfung kann später durch die Wahl der Reglerparameter beeinflusst werden. In Abbildung 4.2 sind zur Veranschaulichung die Sprungantworten für verschiedene Dämpfungsgrade D aufgezeichnet. Die Zeitachse ist dabei auf die Zeitkonstante $T = 1/\omega_0$ normiert, wobei ω_0 die *Eigenfrequenz* des Systems angibt. Die Amplitude y wird auf einen Endwert von $y = 1$ normiert.

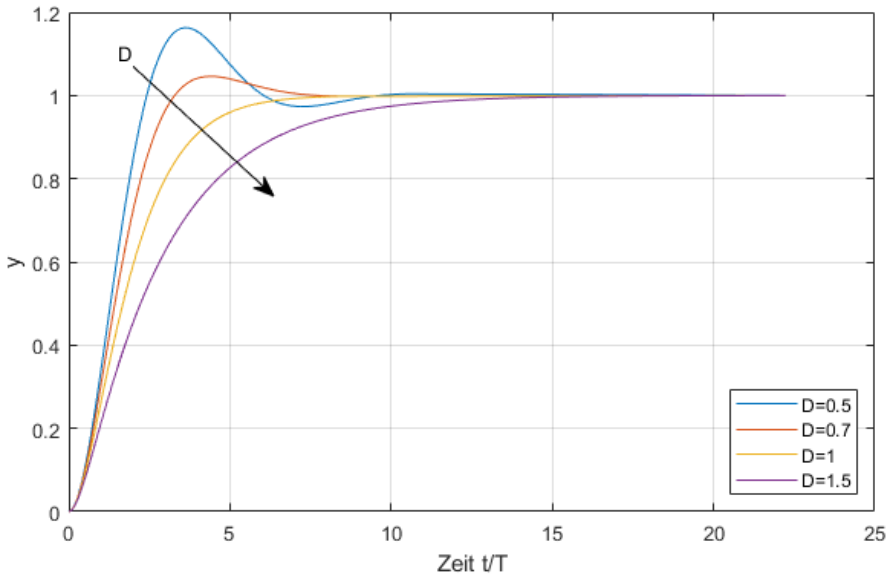

Abbildung 4.2: Sprungantwort eines Systems für die Dämpfungsgrade $D = 0{,}5; 0{,}7; 1; 1{,}5$

Zusätzlich werden für die gezeigten Sprungantworten in MATLAB® numerisch einige Kenngrößen ermittelt. Dazu wird die MATLAB® Funktion „stepinfo“ auf entsprechende Übertragungsfunktionen nach Gleichung 4.5 angewendet. Die dabei ermittelten Kenngrößen lauten:

Maximales Überschwingen $e_{max}(D)$
: Gibt an, inwieweit der stationäre Endwert überschritten wird.

Ausregelzeit für ein Toleranzband von 2% $t_{2\%}$
: Gibt die benötigte Zeit an, bis die Regelgröße dauerhaft in einem Toleranzband von $\pm 2\%$ (MATLAB® -Standardwert) des Endwertes verbleibt.

Ausregelzeit für ein Toleranzband von 5% $t_{5\%}$
: Gibt die benötigte Zeit an, bis die Regelgröße dauerhaft in einem Toleranzband von $\pm 5\%$ des Endwertes verbleibt.

Anstiegszeit $T_{A,50}$ (nach (Unbehauen, 2007, S. 212))
: Gibt die benötigte Zeit an, bis die Regelgröße erstmals 50% des Endwertes erreicht.

Die dazu ermittelten Zahlenwerte sind in Tabelle 4.1 dargestellt. Es lässt sich dazu feststellen, dass eine kleine Dämpfung zunächst eine kürzere Anstiegszeit $T_{A,50}$ zur Folge

hat. Des Weiteren sinkt das Überschwingen bei wachsender Dämpfung. Ab einer Dämpfung von $D = 1$ findet kein Überschwingen mehr statt, der Einschwingvorgang wird lediglich zusätzlich zeitlich verzögert. Die Ausregelzeit hängt stark vom definierten Toleranzband ab. So kann etwa ein Überschwingen an die Grenze des Toleranzbandes eine sehr kurze Ausregelzeit zur Folge haben (siehe Tabelle 4.1).

Dämpfung D	0,5	0,7	1,0	1,5
e_{max} [%]	16,30	4,60	0	0
$t_{2\%}$	8,08	5,98	5,83	10,7
$t_{5\%}$	5,29	2,90	4,74	8,26
$T_{A,50}$	1,29	1,43	1,68	2,23

Tabelle 4.1: Kenngrößen der Sprungantwort für verschiedene Dämpfungen D (zeitliche Größen normiert auf T)

Welche Dämpfung und welches Toleranzband für einen spezifischen Regelkreis wünschenswert sind, hängt nun vom jeweiligen Anwendungsfall ab. Wird etwa eine Raumtemperaturregelung mit Sollwertsprüngen von $\Delta w = 2K$ betrachtet, so würde ein Toleranzband von 5% hier $0{,}1K$ entsprechen. Das ist ein Wert, der für Anwendungen normaler Güte praxistauglich erscheint. Nach Tabelle 4.1 ist in diesem Fall etwa eine Dämpfung von $D = 0{,}7$ wünschenswert, da kein Überschwingen außerhalb des Toleranzbandes stattfindet und eine geringe Ausregelzeit $t_{5\%}$ benötigt wird. Bei anderen Anwendungen kann eine geringere Toleranz erlaubt sein.

Besonders bei sogenannten Sequenz-Regelungen sollte auf eine große Dämpfung geachtet werden. Ein Beispiel für eine derartige Regelstrecke ist eine Lüftungsanlage mit einem Heiz- und einem Kühlregister. Bei einem positiven Sollwertsprung kann es bei diesen Strecken z.B. dazu kommen, dass zunächst mit dem Heizregister geheizt wird, dabei ein Überschwingen außerhalb des Toleranzbandes auftritt, und mit Kühlregister dann gegen-gekühlt wird, bis sich die Regelgröße auf den neuen Sollwert einpendelt. Ein solches Verhalten wäre sicherlich aus energetischen Gründen nicht wünschenswert.

Ein weiterer Aspekt bei der Auswahl der Dämpfung ist die Zeitvarianz der Streckenparameter. Die Parameter der Strecke werden nur für den Zeitpunkt der Identifikation bestimmt. Es gibt in der Gebäudeautomation aber eine ganze Reihe an Regelkreisen, deren Parameter zeitabhängig sind. So wird eine Heizungsanlage ein stark witterungsabhängiges Verhalten aufweisen. Wird etwa die Heizleistung eines Heizungsmischkreises in den angeschlossenen Räumen durch Thermostatköpfe an den Heizkörpern begrenzt, so wird der Durchfluss des Heizmediums mit steigender Außentemperatur stark absin-

ken. Dadurch verändern sich möglicherweise die Zeitkonstanten der Regelstrecke und auch die Streckenverstärkung.

Bei einer solchen Strecke ist eine gewisse Kenntnis, abseits des aktuellen Streckenmodells, über die Regelstrecke erforderlich, um eine geeignete Dämpfung zu wählen. Wäre es z.B. eine Erkenntnis, dass ein Heizkreis im Winter eine geringere Schwingneigung als im Sommer aufweist, könnte dies bei der Durchführung einer Reglersynthese im Winter bereits berücksichtigt werden. Es könnte dann eine größere Dämpfung gewählt werden, um eine größere Robustheit des Regelkreises im Sommer zu schaffen.

Der für solche Überlegungen notwendige Erfahrungsschatz kann mit dieser Arbeit nicht abgedeckt werden. Daher sollen im Folgenden jeweils Reglerparameter für verschiedene Dämpfungsgrade vorgegeben werden. Es obliegt dann dem Nutzer zu entscheiden, welcher Dämpfungsgrad für die entsprechende Anwendung geeignet ist.

4.2 Stabilitätsprüfung der Regelstrecke

Eine weitere Einschränkung der Synthese soll sein, dass lediglich BIBO („bounded input, bounded output“)-stabile Regelstrecken (siehe Frey und Bossert (2008) S. 6) betrachtet werden. Diese zeichnen sich dadurch aus, dass sie auf ein begrenztes Eingangssignal mit einem begrenzten Ausgangssignal antworten. Es ist davon auszugehen, dass diese Eigenschaft bei den Regelstrecken des vorgesehenen Einsatzgebietes vorhanden ist. Allerdings kann es durch eine fehlerhafte Identifikation des Systems zu einem instabilen Systemmodell kommen. Unabhängig davon, dass das identifizierte Modell in diesem Fall keine gültigen Aussagen über das zu identifizierende Modell zulässt, ist das im Folgenden aufgezeigte Syntheseverfahren nicht für instabile Übertragungsfunktionen zulässig. Das liegt daran, dass die Identifikation schlussendlich im Frequenzbereich durchgeführt wird und dabei ein Übergang vom Bild- in den Zeitbereich notwendig ist, was eine Stabilität des Regelkreises voraussetzt (Frey und Bossert, 2008, S. 175). Aus oben genannten Gründen ist es daher notwendig, die Stabilität des Streckenmodells zu prüfen. Fällt die Prüfung negativ aus, wird der Syntheseprozess an dieser Stelle mit einer Fehlermeldung abgebrochen.

Da die Streckenübertragungsfunktion $G_s(s)$ (Gleichung 4.1) im Laplace-Bildbereich zu diesem Zeitpunkt bereits bekannt ist, bietet es sich an, ein algebraisches Stabilitätskriterium zu verwenden. Die Übertragungsfunktion wird dazu gedanklich in zwei Teile aufgetrennt: In den gebrochen rationalen Teil und die Totzeit (entsprechend dem Term $e^{-T_t s}$ aus Gleichung 4.1). Dieser gedankliche Aufbau ist dabei in Abbildung 4.3 dargestellt.

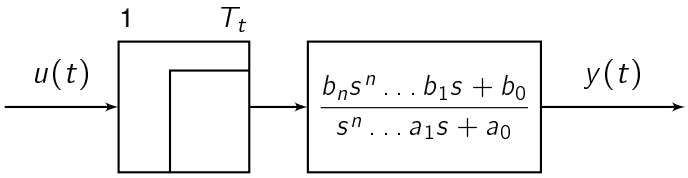

Abbildung 4.3: Aufteilung der Übertragungsfunktion des offenen Regelkreises $G_s(s)$

Ein Eingangssignal $u(t)$ wird dabei verschoben, um eine Totzeit an den gebrochen-rationalen Anteil der Übertragungsfunktion weiterzugeben. Diese Darstellung ist zulässig, da es sich um ein lineares System handelt. Die Übertragungsglieder lassen sich bei einem solchen System aufteilen, und da für die betrachteten linearen, zeit-invarianten Systemmodelle Kommutativität gilt (Frey und Bossert, 2008, S. 44), darf die Reihenfolge der Übertragungsglieder getauscht werden. Anhand dieser Vorstellung lässt sich sich leicht überlegen, dass es nun zur Prüfung der Stabilität des Regelstreckenmodells ausreicht, die Stabilität des gebrochen-rationalen Anteils in der Übertragungsfunktion zu prüfen. Dabei wird nachgewiesen, dass dieser Anteil des Übertragungsverhaltens für jedes endliche Eingangssignal ein endliches Ausgangssignal liefert. Es ist dann offensichtlich, dass das Ausgangssignal auch für um eine Totzeit T_t verschobene Signale endliche Ausgangssignale liefern wird. Das Ausgangssignal wird sich dabei lediglich zu einem späteren Zeitpunkt einstellen, aber die gleichen Werte annehmen.

Die Regelstrecke ist nun genau dann asymptotisch stabil, wenn alle Nullstellen des Nennerpolynoms („Pole der Übertragungsfunktion“)

$$A(s) = s^n \ldots a_1 s + a_0 \tag{4.6}$$

in der linken s-Halbebene liegen. Dies ist der Fall, wenn für den Realteil jedes Pols $\boldsymbol{\alpha}_i$ die Bedingung

$$\mathrm{Re}(\boldsymbol{\alpha}_i) < 0 \tag{4.7}$$

erfüllt ist (Unbehauen, 2007, S. 140/141).
Zum Prüfen der Stabilität werden im erstellten Programm die Pole numerisch ermittelt, und die Bedingung aus Gleichung 4.7 wird direkt geprüft. Die Bestimmung der Polstellen ist zur folgenden Polkompensation ohnehin notwendig (siehe Abschnitt 4.3). Diese Art der Stabilitätsprüfung erfordert dann also nur noch einen geringen Aufwand. Der Sonderfall von grenzstabilen Systemen soll im Folgenden nicht betrachtet werden. Dieser Sonderfall zeichnet sich dadurch aus, dass auch einfache Pole auf der Imaginärachse vorhanden sein können. Es gilt also für mindestens einen Pol

$$\mathrm{Re}(\boldsymbol{\alpha}_i) = 0 \tag{4.8}$$

In diesem Fall könnte das System zum Beispiel eigenerregt schwingen oder integrales Verhalten für $t \to \infty$ aufweisen. Diesen Fall zu identifizieren wäre mit größerem Aufwand verbunden, da die numerische Ermittlung der Pole dabei einen Wert genau gleich Null liefern müsste. Das ist nahezu ausgeschlossen. Es wäre also notwendig einen Schwellwert festzulegen, ab dem der Realteil zu Null angenommen wird. Dieser Aufwand steht nicht im Verhältnis zum erwarteten Nutzen, da wie in Abschnitt 2.2 dargelegt, nicht mit Strecken gerechnet wird, die ein solches Verhalten aufweisen.

4.3 Bestimmung der Nachstellzeit T_N durch Polkompensation

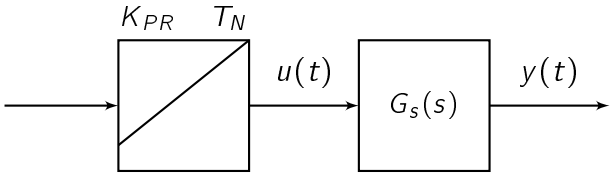

Abbildung 4.4: Signalflussdiagramm des offenen Regelkreises

Im Allgemeinen ergibt sich für den offenen Regelkreis nun die in Abbildung 4.4 dargestellte Anordnung aus PI-Regler und Strecke. Im Folgenden soll die Übertragungsfunktion $G_s(s)$ (Gleichung 4.1) in Pol-Nullstellen-Darstellung betrachtet werden. Dazu werden das Zähler- und Nennerpolynom aus Gleichung 4.1 jeweils als Produkt ihrer Wurzeln dargestellt. Zusammen mit der Reglerübertragungsfunktion (Gleichung 4.2) ergibt sich dann die folgende Übertragungsfunktion für den offenen Regelkreis:

$$\begin{aligned} G_0(s) &= G_r(s) \cdot G_s(s) \\ &= \frac{K_{PR}(1+T_N s)}{T_N s} \cdot e^{-T_t s} \cdot \frac{b_{cn}s^n \ldots b_{c1}s + b_{c0}}{s^n \ldots a_{c1}s + a_{c0}} \\ &= \frac{K_{PR}(1+T_N s)}{T_N s} \cdot e^{-T_t s} \cdot \frac{(s-\boldsymbol{\beta}_1)\cdot(s-\boldsymbol{\beta}_2)\ldots(s-\boldsymbol{\beta}_n)}{(s-\boldsymbol{\alpha}_1)\cdot(s-\boldsymbol{\alpha}_2)\ldots(s-\boldsymbol{\alpha}_n)} \end{aligned} \tag{4.9}$$

Dabei stehen $\boldsymbol{\alpha}_1 \ldots \boldsymbol{\alpha}_n$ für die Pol-, $\boldsymbol{\beta}_1 \ldots \boldsymbol{\beta}_n$ für die Nullstellen der Übertragungsfunktion. Um die Zahl der Pole der Führungsübertragungsfunktion möglichst gering zu halten, wird in Anlehnung an (Unbehauen, 2007, Kapitel 8.3.4: „Reglerentwurf mit dem Wurzelortskurvenverfahren“) zunächst eine Polkompensation durchgeführt. Das heißt, der dominierende Pol der Streckenübertragungsfunktion $G_s(s)$ wird durch das Einstellen einer geeigneten Nullstelle der Reglerübertragungsfunktion $G_R(s)$ kompensiert. Wie in Abschnitt 2.2 beschrieben, wird davon ausgegangen, dass im Wesentlichen PT_n-Verhalten der Strecken vorliegt. Der dominierende Pol wird dabei von der größten Streckenzeitkonstante bestimmt. Da der Betrag des Pols der reziproken zugehörigen Zeitkonstante entspricht, ist der dominierende Pol derjenige Pol, der den kleinsten Betrag aufweist.

Es wird nun zur Vereinfachung festgelegt, dass die Pole $\boldsymbol{\alpha}_1 \ldots \boldsymbol{\alpha}_n$, nach Betrag sortiert, in aufsteigender Reihenfolge indexiert sind. Für die Nachstellzeit T_N gilt die Einschränkung, dass hier lediglich ein reeller Zahlenwert eingestellt werden kann. Der dominierende Pol $\boldsymbol{\alpha}_1$ kann jedoch auch einen komplexen Wert annehmen (etwa bei einer schwingenden PT_2-Strecke). In diesem Fall ist keine vollständige Kompensation des Pols durch die Nachtstellzeit möglich. Allerdings wird entschieden auch hier die zusätzliche Nullstelle des Reglers auf den Kehrwert der Eigenfrequenz des dominierenden Polpaars zu legen. Der Betrag der Eigenfrequenz dieses Polpaars ist nach (Unbehauen, 2007, Bild 8.3.26) durch den Betrag der entsprechenden Pole gekennzeichnet. Daraus ergibt sich dann als Einstellvorschrift für die Nachstellzeit des Reglers

$$T_N = \frac{1}{|\boldsymbol{\alpha_1}|} \tag{4.10}$$

Bei den verwendeten Strecken wird allerdings angenommen, dass ein nahezu reeller Pol vorliegt. Ist der Pol rein reell ($\boldsymbol{\alpha}_1 = \mathrm{Re}(\boldsymbol{\alpha}_1) = \alpha_1$), vereinfacht sich die Einstellregel 4.10 zu

$$T_N = -\frac{1}{\alpha_1} \tag{4.11}$$

Unter der Bedingung eines reellen, dominierenden Pols vereinfacht sich damit die Übertragungsfunktion des offenen Regelkreises (Gleichung 4.9) durch Einsetzen der Einstellregel aus Gleichung 4.11 zu

$$
\begin{aligned}
G_0(s) &= \frac{K_{PR}(1-\frac{1}{\alpha_1}s)}{-\frac{1}{\alpha_1}s} \cdot e^{-T_t s} \cdot \frac{(s-\beta_1)\cdot(s-\beta_2)\ldots(s-\beta_n)}{(s-\alpha_1)\cdot(s-\alpha_2)\ldots(s-\alpha_n)} \\
&= \frac{K_{PR}\cdot\left(-\frac{1}{\alpha_1}\right)\cancel{(s-\alpha_1)}}{\cancel{-\frac{1}{\alpha_1}s}} \cdot e^{-T_t s} \cdot \frac{(s-\beta_1)\cdot(s-\beta_2)\ldots(s-\beta_n)}{\cancel{(s-\alpha_1)}\cdot(s-\alpha_2)\ldots(s-\alpha_n)} \\
&= \frac{K_{PR}}{s} \cdot \frac{(s-\beta_1)\cdot(s-\beta_2)\ldots(s-\beta_n)}{(s-\alpha_2)\ldots(s-\alpha_n)} \cdot e^{-T_t s} \qquad (4.12)
\end{aligned}
$$

Damit ist die dominierende Zeitkonstante eliminiert, was das dynamische Verhalten des offenen Regelkreises deutlich verbessert.

4.4 Auswahl und Einstellen eines geeigneten Phasenrands Ψ_R

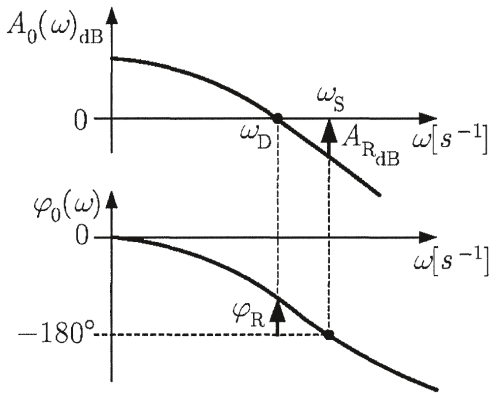

Abbildung 4.5: Phasen- und Amplitudenrand im Bode-Diagramm (Unbehauen, 2007, Bild 6.4.14 (b))

In Abschnitt 4.1 wurde eine einstellbare Dämpfung des zu regelnden Systems gefordert. In Abbildung 4.1 ist ein dominierendes Polpaar in der s-Ebene dargestellt.

Im Diagramm ist für den Realteil der Pole die Beziehung

$$\mathrm{Re}(\boldsymbol{\alpha}_i) = D \cdot \omega_0 \tag{4.13}$$

eingetragen. Da die Imaginärachse die Stabilitätsgrenze darstellt, steigt anschaulich mit steigender Dämpfung auch der „Abstand“ zu dieser. Diese graphisch im Diagramm aufgezeigte Beziehung gilt zunächst nur für Dämpfungen $D \leq 1$ (Unbehauen, 2007, S. 102), die folgenden Formeln gelten jedoch allgemein. Ein Maß, um diesen „Abstand“ zu messen, stellt nun der sogenannte Phasenrand Ψ_R (je nach Quelle auch „Phasenreserve“ genannt) dar. Dieser ist graphisch in Abbildung 4.5 (dort mit dem Formelzeichen φ_R) dargestellt. Dieser Parameter gibt den verbleibenden Phasenwinkel des offenen Regelkreises φ_0 zu einem Wert von $-180°$ bei der Durchtrittsfrequenz w_D an. Die Durchtrittsfrequenz beschreibt dabei die Frequenz, bei der der Betrag der Übertragungsfunktion des offenen Regelkreises (im Diagramm als $A_0(\omega)$ gekennzeichnet) kleiner als $0dB$ wird. Bei Phasenreserven von $\Psi_R < 0$ wird der geschlossene Regelkreis instabil, da das sogenannte Nyquist-Kriterium in diesem Fall nicht mehr erfüllt ist (Unbehauen, 2007, S. 166).

Für Systeme, deren offener Regelkreis durch eine Übertragungsfunktion der Form

$$G_0(s) = \frac{G_w(s)}{1 - G_w(s)} \quad \text{mit} \quad G_w(s) = \frac{\omega_0^2}{s^2 + 2D\omega_0 s + \omega_0^2} \tag{4.14}$$

$$G_0(s) = \frac{\omega_0^2}{s(s + 2D\omega_0)} \tag{4.15}$$

beschrieben werden kann, lässt sich die Phasenreserve Ψ_R nun über die Formel

$$\Psi_R = \arctan\left(2D\frac{\omega_0}{\omega_D}\right) \tag{4.16}$$

beschreiben. Des Weiteren gilt für diese Systeme der Zusammenhang

$$\frac{\omega_D}{\omega_0} = \sqrt{\sqrt{4D^4 + 1} - 2D^2} \tag{4.17}$$

(Unbehauen, 2007, S. 217). Durch Einsetzen von Gleichung 4.17 in Gleichung 4.16 ergibt sich dann für den Zusammenhang zwischen Phasenreserve und Dämpfung

$$\Psi_R = \arctan\left(\frac{2D}{\sqrt{\sqrt{4D^4 + 1} - 2D^2}}\right) \tag{4.18}$$

Dieser Zusammenhang ist grafisch in Abbildung 4.6 dargestellt und soll im folgenden Abschnitt zur Reglerdimensionierung genutzt werden.

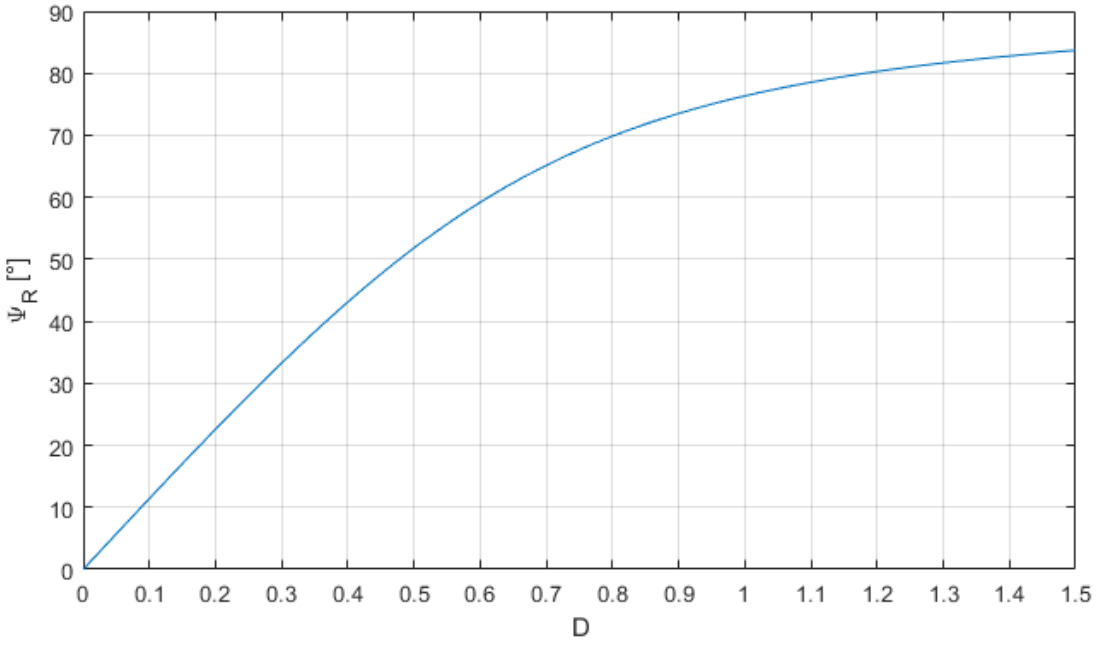

Abbildung 4.6: Phasenreserve Ψ_R des offenen Regelkreises eines PT_2-Glieds in Abhängigkeit der Dämpfung D

Bestimmung des Betrags der Reglerverstärkung K_{PR}

Nachdem die Nachstellzeit T_N des Reglers im Abschnitt 4.3 bereits festgelegt wurde, soll nun der zweite Reglerparameter, die Reglerverstärkung K_{PR}, bestimmt werden.

Die zu erwartenden Strecken werden in der Regel eine Totzeit $T_t > 0$ aufweisen. Bei einem Totzeitglied ist die Phase

$$\varphi_{T_t} = -T_t \omega \tag{4.19}$$

(Otto, 1989) direkt proportional zur Kreisfrequenz ω. Dies hat eine schnelle Drehung der Phase des Frequenzgangs des offenen Regelkreises zur Folge. Das bedeutet, dass der geschlossene Regelkreis deutlich schneller instabil werden kann, als der gebrochenrationale Teil der Übertragungsfunktion des offenen Regelkreises $G_0(s)$ (Gleichung

4.12) vermuten ließe. Es wird daher entschieden, die Auslegung hier im Frequenzbereich über eine wählbare Phasenreserve durchzuführen. Dieses Vorgehen hat den Vorteil, dass bei einer Phasenreserve von $\Psi > 0$ für beliebige Übertragungsfunktionen des Regelkreises sichergestellt werden kann, dass das Nyquist-Kriterium erfüllt ist (siehe vorheriger Abschnitt). Das heißt auch, falls die Näherung durch ein PT_2-Glied unzulässig ist, wird zumindest ein stabiles Verhalten des geschlossenen Regelkreises erreicht. Das Einschwingverhalten wird in der Regel jedoch mehr oder weniger stark von der Vorgabe mittels Dämpfung abweichen. Dies ist der Tatsache geschuldet, dass die Dämpfung in der hier dargestellten Form nur für PT_2-Glieder definiert ist, und jeder zusätzliche Pol $\boldsymbol{\alpha}$, jede zusätzliche Nullstelle $\boldsymbol{\beta}$ bzw. jede zusätzliche Totzeit T_t eine Veränderung des Übertragungsverhaltens zur Folge haben.

Aus den oben genannten Gründen ist also ein Übergang vom Laplace- in den Frequenzbereich notwendig. Für stabile Systeme erfolgt dieser Übergang durch die Substitution $s = j\omega$ in der Übertragungsfunktion (Frey und Bossert, 2008, S. 173/175). Um Betrag und Phase des offenen Regelkreises zu berechnen, ist es dabei möglich die Reglerübertragungsfunktion $G_R(s)$ (Gleichung 4.2) und die Streckenübertragungsfunktion $G_s(s)$ (Gleichung 4.1) getrennt zu betrachten, da die vorliegende Reihenschaltung beider Systeme der Multiplikation der beiden Systeme entspricht (Frey und Bossert, 2008, S. 139). Dieses Vorgehen vereinfacht im Folgenden die analytische Auswertung. Um nun die Reglerverstärkung K_{PR} auslegen zu können, wird zunächst die gewünschte Durchtrittsfrequenz ω_D des offenen Regelkreises bestimmt. Dies ist anschaulich nach Abbildung 4.5 die Frequenz, bei der der Phasenwinkel des offenen Regelkreises um die Phasenreserve Ψ größer ist als $180°$:

$$-180° + \Psi_R = \varphi_0 \tag{4.20}$$

Der Phasenwinkel φ_0 des offenen Regelkreises ergibt sich dabei als Summe der Phasenwinkel der Strecke und des Reglers. Dies folgt aus der Tatsache, dass der Phasengang des offenen Regelkreises der Multiplikation der komplexen Phasengänge von Regler und Strecke entspricht. Damit ergibt sich

$$-180° + \Psi_R \stackrel{!}{=} \varphi_R(\omega_D) + \varphi_S(\omega_D) \tag{4.21}$$

als Bestimmungsgleichung für die Durchtrittsfrequenz ω_D. Die Phase des PI-Reglers φ_R kann dabei über die Formel

$$\varphi_R(\omega) = -\frac{pi}{2} + \arctan(T_N\omega) \tag{4.22}$$

(Otto, 1989) berechnet werden. Die Bestimmungsgleichung (in Bogengrad) lautet

dann

$$-180^\circ + \Psi_R \stackrel{!}{=} -90^\circ + \arctan(T_N\omega) + \varphi_S(\omega_D) \tag{4.23}$$

In dieser Formel wird deutlich, dass die Phase des Reglers unabhängig von der Reglerverstärkung bestimmt werden kann. Das Lösen der Gleichung 4.23 nach ω_D erfolgt dann im späteren Programmablauf numerisch (siehe Abschnitt 5.6)

Damit die Durchtrittsfrequenz ω_D nun tatsächlich an der gewünschten Stelle liegt, muss per Definition der Durchtrittsfrequenz für den Betrag des offenen Regelkreises die Bedingung

$$1 \stackrel{!}{=} |G_0(j\omega_D)| \tag{4.24}$$

$$1 \stackrel{!}{=} |G_R(j\omega_D)| \cdot |G_s(j\omega_D)| \tag{4.25}$$

erfüllt sein. Durch Einsetzen der Reglerübertragungsfunktion aus Gleichung 4.2 mit der oben beschriebenen Substitution $s = jw$ folgt

$$1 \stackrel{!}{=} \left|\frac{K_{PR}(1 + jT_N\omega_D)}{jT_N\omega_D}\right| \cdot |G_s(j\omega_D)| \tag{4.26}$$

$$\Leftrightarrow 1 \stackrel{!}{=} \left|K_{PR}\left(1 - j\frac{1}{T_N\omega_D}\right)\right| \cdot |G_s(j\omega_D)| \tag{4.27}$$

$$\Leftrightarrow K_{PR} \stackrel{!}{=} \left|\frac{1}{\left(1 - j\frac{1}{T_N\omega_D}\right) \cdot |G_s(j\omega_D)|}\right| \tag{4.28}$$

als Dimensionierungsvorschrift für die Reglerverstärkung K_{PR}. Damit ist der Regler nun vollständig ausgelegt.

5 Implementierung

In diesem Abschnitt wird der prinzipielle Aufbau des erstellten Programms „sysIdent“ dargestellt. Die dafür verwendete Programmiersprache ist C++. Diese wird ausgewählt, da sie eine hardwarenahe Programmierung und eine hohe Ausführungsgeschwindigkeit ermöglicht. Gleichzeitig ist es in dieser objektorientierten Erweiterung von C möglich, den Quellcode durch das Kapseln in verschiedene Klassen gut zu modularisieren. Der Code ist damit gut wartbar, und es ist möglich, Softwarmodule voneinander getrennt zu testen. Zur Dokumentation der Software wurde zusätzlich mit dem Tool „Doxygen“ (siehe Doxygen (2018)) eine HTML-Softwaredokumentation erstellt. Um den Rahmen dieser Arbeit zu begrenzen, soll hier daher nur auf eine Auswahl wichtiger Programmbestandteile eingegangen werden.

5.1 Programmablauf

Um die eingestellte Abtastzeit möglichst genau einzuhalten, ist es sinnvoll, das Senden und Empfangen von BACnet Telegrammen in getrennten Threads zu realisieren. Dieser Umstand ist in Abschnitt 5.2 genauer erläutert. Dazu wird das Programm in einen Main- und einen Anregungs-Thread aufgeteilt.

5.1.1 Main-Thread

Abbildung 5.1 zeigt ein Zustandsdiagramm für den Ablauf der main()-Funktion. Der aktuelle Zustand des Programms wird in der main()-Funktion in der *enumerated type* Variable „prog_state“ gespeichert. Die einzelnen Zustände werden im Folgenden kurz zusammengefasst.

Initializing

Die Parameter für die BACnet Kommunikation und die Anregung der Regelstrecke werden per Benutzerabfrage oder als Parameter beim Programmaufruf übergeben. Anschließend werden diese auf Gültigkeit untersucht. Dabei wird auf Vollständigkeit der übergebenen Parameter und einen gültigen Wertebereich geprüft. Sind die Parameter fehlerhaft, werden diese so lange erneut abgefragt, bis eine gültige Eingabe erreicht wurde.

Binding

Die Verbindung vom Programm zum Automationscontroller wird hergestellt. Dazu wird ein sogenanntes *Device Binding* über das BACnet Protokoll ausgeführt (siehe Kranz (2013) S. 308). Dabei wird die Netzwerkadresse des Automationscontrollers erfragt, und das virtuell erstellte BACnet Device wird dem Automationscontroller bekannt gemacht. Der Quellcode für diesen Abschnitt stammt aus dem Programm „readprop“ von Steve Karg (siehe Abschnitt 6.2). Ist das *Device Binding* nach einer bestimmten Zeit nicht erfolgreich, wird in den Zustand „Exit“ gewechselt und das Programm vorzeitig beendet.

Measuring

In diesem Schritt wird ein zweiter Thread für das Senden von Anfragen zum Lesen und Schreiben von BACnet Objekten gestartet. Die Anfragen werden ausschließlich über diesen Thread gesendet, während der main-Thread weiterhin die empfangenen BACnet Telegramme verarbeitet. Nach dem Starten des Threads wird sekündlich der Transport State Machine (tsm)-Timer in der BACnet Bibliothek aktualisiert. Dies ist notwendig, um eine Zeitüberschreitung eines angeforderten Wertes zu überwachen.

Listening

Dieser Zustand wird aktiviert, sobald der zweite Thread meldet, dass alle Anfragen an den Automationscontroller gesendet wurden. Der Unterschied zum vorherigen Zustand besteht hier darin, dass in der Konsole eine aktuelle Statistik über empfangene, ausstehende und fehlgeschlagene Anfragen an den Controller ausgegeben wird. Außerdem wird nun fortlaufend geprüft, ob entweder alle Anfragen

empfangen wurden, oder die Wartezeit für die verbleibenden Anfragen abgelaufen ist. Ist eine der Bedingungen erfüllt, erfolgt eine Transition in den Zustand *Analyze*.

Analyze

In diesem Zustand wird für jede Anfrage an den Automationscontroller im Nachhinein nochmal geprüft, ob eine Verletzung der Abtastzeit (gemessen vom geplanten Zeitpunkt bis zum Empfangen der Antwort) von mehr als 10% vorliegt. Außerdem wird der Mittelwert und die Standardabweichung der tatsächlich erreichten Abtastintervalle berechnet und ausgegeben. Dies soll dem Nutzer die Möglichkeit geben, die Qualität der Verbindung einzuschätzen.

Fit

In diesem Schritt wird aus den aufgezeichneten Signalen ein Systemmodell gebildet und anschließend eine Reglersynthese durchgeführt. Das Vorgehen hierzu ist in den Abschnitten 5.4 und 5.6 dargestellt. Hier soll lediglich angemerkt werden, dass die notwendigen Schritte im Programm durch einen try-catch Block geschützt werden. Damit werden zum Beispiel *Exceptions* beim Lösen eines schlecht konditionierten Ausgleichproblems abgefangen, und ein Programmabsturz wird verhindert. In diesem Fall erfolgt dann lediglich die Ausgabe einer Fehlermeldung.
Nach Abschluss der Reglersynthese wird sofort in den Zustand „Exit“ gewechselt.

Exit

In diesem Zustand werden keine Aktionen ausgeführt. Ist dieser Zustand erreicht, wird die Hauptprogrammschleife im nächsten Durchgang beendet.

Zusätzlich zum oben beschriebenen Zustandsautomaten erfolgt im main-Thread ein Polling der Netzwerkschnittstelle und eine Verarbeitung der eingehenden BACnet Netzwerktelegramme.

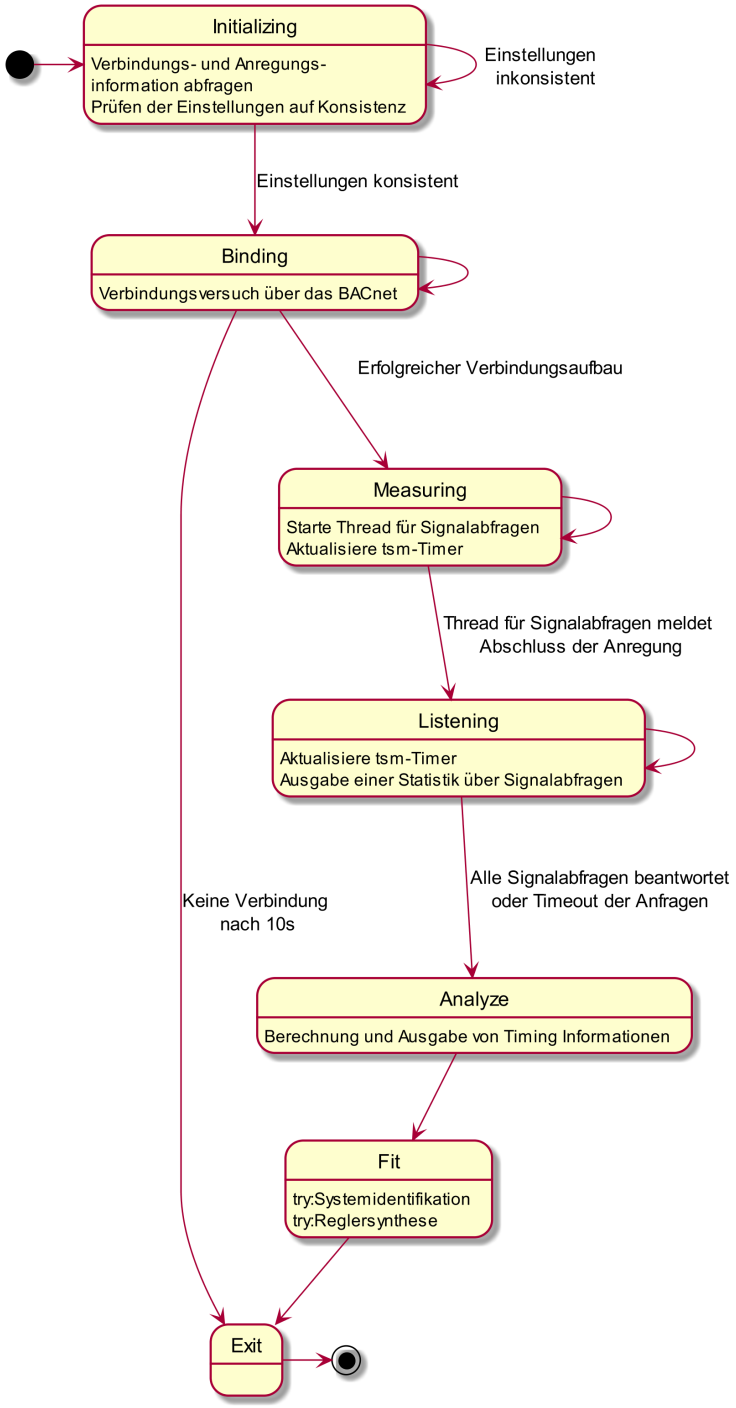

Abbildung 5.1: Aktivitätsdiagramm der Main-Funktion

5.1.2 Anregungs-Thread

Im Anregungs-Thread werden alle Anfragen an den Automationscontroller erzeugt. Im Wesentlichen besteht dieser Thread aus einer Instanz der Funktion „gen_requests“. Der

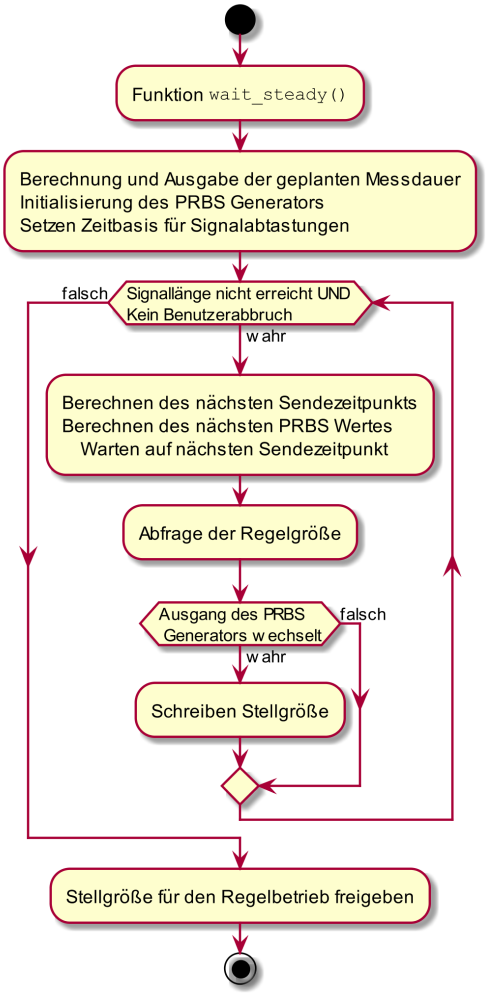

Abbildung 5.2: Aktivitätsdiagramm zur Anregung mittels PRBS

Ablauf für das Erzeugen der PRBS-Anregung ist exemplarisch in Abbildung 5.2 dargestellt. Für die Aufnahme einer Streckensprungantwort ist der Ablauf ähnlich, es wird jedoch lediglich einmalig ein Schreibbefehl für die Ausgangsgröße gesendet. Für die geregelte Sprungantwort werden Regel- und Stellgröße lediglich zyklisch ausgelesen. Es ist eine externe Anregung notwendig.

Wie in Abbildung 5.2 zu sehen beginnt der Anregungs-Thread mit dem Aufruf der Funktion „wait_steady“. Der Ablauf dieser Funktion ist nochmal in Abbildung 5.3 dargestellt. Es wird dort zunächst die Stellgröße ausgelesen und mit der BACnet Priorität 8 (siehe

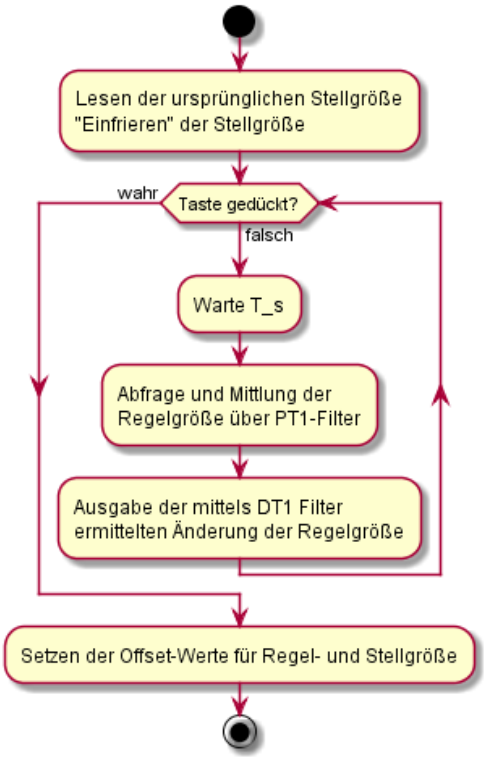

Abbildung 5.3: Aktivitätsdiagramm der Funktion „`wait_steady`“

2.1.1) zurück in den Automationscontroller geschrieben. Damit wird der normale Regelbetrieb der Strecke überschrieben. Anschließend wird zyklisch mit der Abtastzeit T_s der Wert der Regelgröße gemessen. Der Wert der Regelgröße wird dann mit zwei zeitdiskreten Messwertfiltern separat aufgearbeitet. Zum einen wird mit einem PT_1-Filter mit einer Zeitkonstante von $T_1 = 5T_s$ der Messwert tiefpassgefiltert, um eine Mittelwertbildung über die letzten Messwerte zu erreichen. Zum anderen wird mit einem DT_1-Filter mit der gleichen Zeitkonstante und einer Verstärkung $K_D = 1$ die Änderungsrate der Regelgröße ermittelt. Der aktuelle Messwert, die gefilterte Regelgröße und die gefilterte Änderungsrate der Regelgröße werden dann auf dem Bildschirm ausgegeben (siehe Abbildung 5.4). Zusätzlich wird die gefilterte Änderungsrate noch in Relation zu deren Maximalwert gesetzt.

```
Reading initial output... 27.3227
Output locked.
Wait till input is constant. Then press any key to continue.

                        Filtered
Input    Filtered       Slope[sec^-1]   % of Max Slope
 29.049  29.122          -0.029                 49.759
```

Abbildung 5.4: Beispiel für eine Bildschirmausgabe der Funktion „`wait_steady`“

Der Benutzer muss nun anhand der angezeigten Werte entscheiden, ab welchem Zeitpunkt die Regelgröße als konstant angenommen werden kann. Zu diesem Zeitpunkt besteht eine Änderung der Regelgröße hauptsächlich aus Rauschen. Da hier noch keinerlei Kenntnisse über die Strecke vorliegen, ist es schwierig, diesen Zeitpunkt automatisiert zu bestimmen. Aus diesem Grunde ist dieser Schritt an den Benutzer ausgelagert worden.

5.2 Implementierung der BACnet-Kommunikation

Wie bereits in Abschnitt 2.1.1 dargestellt, kommuniziert das erstellte Programm bei der Aufnahme der zur Identifikation benötigten Testsignale über Ethernet mit einem externen Automationscontroller. Es ist also notwendig, über das BACnet Protokoll Schreib- und Leseanfragen zu verschicken. Dazu sind einige Schritte notwendig. So muss ein virtuelles BACnet Client Device eingerichtet werden, das Anfragen senden und die Antworten darauf empfangen kann.

In dieser Arbeit wird dafür die Bibliothek „*BACnet Stack*“ (Karg, 2017) in der Version 0.8.4 genutzt. Es handelt sich dabei um eine C/C++-Bibliothek, die unter Windows, Linux und auch auf Mikrocontrollern lauffähig ist. Eine vollständige Umsetzung des BACnet Standards ist zum Zeitpunkt der Veröffentlichung dieser Arbeit noch nicht implementiert, es sind jedoch alle hier benötigten Funktionen vorhanden.

BACnet Stack wird mit einer Auswahl lauffähiger Demo-Programme ausgeliefert, die jeweils eine einfache Funktionalität erfüllen. Für diese Arbeit werden die Programme „readprop“ und „writeprop“ vom Autor Steve Karg als Grundlage genutzt. Das Programm „readprop“ ist in der Lage einen BACnet Datenpunkt zu lesen, das Programm „writeprop“ kann einen Datenpunkt beschreiben. Bei beiden Programmen handelt es sich um

kommandozeilen-basierte Tools. Bereits für diese beiden Minimalbeispiele wurden einige hundert Zeilen Quellcode geschrieben. Dort wird u.a. ein virtuelles BACnet Client Device emuliert, dieses wird mit verschiedenen Handlern für Netzwerkanfragen ausgestattet, die Netzwerkschnittstelle wird mittels Polling-Verfahren auf BACnet Telegramme abgefragt, und ankommende Telegramme werden entschlüsselt.

Ein einfacher Ansatz wäre hier, diese Programme unverändert durch ein übergeordnetes Programm aufzurufen und dort die zurückgegebenen Werte zu verarbeiten. Allerdings wird dabei jedes Mal erneut das gesamte Programm gestartet, eine neue BACnet Verbindung geöffnet und dann erst die gewünschte Operation durchgeführt. Dies kann zu hohen Latenzen der Schreib-/Lesezugriffe führen. Das Identifikationsverfahren aus Abschnitt 3 ist jedoch für zeitlich äquidistante Abtastung ausgelegt. Es ist also notwendig, die zeitliche Verzögerung beim Schreiben und Lesen möglichst gering zu halten. Soll z.B. in einem Abtastschritt die Stellgröße geschrieben und die Regelgröße ausgelesen werden, müssen beide Befehle innerhalb einer, im Vergleich zur Abtastzeit, möglichst kurzen Zeit ausgeführt werden. Um dies zu ermöglichen, wird sich dafür entschieden die BACnet Kommunikation direkt in C++ in das zu erstellende Programm zu integrieren.

Dazu wird das Programm „readprop“ von Steve Karg als Gerüst genutzt, erweitert, teils restrukturiert und um Funktionalitäten aus dem Programm „writeprop“ erweitert. Um mehrere Datenpunkte möglichst schnell hintereinander (ideal wäre: zeitlich parallel) lesen/schreiben zu können, wird hierbei das Senden von Anfragen und das Empfangen der Antwort des Automationscontrollers voneinander getrennt und in zwei seperate Threads ausgelagert.

Für Regel- und Stellgröße wird je zunächst eine Objektinstanz der Klasse „Req_List“ erzeugt. Diese Klasse besteht im Wesentlichen aus einer Liste ausstehender und empfangener Anfragen. Diese Anfragen werden jeweils als Instanz der Klasse „Request“ repräsentiert.

Beim Senden einer Anfrage zum Lesen oder Schreiben wird also ein Objekt dieser Klasse erzeugt und in die Liste ausstehender Anfragen („std :: list <Request> pending“) der zugehörigen Instanz von „Req_List“ eingetragen. Wird nun die Antwort des Automationscontrollers empfangen, werden in einem getrennten Empfangs-Thread die notwendigen Felder aktualisiert, und das Objekt wird aus der „pending“- in eine gleichartige „received“-Liste verschoben.

Da hier zwei Threads auf einem gemeinsamen Datenbereich tätig sind, ist es notwendig, diesen Bereich zu schützen, um die Integrität der Daten sicherzustellen. Dabei soll vermieden werden, dass durch gleichzeitiges Lesen und Schreiben im gleichen Bereich unerwartete Daten entstehen. Die Zugriffe auf die Listen werden also jeweils mit einer Instanz der „std :: mutex“ und den zugehörigen Befehlen „mutex::lock()“ und

„mutex::unlock()“ abgesichert. Die Klasse „std :: mutex“ ist Bestandteil der C++11 Standards (cplusplus.com, 2018).

5.3 Erzeugung von PRBS-Signalen

Im Folgenden wird die Implementierung der Erzeugung von PRBS-Signalen in C++ beschrieben. Der zugehörige Quellcode befindet sich in der Klasse „PRBS_gen“.

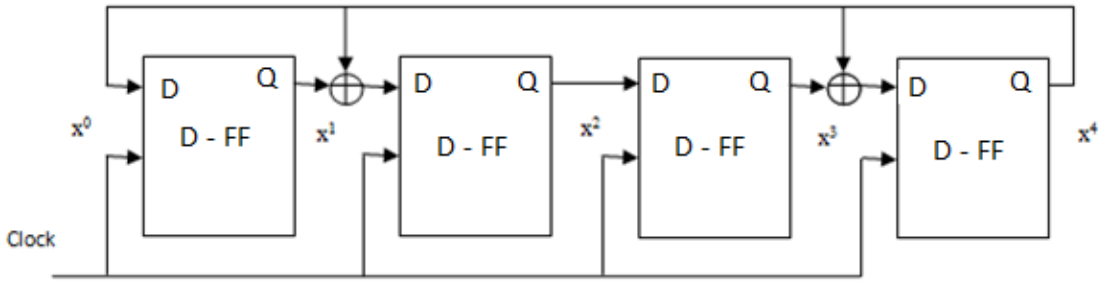

Abbildung 5.5: Galois LFSR (Nithya u. a., 2015, Figure 2)

Ein PRBS-Signal kann mittels eines primitiven Generatorpolynoms über den Galois-Körper erzeugt werden (Bohn und Unbehauen, 2016, S. 543). Dieses Polynom lässt sich nun über ein linear rückgekoppeltes Schieberegister nach Abbildung 5.5 repräsentieren. Dort ist als Beispiel das Polynom

$$\begin{aligned} &1 \cdot x^0 + 1 \cdot x^1 + 0 \cdot x^2 + 1 \cdot x^3 + 1 \cdot x^4 \\ =&1 + x + x^3 + x^4 \end{aligned} \tag{5.1}$$

dargestellt. Für die Implementierung dieser Darstellung in C++ können nun zunächst die dargestellten charakteristischen XOR-Verknüpfungen als Bitmaske abgespeichert werden. Für den Eingang des linken FlipFlops wird auch ein Bit der Maske gesetzt. Streng genommen liegt hier keine XOR-Verknüpfung vor. Da aber aber kein weiteres Signal an dieser Verknüpfungsstelle ankommt, kann diese Stelle auch als ein XOR angesehen werden, dessen einer Eingang stets den Wert 0 annimmt. Das vereinfacht die Implementierung. Für das obige Polynom ergibt sich dann

$$\text{pol} = 11010_2 \tag{5.2}$$

als Bitmaske. Die aktuellen Zustände der FlipFlops werden zusätzlich binär in eine Variable kodiert.

Abhängig von der Anzahl der zu bestimmenden Parameter und des zu untersuchenden Zeitbereichs kann es sinnvoll sein, Testsignale unterschiedlicher Längen zu verwenden. Aus diesem Grunde werden primitive Polynome über den Galois-Körper verschiedener Ordnung implementiert. Die Länge der Periode ergibt sich jeweils aus der Anzahl der möglichen Zustände der Registerwerte. Davon ausgenommen ist der Zustand, dass alle Register den logischen Wert '0' annehmen, da in diesem Fall keine Änderung des Zustandes mehr stattfinden kann. Dieser Umstand ergibt sich daraus, dass die Register eigenerregt sind und keinen Eingang besitzen. Für die Signallänge (bis zur Wiederholung der PRBS-Sequenz) gilt bei einer Registerlänge n somit:

$$\text{Signallänge} = n^2 - 1 \tag{5.3}$$

Wie in Abschnitt 3.1.1 erläutert, ist die Eigenschaft eines sehr gleichmäßigen Spektrums der PRBS-Signale nur gegeben, wenn sie über eine gesamte Periodendauer betrachtet werden. Ein Kürzen des Signals führt zu erheblichen Änderungen im Frequenzgang. Um unterschiedliche Signallängen zu ermöglichen, werden dann die in Tabelle 5.1 aufgelisteten Polynome implementiert. Die gelisteten Bitmasken werde, wie oben angedeutet, aus den jeweiligen Polynomen gewonnen.

n	Signallänge	Primitives Polynom	Bitmaske(Binär)	Bitmaske(Hex)
4	15	$x^4 + x^3 + 1$	1001_2	0x 09
5	31	$x^5 + x^3 + 1$	$1\ 0010_2$	0x 12
6	63	$x^6 + x^5 + 1$	$10\ 0001_2$	0x 21
7	127	$x^7 + x^6 + 1$	$100\ 0001_2$	0x 41
8	255	$x^8 + x^7 + x^6 + x + 1$	$1011\ 0001_2$	0x B1
9	511	$x^9 + x^5 + 1$	$1\ 0000\ 1000_2$	0x108
10	1023	$x^10 + x^7 + 1$	$10\ 0000\ 0100_2$	0x204

Tabelle 5.1: Übersicht über die verwendeten PRBS-Signale und die resultierenden Bitmasken. Die gelisteten Generatorpolynome stammen aus Bohn und Unbehauen (2016), Tabelle 9.1

Im Programm werden diese Bitmasken, sowie Signalamplitude und Offset des Ausgangssignals beim Aufrufen der Initialisierungsfunktion „PRBS_gen::init“ (Listing 5.1) gesetzt.

```
void PRBS_gen::init(int n, double offset_val, double ampl_val,
                    double up_val)
{
    offset = offset_val;     //set offset of output
    ampl = ampl_val;         //set amplitude
    reg = 1;                 //initialize shift register
    upsample = up_val;       //set upsample factor

    //set polynomial
    switch (n)
    {
    case 4:
        poly = 0x09;
        break;
    case 5:
        poly = 0x12;
        break;
...
    default:
        printf("Class PRBS_gen: Unsupported Parameter for n!\n");
        if(n>10){
            n=10;
            printf("n was set to maximum allowed value (n=10)\n!");
        }else{
            n=4;
            printf("n was set to minimum allowed value (n=4)\n!");
        }
    }
}
```

Listing 5.1: Funktion „PRBS_gen::init“ zur Initialisierung des PRBS-Generators (gekürzt)

Zusätzlich kann ein ganzzahliger Faktor upsample angegeben werden. Dieser gibt an, nach wie vielen Abtastzyklen das Schieberegister aktualisiert werden soll. Damit wird die Anregung der Regelstrecke "verlangsamt", was eine niederfrequentere Anregung der Strecke zur Folge hat. Es ist also möglich das Ausgangssignal y schnell abzutasten, ohne viel Signalleistung in einen hohen Frequenzbereich zu verlagern. Das ist etwa zur Schätzung einer Totzeit von n_d Abtastschritten sinnvoll, da diese durch kleinere Abtastschritte genauer bestimmt werden kann. Ist gleichzeitig eine sehr viel größere Zeitkonstante im System vorhanden, ist es jedoch nicht sinnvoll, einen großen Teil der Signalleistung in den stark gedämpften oberen Frequenzbereich (jenseits der dominierenden Zeitkonstante) des Systems zu legen. Das ist mit einem entsprechenden upsample Faktor möglich.

Das Schieberegister wird bei der Initialisierung mit dem Wert $\texttt{reg} = 1$ initialisiert, es wird also lediglich das *Least Significant Bit* (LSB) gesetzt. Der Systemausgang beginnt damit mit einem *High*-Pegel. Der jeweils nächste Ausgangs- und Schieberegisterwert wird mittels der Funktion „PRBS_gen::next_out“ wie folgend bestimmt:

```
double PRBS_gen::next_out()
{
    static int req_count = 0;    //request counter used for up-sampling
    old_val = reg&0x01;          //store last state

    // update only after n=upsample values and after first request
    if (!first_req && !(req_count%upsample))
    {
        // if LSB (output-bit) is high
        if (reg&0x01)
        {
            reg = (reg>>1)^poly; //XOR with mask and shift
        }
        else         //LSB is low
        {
            reg = reg>>1; //just shift right
        }
    }

    // update request count
    req_count++;

    // returns high or low level, according to LSB
    return (reg&0x01) ? offset+ampl : offset-ampl;
}
```

Listing 5.2: Funktion „PRBS_gen::next_out“ zum Berechnen des nächsten Registerwertes und des Systemausgangs des PRBS-Generators (gekürzt)

Dieser Code arbeitet äquivalent zu einem Schieberegister nach Abbildung 5.5.

5.4 Numerische Lösung des linearen Ausgleichsproblems

Im Kapitel 3 wurde das Ausgleichsproblem mit der Bestimmungsgleichung 3.20

$$M^T M p_M = M^T y$$

aufgestellt. Dazu ist es hier nun zunächst notwendig, die Datenmatrix M

$$M = \left[\begin{array}{ccc:ccc} -y(n+n_d) & \dots & -y(n_d) & u(n) & \dots & u(0) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -y(N-1) & \dots & -y(N-1-n) & u(N-n_d) & \dots & u(N-n_d-n) \end{array}\right]$$

aus Gleichung 3.18 zu füllen, was mit der Funktion „system_fit::gen_M“ aus Listing 5.3 durchgeführt wird. Die Matrix wird dort allerdings so erweitert, dass N-Zeilen entstehen. Es wird dadurch mit dem Erstellen der ersten Zeile nicht gewartet, bis genügend Messwerte für alle Spalten vorhanden sind. Stattdessen werden diese Werte zu Null definiert. Damit wird erreicht, dass auch für die Messwerte innerhalb der $n+n_d-1$ ersten Abtastschritte eine Spalte der Matrix und damit später ein Element im Residuenvektor r (siehe Gleichung 3.15) erzeugt wird. Ohne diese Erweiterung werden diese Messwerte bei der Ermittlung des Gütefunktionals I_v (Gleichung 3.24) nicht berücksichtigt. Es würden also bei großen Werten der Systemordnung n und der Totzeit n_d „künstlich“ die Werte für das Gütefunktional I_v verbessert, was zu falschen Ergebnissen für n und n_d führen kann. Durch das oben beschriebene Vorgehen tritt dieser Effekt nicht ein.

```
void system_fit::gen_M(real_2d_array &M, real_1d_array &u,
                       real_1d_array &y, int n, int nd)
{
    int N = u.length();
    M.setlength(N,2*n+1);
    int index;

    //iterate over rows
    for(int k=0; k<N; k++)
    {
        //fill y
        for(int i=0; i<n; i++)
        {
            index = k-i-1;
            if (index<y.length() && index>=0)
```

```
16                  M[k][i]=-y[index];
                else
18                  M[k][i]=0;
            }
20
            //fill u
22          for(int i=0; i<n+1; i++)
            {
24              index = k-i-nd;
                if (index<u.length() && index>=0)
26                  M[k][i+n]=u[index];
                else
28                  M[k][i+n]=0;
            }
30      }
}
```

Listing 5.3: Funktion „system_fit::gen_M“ zum Füllen der Datenmatrix M

Nach dem Erzeugen der Datenmatrix M kann nun das Ausgleichsproblem gelöst werden. Dazu wird die C++-Variante der Bibliothek „*ALGLIB*“ (ALGLIB-Project, 2018b) in Version 3.13.0 eingesetzt. Diese enthält eine Funktion alglib :: lsfitlinear , die das lineare Problem lösen kann. Dabei wird zunächst eine QR-Zerlegung der Matrix durchgeführt, und anschließend das Ausgleichsproblem mit einem je nach Konditionierung der Matrix passenden Algorithmus gelöst (ALGLIB-Project, 2018a, Abschnitt „lsfitlinear function“). Durch das Nutzen dieses recht aufwendigen Algorithmus kann die Identifikation auch bei schwachen Anregungen bzw. schlecht konditionierten Matrizen M noch gelöst werden.

5.5 Evaluation des identifizierten Systems

Zur Vorbereitung der Reglersynthese ist es zunächst notwendig, die bei der Identifikation ermittelte z-Übertragungsfunktion der Strecke mathematisch auszuwerten. Das Vorgehen dazu wird untenstehend erläutert.

5.5.1 Transformation in den s-Bereich

Bei der Identifikation des Systems wird über den Parametervektor $\boldsymbol{p}$ und die Prüfung der Modellordnung n und Totzeit n_d eine Übertragungsfunktion

$$G_M(z) = \frac{\tilde{Y}_M(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} \cdot z^{-n_d}$$

aus Gleichung 3.7 bestimmt. Um daraus Zeitkonstanten bestimmen zu können, ist es notwendig, diese in den s-Bereich zu transformieren. Die Totzeit in Abtastschritten n_d lässt sich mit der Abtastzeit leicht in den s-Bereich übertragen. Mit der Abtastzeit T_s gilt damit offensichtlich der Zusammenhang

$$T_t = n_d \cdot T_s \tag{5.4}$$

So ist z.B. eine diskrete Totzeit von $n_d = 3$ Abtastschritten bei einer Abtastzeit von $T_s = 500msek$ äquivalent zu einer kontinuierlichen Totzeit von $T_t = 1,5sek$. Damit ändert sich der Term für die Totzeit in der Übertragungsfunktion beim Übergang in den s-Bereich wie folgend:

$$z^{-n_d} \rightarrow e^{-n_d T_s s} = e^{-T_t s} \tag{5.5}$$

(Frey und Bossert, 2008, S.138)

Der gebrochen-rationale Teil der Übertragungsfunktion lässt sich über die sogenannte *Bilineare Transformation*

$$z = \frac{1 + Ts/2}{1 - Ts/2} = \frac{2 + Ts}{2 - Ts} \tag{5.6}$$

in den s-Bereich transformieren (von Grünigen, 2014, S. 255).

Zunächst wird Gleichung 3.7 zur besseren Übersicht mit z^n erweitert.

$$\begin{aligned} G_M(z) &= \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} \cdot \frac{z^n}{z^n} \cdot z^{-n_d} \\ &= \frac{b_0 z^n + b_1 z^{n-1} + \ldots + b_n}{z^n + a_1 z^{n-1} + \ldots + a_n} \cdot z^{-n_d} \end{aligned} \tag{5.7}$$

Durch Einsetzen dieser Transformationsbedingung (Gleichung 5.6) und der oben beschriebenen Transformation der Totzeit ergibt sich nun als s-Übertragungsfunktion

$$G_M(s) = \frac{b_0\left(\frac{2+Ts}{2-Ts}\right)^n + b_1\left(\frac{2+Ts}{2-Ts}\right)^{n-1} + \ldots + b_n}{\left(\frac{2+Ts}{2-Ts}\right)^n + a_1\left(\frac{2+Ts}{2-Ts}\right)^{n-1} + \ldots + a_n} \cdot e^{-T_t s} \tag{5.8}$$

Diese Gleichung wird mit $(2-Ts)^n$ erweitert, und ein Koeffizient $a_0 = 1$ wird im Nenner eingeführt:

$$\begin{aligned} &G_M(s) \\ &= \frac{b_0\left(\frac{2+Ts}{2-Ts}\right)^n + b_1\left(\frac{2+Ts}{2-Ts}\right)^{n-1} + \ldots + b_n}{a_0\left(\frac{2+Ts}{2-Ts}\right)^n + a_1\left(\frac{2+Ts}{2-Ts}\right)^{n-1} + \ldots + a_n} \cdot \frac{(2-Ts)^n}{(2-Ts)^n} \cdot e^{-T_t s} \\ &= \frac{b_0(2+Ts)^n(2-Ts)^0 + \ldots + b_n(2+Ts)^0(2-Ts)^n}{a_0(2+Ts)^n(2-Ts)^0 + \ldots + a_n(2+Ts)^0(2-Ts)^n} \cdot e^{-T_t s} \end{aligned} \tag{5.9}$$

In Summenform ergibt sich damit:

$$G_M(s) = \frac{\sum_{i=0}^{n}\left(b_i(2+Ts)^{n-i}(2-Ts)^i\right)}{\sum_{i=0}^{n}\left(a_i(2+Ts)^{n-i}(2-Ts)^i\right)} \cdot e^{-T_t s} \tag{5.10}$$

Diese Darstellung soll nun in eine äquivalente Darstellung über zwei s-Polynome $B_c(s)$ und $A_c(s)$ und ein Totzeitglied $e^{-T_t s}$ umgerechnet werden:

$$G_M(s) \stackrel{!}{=} \frac{B_c(s)}{A_c(s)} \cdot e^{-T_t s} \tag{5.11}$$

$$= \frac{b_{c0}s^n + b_{c1}s^{n-1} + \ldots + b_{cn}}{s^n + a_{c1}s^{n-1} + \ldots + a_{cn}} \cdot e^{-T_t s} \tag{5.12}$$

Die Berechnung kann für das Zähler- und das Nennerpolynom getrennt durchgeführt werden. Diese Umrechnung werden in der Funktion „system_fit :: bil2d2c“ implementiert. Sie rechnet jeweils eins der z-Polynome in ein s-Polynom um. Bei der Umrechnung des Polynoms

$$A(z) = z^n + a_1 z^{n-1} + \ldots + a_n \cdot z^{-n_d} \tag{5.13}$$

in das Polynom

$$A_c(s) = s^n + a_{c1}s^{n-1} + \ldots + a_{cn} \tag{5.14}$$

wird dabei in einer Schleife über die Werte $i = 0 \ldots n$ jeweils der Term $\left(a_i(2+Ts)^{n-i}(2-Ts)^i\right)$ aus Gleichung 5.10 getrennt berechnet und aufsummiert. Dabei werden zunächst die Polynome

$$s_p = (2+Ts)^{n-i} \quad \text{sowie} \tag{5.15}$$

$$s_m = (2+Ts)^i \tag{5.16}$$

berechnet. Anschließend werden diese miteinander und mit dem zugehörigen Parameter a_i multipliziert und zum Ergebnispolynom addiert. Der zugehörige Quellcode findet sich in Listing 5.4.

```
real_1d_array system_fit::bil2d2c(real_1d_array a, double Ts)
{
    int n = a.length();          //length of polynomial
    int i,j;                     //counters
    double bil_p[] = {2, Ts};    //vector for (2+T_s*s)
    double bil_m[] = {2, -Ts};   //vector for (2-T_s*s)
    const double init[] = {1};   //value used to initialize s_p term
    real_1d_array s_p,s_m,b_p,b_m,s2_p,s2_m,temp;  //temporary results
    real_1d_array a_cont;        //final results

    //initialize output vector
    a_cont.setlength(n);
    for (i=0;i<n;i++)
    {
        a_cont[i] = 0;
    }

    //set length for temporary results
    s_p.setlength(n);
    s2_p.setlength(n);
    s2_m.setlength(n);
    s_m.setlength(n);
    temp.setlength(n);
    b_p.setlength(2);
    b_p.setcontent(2,bil_p);
    b_m.setlength(2);
    b_m.setcontent(2,bil_m);

```

```
    //calculate bilinear transformation
30  for (i=0;i<n;i++)
    {
32      //calculate (1+s)^i
        s_p.setcontent(1,init);
34      for(j=0;j<i;j++)
        {
36          //polynomial multiplication
            convr1d(s_p,1+j,b_p,2,s2_p);
38          s_p = s2_p;
        }
40
        //calculate (1-s)^n-1-i
42      s_m.setcontent(1,init);
        for(j=0;j<n-1-i;j++)
44      {
            //polynomial multiplication
46          convr1d(s_m,1+j,b_m,2,s2_m);
            s_m = s2_m;
48      }
        //multiply s_m and s_p
50      convr1d(s_m,s_m.length(),s_p,s_p.length(),s2_m);

52      //multiply with parameter a
        vmul(s2_m.getcontent(),s2_m.length(),a[n-1-j]);
54      //add to result
        vadd(a_cont.getcontent(),s2_m.getcontent(),a_cont.length());
56  }

58  // return s-polynomial
    return a_cont;
60  }
```

Listing 5.4: Funktion „system_fit::bil2d2c“ zur Anwendung der bilinearen Transformation auf ein Polynom

Nachdem die obige Funktion auf die beiden z-Polynome angewendet wurde, werden diese nur noch jeweils durch das berechnete Element ac_0 geteilt, um wie in Gleichung 5.12, eine Normierung von $ac_0 = 1$ zu erreichen.

Um die korrekte Implementierung der gezeigten Transformation zu verifizieren, werden die Ergebnisse für einige Testübertragungsfunktionen mit denen der MATLAB® - Funktion „d2c (..., ' tustin ')“ verglichen. Der Parameter ' tustin ' spezifiziert hier, dass

eine bilineare Transformation durchgeführt werden soll. Die erzielten Ergebnisse waren bei der Anwendung der MATLAB® -Funktion und der eigenen Implementierung identisch, das gezeigte Verfahren kann also als korrekt angenommen werden.

5.5.2 Bestimmung der Pole und Zeitkonstanten

Nachdem das System im vorherigen Abschnitt in den s-Bereich transformiert wurde, können dort nun einige relevante Kenngrößen bestimmt werden. Diese werden in der Funktion „system_fit::complete_eval“ für jede getestete Identifikation berechnet und in einer Datei („Fit.log“) im Ausführungsverzeichnis des Programms gespeichert. Dazu werden zunächst die Nullstellen des Nennerpolynoms der Streckenübertragungsfunktion $A_c(s)$ aus Gleichung 5.11 berechnet. Die Berechnung erfolgt mit der Funktion „alglib::polynomialsolve“ der *ALGLIB*-Bibliothek. Diese Nullstellen werden als die Pole $\alpha_1 \ldots \alpha_n$ bezeichnet. Anschließend werden folgende Parameter berechnet:

Stabilität

Gemäß Abschnitt 4.2 wird die Stabilität der Strecke anhand der Lage der komplexen Pole überprüft. Alle Pole müssen bei einem stabilen System in der linken s-Halbebene liegen (falls Grenzstabilität, wie hier, als instabil gewertet wird).

Statische Verstärkung K_S

Die statische Signalverstärkung der Regelstrecke für $t \to \infty$ kann nach dem *Endwertsatz der Laplace-Transformation* (Frey und Bossert, 2008, S.342) über die Grenzwertbildung $\lim_{t\to\infty} g_M(t) = \lim_{s\to 0} G_M(s)$ gebildet werden. Dieser Wert wird im Programm durch die Division der konstanten Koeffizienten der s-Polynome bestimmt:

$$K_S = b_{c0}/a_{c0} \tag{5.17}$$

Damit findet keine echte Grenzwertbildung statt, für Koeffizienten $b_{c0}, a_{c0} \in \mathbb{R}, a_{c0} \neq 0$ kann der Grenzwert jedoch so bestimmt werden. Dieser Wert wird ausschließlich für den Benutzer zur Plausibilitätsprüfung der Identifikation ausgegeben. Für die weitere Identifikation wird dieser Wert nicht benötigt.

Vektor der Zeitkonstanten T

Die einzelnen Zeitkonstanten des Systems $T_1 \ldots T_n$ werden (wie in Gleichung 4.10) als Kehrwert des Betrags der Pole $\alpha_1 \ldots \alpha_n$ bestimmt:

$$T_i = \frac{1}{|\alpha_i|} \tag{5.18}$$

Zusammengesetzt ergibt sich daraus der Vektor $\boldsymbol{T}$. Die darin enthaltenen Zeitkonstanten werden für die Reglersynthese benötigt.

5.6 Reglersynthese

Im Anschluss an eine erfolgreiche Systemidentifikation wird die Reglersynthese durchgeführt. Dies geschieht mit der Funktion „controller ::synthesize“. Das dazu verwendete Vorgehen ist in Abschnitt 4 ausführlich beschrieben.
Die Eingangsgröße der Reglersynthese stellt eine Objektinstanz der Klasse „system_fit“ dar. Dort sind die s-Polynome der Übertragungsfunktion und die unter Abschnitt 5.5.2 gelisteten Parameter bereits enthalten. Die Nachstellzeit kann damit nach Abschnitt 4.3 direkt auf den Wert der größten Zeitkonstante festgelegt werden. Anschließend wird nach Gleichung 4.18 der gewünschte Phasenrand Ψ_R ausgegeben. Nun muss die gewünschte Durchtrittsfrequenz ω_D des offenen Regelkreises berechnet werden (siehe Abschnitt 4.4). Die Bestimmungsgleichung der Phasenbedingung

$$\begin{aligned} -180^\circ + \Psi_R &\stackrel{!}{=} -90^\circ + arctan(T_N\omega) + \varphi_S(\omega_D) \qquad \text{bzw.} \\ \pi + \Psi_R &\stackrel{!}{=} -\frac{\pi}{2} + arctan(T_N\omega) + \varphi_S(\omega_D) \end{aligned} \tag{5.19}$$

(nach Gleichung 4.23) muss also erfüllt werden. Um einen geringen Implementierungsaufwand treiben zu müssen, wird die Durchtrittsfrequenz ω_D hier durch einfaches Einsetzen von Testwerten ermittelt. Dies wird über eine for-Schleife implementiert:

```
for(w=wstep; w<wmax&&!limit; w+=wstep)
{
    Gs = eval_tf(w, system);
    phi_r = -atan(1/(Tn*w));
    limit = phi_r+arg(Gs)<-M_PI+argmargin;
}
```

Listing 5.5: Auszug aus Funktion „controller::controller“ zur Synthese eines Reglers

Dort wird bis zu einer maximalen Kreisfrequenz

$$\omega_{\text{max}} = 5 \cdot \frac{1}{T_s} = 5 \cdot f_s, \tag{5.20}$$

also bis zum zehnfachen Wert der Nyquist-Frequenz, mit einer Schrittweite von

$$\omega_{step} = \frac{\omega_{max}}{10000} = \frac{5 \cdot f_s}{\omega_{max}} = \frac{f_s}{2000} \tag{5.21}$$

getestet, ob die Phasenbedingung erreicht wurde. Bei einer Abtastrate von $f_s = 2Hz$ wird die Durchtrittsfrequenz ω_D also beispielsweise mit einer Schrittweite von $\omega_{step} = 1\,\mathrm{mHz}$ ermittelt. Diese Genauigkeit wird als ausreichend angenommen. Das Erreichen der Phasenbedingung wird in der for-Schleife durch das Flag „ limit “ angezeigt.

Die dafür notwendige Auswertung der Streckenübertragungsfunktion erfolgt über die Funktion „controller :: eval_tf“ (siehe folgendes Listing 5.6). Zur Erläuterung dieser Funktion wird mit der Streckenübertragungsfunktion nach Gleichung 4.1 begonnen:

$$G_s(s) = e^{-T_t s} \cdot \frac{b_{cn}s^n \ldots b_{c1}s + b_{c0}}{s^n \ldots a_{c1}s + a_{c0}}$$

Da vorher bereits die Stabilität der Strecke geprüft wurde, erfolgt der Übergang in den Frequenzbereich nun durch die Transformation $s \rightarrow j\omega$ (Frey und Bossert, 2008, S. 173/175).

$$G_s(j\omega) = e^{-j\omega T_t} \cdot \frac{b_{cn}(j\omega)^n \ldots b_{c1}(j\omega) + b_{c0}}{(j\omega)^n \ldots a_{c1}(j\omega) + a_{c0}} \tag{5.22}$$

Der gebrochen-rationale Teil der Streckenübertragungsfunktion und der Anteil der Totzeit werden nun getrennt betrachtet. Die Totzeit

$$G_{T_t}(j\omega) = e^{-j\omega T_t} \tag{5.23}$$

hat dabei einen konstanten Betrag von $|G_{T_t}(j\omega)| = 1$ und einen Winkel von $\varphi_{T_t}(j\omega) = -\omega T_t$. Der komplexe Wert des Nenners und des Zählers wird jeweils getrennt über eine for-Schleife berechnet. Diese drei Komponenten ergeben dann zusammengesetzt den Wert der Übertragungsfunktion.

```
std::complex<double> controller::eval_tf(double w,
                                         system_fit open_loop)
{
    std::complex<double> result = 0;
    std::complex<double> den = 0;        //denominator
    std::complex<double> enu = 0;        //enumerator
    std::complex<double> jw = w*1i;      //  (j\omega)

    //calculate enumerator
    for (int k=0;k<open_loop.Geta_cont().length();k++)
```

```
    {
12      enu += pow(jw,k)*open_loop.Geta_cont()[k];
    }
14  //calculate denominator
    for (int k=0;k<open_loop.Getb_cont().length();k++)
16  {
      den += pow(jw,k)*open_loop.Getb_cont()[k];
18  }

20  // result equals den/enu rotated by phase shift of transport delay
    result = den/enu*std::polar(1.0,-open_loop.GetTt()*w);
22  return result;
}
```

Listing 5.6: Funktion „controller::eval_tf“ zur Berechnung des komplexen Wertes der Streckenübertragungsfunktion bei einer bestimmten Frequenz

Die Reglersynthese wird jeweils für die Dämpfungswerte $D = 0{,}5; 0{,}7; 1; 1{,}5$ ausgeführt. Dabei bleibt die Nachstellzeit konstant, und die Reglerverstärkung wird jeweils neu berechnet. Die Ergebnisse werden tabellarisch ausgegeben und in der Datei „controller.log“ im Ausführungsverzeichnis gespeichert. Dann kann vom Benutzer je nach Anforderung an das Syntheseergebnis (siehe Abschnitt 4.1) ein passender Parametersatz ausgewählt werden.

5.7 Benutzerführung

Der Benutzer kann folgende Werte für die durchzuführende Systemidentifikation parametrieren:

Device Instance (zugelassener Wertebereich 0 bis 4194303)
BACnet Gerätekennung des Zielgeräts (Automationscontroller)

BACnet Objekt für die Stellgröße
Object Type (AO=1, AVAL=2): Typ des BACnet Objekt. Zulässige Werte sind Analog Output (AO) und Analog Value (AVAL)
Object Instance (1 bis 4194303): Instanznummer des Objekttyps auf dem Zielgerät

BACnet Objekt für die Regelgröße
Object Type (AI=0, AVAL=2): Typ des BACnet Objekt. Zulässige Werte sind Analog Input (AI) und AVAL
Object Instance (1 bis 4194303): Instanznummer des Objekttyps auf dem Zielgerät

Art der Anregung (Step = 0, PRBS = 1, Measure = 2)
Auswahl der gewünschten Anregungsform nach Abschnitt 3.1. „Step“ steht dabei für die ungeregelte Streckensprungantwort, „Measure“ muss für die geregelte Sprungantwort verwendet werden.

Amplitude (Wertebereich -100 bis 100)
Amplitude der Anregung (nur bei Streckensprungantwort und PRBS-Anregung). Bei der geregelten Sprungantwort erfolgt die Anregung extern.

Abtastzeit T_s (Wertebereich 10ms bis 100s)
Gibt die zeitliche Auflösung der Abtastung an. Die Abtastrate bestimmt den untersuchten Frequenzbereich maßgeblich. Auch kann als Totzeit, aufgrund der Messung dieser in Abtastschritten, nur ein ganzzahliges Vielfaches dieses Parameters annehmen.

PRBS bits n (nur bei PRBS-Anregung; Wertebereich: 4 bis 10)
Gibt die Länge des PRBS-Schieberegisters an (siehe Abschnitt 5.3).

Faktor $\mathrm{upsample}$ (nur bei PRBS-Anregung; Wertebereich: 1 bis 1000)
Gibt an, nach wie vielen Abtastzyklen das Schieberegister zur PRBS-Anregung aktualisiert wird (siehe Abschnitt 5.3).

In Abbildung 5.6 ist eine beispielhafte Abfrage dieser Parameter dargestellt. Diese wird ausgeführt, falls keine oder ungültige Parameter beim Programmaufruf übergeben wurden.

```
C:\workspace\sysIdent\bin\Release>sysIdent
Device Instance (0 - 4194303, default: 0): 2098180
Output to System:
Object Type (AO=1, AVAL=2):  (1 - 2, default: 1): 2
Object Instance:  (1 - 4194303, default: 1): 3
Input from System:
Object Type (AI=0, AVAL=2):  (0 - 2, default: 0): 2
Object Instance:  (1 - 4194303, default: 1): 4
Please enter the following Parameters for the test signal:
Signal Type (Step = 0, PRBS = 1, Measure = 2): 1
Amplitude (-100 - 100, default: 30): 30
Sample Time Ts. (Should be set to approximately 5-10 times expected
system bandwidth.)
 (10 - 100000, default: 500): 500
```

Abbildung 5.6: Abfrage der Programmparameter vom Benutzer

Wird diese Art der Parameterabfrage genutzt, gibt das Programm anschließend folgenden Hinweis aus:

„- for step response analysis -
sysIdent device-instance object-type(receive)
object-instance(receive) object-type(transmit)
object-instance(transmit) sample_time(ms) test_type(=0)
amplitude

- OR for PRBS signal-
sysIdent device-instance object-type(receive)
object-instance(receive) object-type(transmit)
object-instance(transmit) sample_time(ms) test_type(=1)
amplitude PRBS-length upsample

- for pure measurement of in- and output -
sysIdent device-instance object-type(receive)
object-instance(receive) object-type(transmit)
object-instance(transmit) sample_time(ms) test_type(=2)“

Es ist also auch möglich, eine äquivalente Konfiguration des Programms durch einen Aufruf mit Parameterübergabe über eine Windows-Konsole zu erreichen. Der Befehl für das Beispiel aus Abbildung 5.6 lautet dazu:

```
sysIdent 2098180 2 3 2 4 500 1 30 4 4
```

Diese Schnittstelle beschleunigt die Benutzung des Programms bei häufiger Nutzung. Gleichzeitig kann jederzeit auf die schrittweise Abfrage zurückgegriffen werden, wenn auf die Übergabe von Parametern verzichtet wird.

Nach der Konfiguration wird die BACnet Verbindung hergestellt. Anschließend wird (in der Funktion „wait_steady()“, siehe Abschnitt 5.1.2) auf den Beginn der Messung gewartet. Die zugehörige Programmausgabe ist in Abbildung 5.7 dargestellt.

```
C:\workspace\sysIdent\bin\Release>sysIdent 2098180 2 4 2 3 500 1 30 4 4
Amount of samples increased to minimal supported PRMS length.
Trying to bind with device. Done!
Reading initial output... 30.0929
Output locked.
Wait till input is constant. Then press any key to continue.

                       Filtered
Input   Filtered       Slope[sec^-1]   % of Max Slope
 30.017  30.017          0.000                  nan
```

Abbildung 5.7: Bildschirmausgabe vor Beginn der Messung der Testsignale

Der Benutzer muss nun durch einen Tastendruck bestätigen, dass sich das System in Ruhe befindet. Danach beginnt die Anregung des Systems. Dabei wird zunächst die voraussichtliche Dauer der Messung ausgegeben (siehe Abbildung 5.8). Außerdem wird dort stets die aktuelle Abtastung der Regelgröße, sowie ihre (PT1-gefilterte) Änderung angegeben. Es sind nun keine weiteren Benutzereingaben mehr erforderlich. In Abbil-

```
PRMS bit count: 4  Samples: 60
Resulting duration: 30 s.
Note: Measurement can be aborted by striking ESC.
Sample: 28, Input: 20.182 Slope:  0.3 sec^-1 (18.7% of Max)
```

Abbildung 5.8: Bildschirmausgabe während der Aufzeichnung der Testsignale

dung 5.9 sind die folgenden Programmausgaben dargestellt. Im dargestellten Beispiel wird eine, auf dem Automationscontroller simulierte, PT2-Strecke mit den Zeitkonstanten $\boldsymbol{T} = [10s,\ 35s]$ und einer Verstärkung von $K_S = 1$ identifiziert. Diese Streckenparameter lassen sich in Näherung auch im ausgegebenen Identifikationsergebnis unter „System Fitting“ wiederfinden.

```
Note: Measurement can be aborted by stroking ESC.
Sample: 59, Input: 33.155 Slope: -0.2 sec^-1 (58.0% of Max)
Output relinguished!
Sample: 60, Input: 32.98 Slope: -0.23 sec^-1 (66.1% of Max)
All requests sent. Listening...

Received all requests!
Writing to file 'out.csv'...success!
Writing to file 'in.csv'...success!


+++++++Timing Analysis+++++++
(All measurements in ms)
receive_reqs:

Min     Max     Average SDev    Timing Violations
------------------------------------------------
500     515     507     4.35    0/60 Requests
send_reqs:

Min     Max     Average SDev    Timing Violations
------------------------------------------------
501     531     513     10.4    0/8 Requests


+++++++System Fitting+++++++
Writing fitting data to 'data.csv'...success!
Tt=0.5 sec, n= 2
log10(Iv)=-6.29559
T=[34.6680,9.7747] sec
K_stat = 0.999422


Recommended Controller Parameters:
Damping Kp      Tn[sec] Expected overshoot[%]
0.5     3.168   34.67   16.3
0.7     1.661   34.67   4.6
1       0.8551  34.67   0
1.5     0.3838  34.67   nan

Normal exit of Program.
```

Abbildung 5.9: Bildschirmausgabe nach Abschluss der Anregung

6 Auswertung

In diesem Kapitel wird die Leistungsfähigkeit der erstellten Lösung zur Systemidentifikation und Reglersynthese untersucht. Die Untersuchung wird hierbei in zwei Teile unterteilt:

1. eine ausführliche Untersuchung an einer Regelstrecke unter Laborbedingungen
2. eine kurze Evaluation an realen Regelstrecken

Für die Auswertung wird hier jeweils lediglich das Führungsverhalten der Regelstrecken untersucht. Eine Betrachtung des Störverhaltens wäre zwar auch interessant, an realen Strecken lassen sich Störungen allerdings nur schwer erzeugen. Aus diesem Grunde wird hier auf die Untersuchung dieser verzichtet.

Hinweis: Alle Messreihen in diesem Kapitel zeigen jeweils Differenzsignale zu einem vorher ermittelten Arbeitspunkt, um den das System linearisiert werden soll (siehe Abschnitt 3.1).

6.1 Messung an einer Labor-Regelstrecke

Um das Reglersyntheseprogramm „sysIdent“ testen zu können, wird zunächst eine Regelstrecke aus einem regelungstechnischen Labor der HAW Hamburg untersucht. Diese Strecke eignet sich gut als Einstieg, da sie eine hohe Verfügbarkeit an Messzeit bietet, und dort anders als bei einer realen Anlage keine Beeinträchtigung eventueller Nutzer zu erwarten ist. Außerdem gibt es an dieser Regelstrecke kaum unvorhergesehene Störungen, da der Systemkontext „steril“ gehalten ist. Dieses System bietet damit bessere Rahmenbedingungen, als sie an realen Strecken der Gebäudeautomation zu erwarten sind.

Bei der verwendeten Regelstrecke handelt es sich um ein Rohr mit einem konstant angesteuerten Ventilator, der einen Luftstrom durch dieses Rohr verursacht. Dieser Luftstrom wird mit einem zu regelnden elektrischen Heizelement geheizt. Am Ende des Rohrs wird mit einer PT-100 Messsonde die Temperatur gemessen. Diese Temperatur wird

über mehrere Messumformer auf ein $0 \ldots 10\,\mathrm{V}$ Signal umgesetzt. Im Folgenden wird diese Spannungsgröße als Regelgröße genutzt. Auf eine Umsetzung in einen Temperaturwert wird verzichtet, da diese keinen Mehrwert bei der Beurteilung des Identifikations- und Reglersyntheseverfahrens liefert. Die Stellgröße ist im Folgenden die Heizleistung des elektrischen Heizelements. Diese wird ebenfalls über mehrere Umsetzer mit einem $0 \ldots 10\,\mathrm{V}$ Signal beeinflusst. Stell- und Regelgröße sind an einen *PXC12-E.D* Gebäudeautomationscontroller der Firma Siemens angeschlossen und werden von dort geregelt.

6.1.1 Verwendete Anregungen

Tabelle 6.1 zeigt eine Übersicht der für diesen Bericht aufgenommenen Messreihen. Die Namen der Messreihen geben dabei die Art der Anregung an. „Step“ im Namen steht dabei für „Streckensprungantwort“, „PRBS“ für eine PRBS-Anregung und „Measure“ für eine externe Anregung über einen Führungsgrößensprung. Die Nummerierung der Messreihen ist hier willkürlich, diese wird aber aus Gründen der Nachverfolgbarkeit der aufgenommenen Daten so beibehalten. Für jede Messreihe wird eine separate Identifikation des Systems und eine Reglersynthese durchgeführt. Die zeitlichen Verläufe der Stell- und Regelgröße, sowie eine Simulation der Regelgröße finden sich im Anhang in den Abbildungen A.1 bis A.8. Die Simulation der Regelgröße erfolgt dabei auf Basis der s-Übertragungsfunktion des identifizierten Systems, der gemessenen Stellgröße $u(t)$ und dem Vektor der Messzeitpunkte. Die MATLAB® Funktion „lsim“ wird zur Durchführung aller Simulationen in diesem Kapitel verwendet.

Die Messungen werden um einen Arbeitspunkt

$$u_0 \approx 5{,}0\,\mathrm{V} \tag{6.1}$$

$$y_0 \approx 3{,}7\,\mathrm{V} \tag{6.2}$$

aufgenommen. Durch Drifterscheinungen kommt es zu leichten Abweichungen der Stellgröße von diesem Arbeitspunkt. Die Regelgröße y wird mit Hilfe eines Reglers zwischen den Messungen jedoch wieder auf den obigen Wert von u_0 eingeregelt. Im Folgenden werden die Signale $u(t)$ und $y(t)$ für Stell- und Regelgröße jeweils als Differenz zum Arbeitspunkt am Anfang der entsprechenden Messung angegeben.

In Tabelle 6.1 sind die Amplitude der Stellgröße $\hat{u}$ und die Amplitude der Regelgröße $\hat{y}$ für die verschiedenen Messungen angegeben, sowie weitere Parameter der Abtastung und Anregung. In der Spalte „Flanke“ wird die Richtung der Regelgrößenänderung an-

Name	$\hat{u}$[V]	$\hat{y}$[V]	T_s[msek]	Messdauer t [sek]	n_{PRBS}	upsample	Flanke
Step1	1,50	1,18	500	112,0	–	–	steigend
Step2	1,50	1,24	500	124,5	–	–	fallend
PRBS1	1,50	1,22	500	450,0	4	60	–
PRBS2	1,50	0,31	500	255,0	8	2	–
PRBS4	1,50	0,73	500	112,5	4	15	–
Measure3	3,36	0,74	500	85,0	–	–	steigend
Measure4	3,41	0,74	500	75,0	–	–	fallend
Measure6	2,73	0,68	250	66,25	–	–	fallend

Tabelle 6.1: Parameter der Messreihen für die Labor-Regelstrecke für verschiedene Anregungen. Die Art der Anregung ergibt sich aus dem Namen. Die Faktoren n_{PRBS} und upsample sind die entsprechenden Parameter einer PRBS Anregung

gegeben. Bei den PRBS wird dieses Feld frei gehalten, da die Signale hier in Näherung symmetrisch um den Arbeitspunkt arbeiten.

Es wird mit zwei Messreihen von Streckensprungantworten (in positiver und negativer Richtung) begonnen. Diese werden absichtlich bereits kurz vor dem vollständigen Erreichen des neuen Endwertes abgebrochen, da der deterministische Signalanteil zum Ende hin im Vergleich zum Signalrauschen immer kleiner wird. Dies ist durch die abnehmende Steigung der Übergangsfunktion begründet. Bei einer „klassischen“ Bestimmung z.B. der Streckenverstärkung wäre eine längere Messung notwendig.

Anschließend wird eine sehr „langsame“ PRBS-Anregung verwendet („Messung PRBS1“). Durch den Faktor $\mathrm{upsample} = 60$ in Kombination mit einer Abtastrate von $T_S = 0{,}5\,\mathrm{sek}$ ergibt sich ein Aktualisierungsintervall des PRBS-Generators von

$$\begin{aligned} T_{\mathrm{PRBS}} &= T_S \cdot \mathrm{upsample} = 0{,}5\,\mathrm{sek} \cdot 60 \\ &= 30\,\mathrm{sek} \end{aligned} \tag{6.3}$$

Dies liegt in der Größenordnung der (aus den Streckensprungantworten abschätzbaren) dominierenden Zeitkonstante des Systems. Die dominierende Zeitkonstante des Systems bestimmt maßgeblich die Bandbreite desselben (Bohn und Unbehauen, 2016, S. 33) und wird daher als Referenzgröße gewählt. Wie aus der Tabelle ersichtlich, wird die Messdauer selbst bei einer kurzen PRBS-Sequenz durch einen derart hohen Faktor erheblich länger als etwa eine einfache Streckensprungantwort. Aus diesem Grunde wird

eine weitere Messung („PRBS4“) mit der gleichen PRBS-Sequenz und einem Faktor $\mathrm{upsample} = 15$ durchgeführt, was zu einer ähnlichen Messdauer wie bei den Streckensprungantworten führt. Als letzte PRBS Messung („PRBS2“) wird eine PRBS-Sequenz mit sehr schnellen Wechseln im Stellgrößensignal gewählt, um das Verhalten der Identifikation auch unter diesen Bedingungen zu testen.

Schlussendlich werden drei verschiedene geregelte Sprungantworten aufgenommen. Dazu wird ein PI-Regler eingesetzt. Dieser wird anhand der vorherigen Versuche auf eine Dämpfung des Führungsübertragungsverhaltens von $D = 0,5$ parametriert. Dazu werden willkürlich die in der Messreihe „Step1“ bestimmten Reglerparameter $K_P = 4,36$ und $T_n = 30\,\mathrm{sek}$ (hier gerundet auf eine ganze Sekunde, da der verwendete Controller keine kleineren Werte unterstützt) verwendet. Die notwendigen Sollwertsprünge werden auf dem Automationscontroller durch eine externe Software ausgelöst. Der Sollwertsprung hat dabei jeweils eine Amplitude von $\hat{w} = 0{,}6\,\mathrm{V}$.

6.1.2 Vergleich der identifizierten Systeme

In Tabelle 6.2 sind als Auswahl aus den in den Versuchsreihen identifizierten Parametern die Streckenverstärkung K_s, die dominierende Zeitkonstante T_1, die nächstgrößere Zeitkonstante T_2, die Totzeit T_t und die Modellordnung n angegeben. Es werden jeweils n-Zeitkonstanten für die Systeme ermittelt. In der Tabelle sind davon jeweils nur die beiden größten dargestellt. Alle weiteren Zeitkonstanten haben einen Wert $< 1\,\mathrm{sek}$ und sind aufgrund mangelnder Signifikanz hier nicht aufgelistet.

	K_s	T_1 [sek]	T_2 [sek]	T_t [sek]	n
Step1	0,81	30,13	2,69	3,00	3
Step2	0,84	30,29	2,54	3,50	3
PRBS1	0,79	27,39	4,13	2,50	4
PRBS2	0,80	22,49	6,49	2,00	4
PRBS4	0,81	25,21	5,56	2,00	2
Measure3	0,92	31,19	4,29	1,50	3
Measure4	0,68	20,53	4,92	2,00	3
Measure6	0,74	26,95	3,42	2,25	3

Tabelle 6.2: Auswahl ermittelter Streckenparameter für die Labor-Regelstrecke bei verschiedenen Anregungen

In der Tabelle 6.2 wird deutlich, dass es selbst unter Laborbedingungen eine erhebliche Streuung der Ergebnisse gibt. Ein entscheidender Parameter ist etwa die dominierende

Zeitkonstante T_1, die direkt als Parameter in den synthetisierten Regler eingeht. Für die gelisteten Messreihen ergibt sich hier dafür ein Mittelwert von

$$T_{1,\text{mean}} = 26{,}8\,\text{sek} \tag{6.4}$$

und eine Standardabweichung von

$$\sigma_{T_1} = 3{,}84\,\text{sek}, \tag{6.5}$$

was $14{,}3\%$ von $T_{1,\text{mean}}$ entspricht. Ein ähnliches Bild ergibt sich für die Streckenverstärkung K_s:

$$K_{\text{s,mean}} = 0{,}799 \tag{6.6}$$

$$\sigma_{K_s} = 0{,}0713\,\text{sek} \tag{6.7}$$

Für diesen Parameter beträgt die Standardabweichung σ_{K_s} $8{,}92\%$ des Mittelwertes $K_{\text{s,mean}}$. Auch die ermittelte Totzeit T_t und die Systemordnung n schwanken von Messung zu Messung nicht unerheblich.

Da alle ermittelten Parameter in den jeweiligen Übertragungsfunktionen zusammenwirken, ist es schwierig, die Ergebnisse anhand dieser zu vergleichen. Aus diesem Grunde wird zusätzlich für alle Systeme eine Sprungantwort simuliert. Diese Simulationen sind in Abbildung 6.1 dargestellt.

Das Zeitverhalten der unterschiedlichen Systemmodelle ist für kleine Zeiten $t < 20\,\text{sek}$ (bis zum Erreichen eines gedachten Wendepunkts) nahezu identisch. Dieser Teil der Sprungantwort wird maßgeblich durch die kleine Zeitkonstante T_2 und die Totzeit T_t bestimmt. In Tabelle 6.2 ist zu sehen, dass die Kombination der Parameter hier durchaus variiert, in der Simulation sind die Unterschiede jedoch marginal.

Für größere Zeiten nimmt die Streuung jedoch zu. In diesem Bereich schwingen sich die normierten Sprungantworten auf den Wert der statischen Streckenverstärkung ein. Bei der Betrachtung der unterschiedlichen Streckenverstärkungen K_s in Tabelle 6.2 ist zu erkennen, dass die Ausreißer im Bild genau den Messreihen der drei geregelten Sprungantworten („Measure3“, „Measure4“, „Measure6“) entspringen. Dies sind genau die drei Sprungantworten, die eine Abweichung vom Mittelwert der Streckenverstärkung von $\Delta K_s > 0{,}05$ aufweisen. Allerdings ist bei diesen Sprungantworten gemäß Tabelle 6.1 auch die Amplitude des Eingangssignals $\hat{u}$ deutlich größer. Die Stellgröße entfernt sich also weiter vom Arbeitspunkt. Eine mögliche Erklärung für die Abweichungen der berechneten Streckenverstärkung ist, dass die Abweichung vom Arbeitspunkt so groß ist, dass Nichtlinearitäten der Strecke hier zum Tragen kommen.

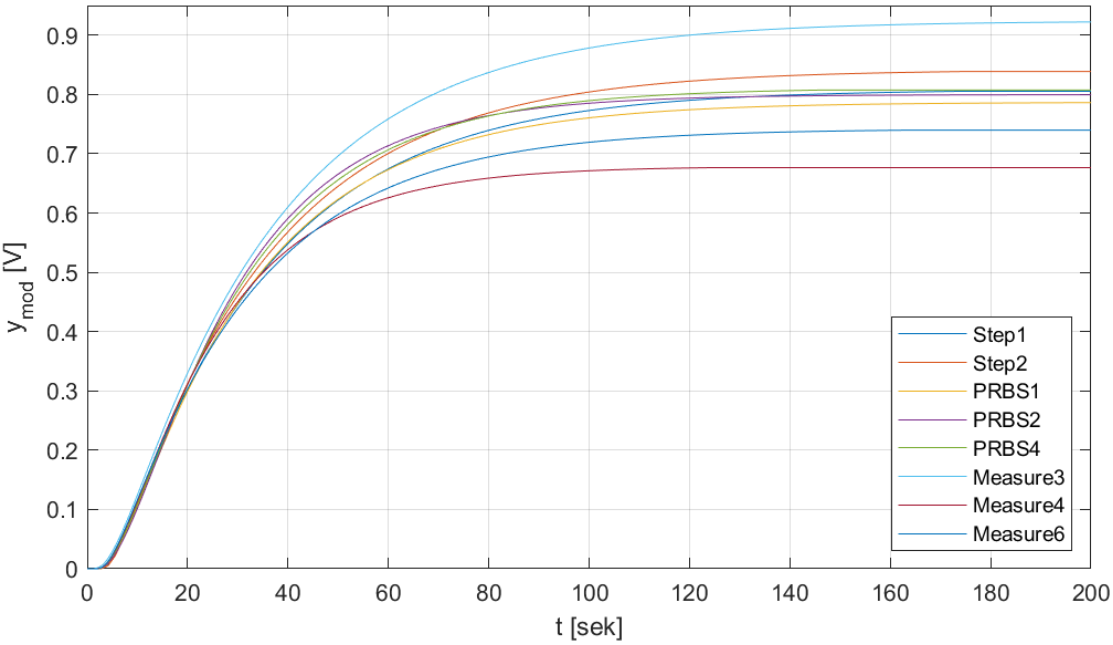

Abbildung 6.1: Streckensprungantworten der identifizierten Systemmodelle der Labor-Regelstrecke, simuliert mit einer Amplitude von $\hat{u} = 1\,\mathrm{V}$

Für diese These spricht auch, dass bei der Anregung mit einer steigenden Flanke der Führungsgröße w (Messreihe „Measure3“) eine größere Streckenverstärkung als der Mittelwert $K_{\mathrm{s,mean}}$ ermittelt wird, während die beiden Messreihen mit einem fallenden Sprung der Führungsgröße w („Measure4“, „Measure6“) einen geringeren Wert aufweisen. Die Streckenverstärkung scheint hier also abhängig von der Entfernung vom Arbeitspunkt zu sein.

6.1.3 Vergleich der Reglerparameter

Zum Vergleich der Reglersyntheseergebnisse werden jeweils die Reglerparameter T_n und K_P für eine Dämpfung von $D = 0{,}5$ bestimmt. Dieser Dämpfungswert eignet sich für die Analyse der erzielten Ergebnisse, da z.B. anhand der relativ großen Überschwingweite gut evaluiert werden kann, ob sich der gewünschte Verlauf des Einschwingvorgangs einstellt, ob also die in Kapitel 3 getroffenen Annahmen und Vereinfachungen zulässig sind. Die ermittelten Parameter sind in Tabelle 6.3, sowie graphisch in Abbildung 6.2 dargestellt.

	K_P	T_n [sek]
Step1	4,67	30,13
Step2	4,36	30,29
PRBS1	3,95	27,39
PRBS2	2,61	22,49
PRBS4	3,30	25,21
Measure3	4,48	31,19
Measure4	3,41	20,53
Measure6	4,90	26,95

Tabelle 6.3: Ermittelte Reglerparameter für die Labor-Regelstrecke bei einer gewünschten Dämpfung von $D = 0{,}5$ für verschiedene Anregungen

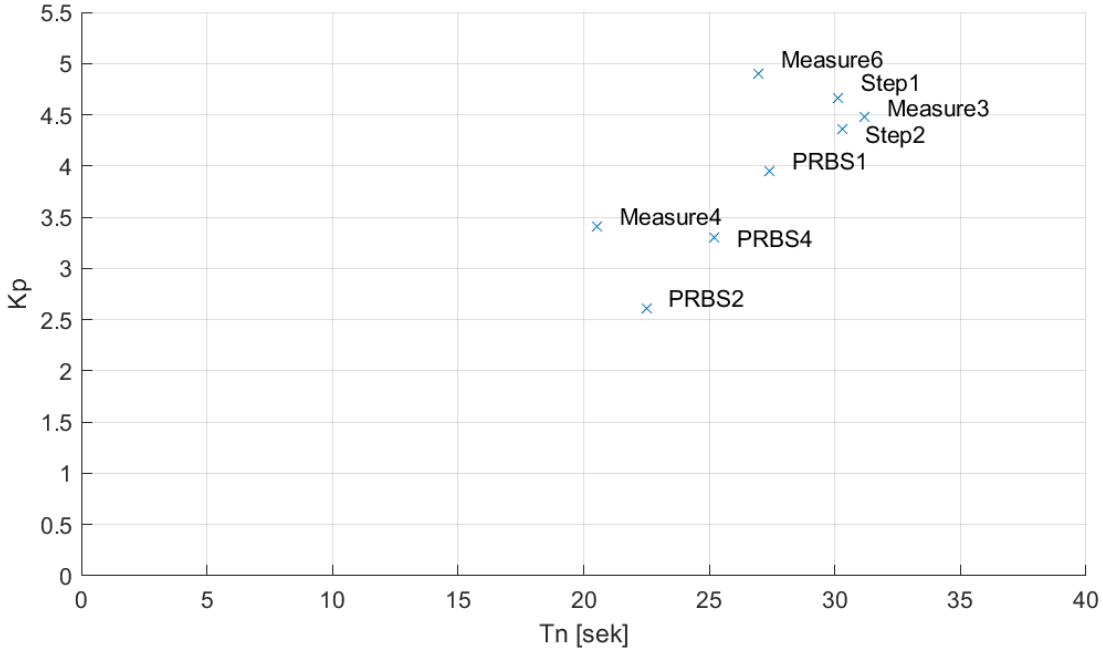

Abbildung 6.2: Ermittelte Reglerparameter für die Labor-Regelstrecke bei einer gewünschten Dämpfung von $D = 0{,}5$ für verschiedene Anregungen

Die Reglerverstärkung K_P bestimmt hierbei den P-Anteil des Reglers, über die Nachstellzeit T_n kann der I-Anteil des Reglers beeinflusst werden. Dabei steht eine kleine Nachstellzeit für ein „starkes“ I-Verhalten und eine große Nachstellzeit für ein langsames Integrieren der Regelabweichung. Qualitativ lässt sich in Abbildung 6.2 feststellen, dass bei einem großen P-Anteil (großer Wert von K_P) auch eine große Nachstellzeit, also ein kleiner I-Anteil für den Regler gewählt wird. Die große Streuung in T_1, und damit in T_n, wird also mutmaßlich zumindest teilweise von den unterschiedlich gewählten Reglerverstärkungen K_P kompensiert. Um diese These zu prüfen, wird nun der geschlossene Regelkreis (siehe Abbildung 6.3) untersucht.

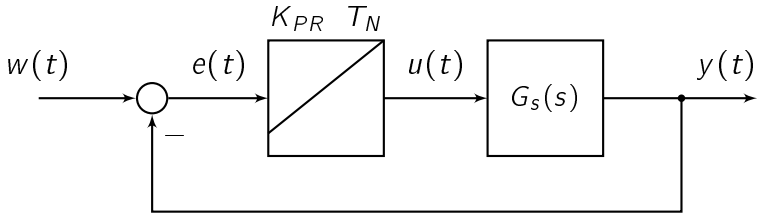

Abbildung 6.3: Signalflussdiagramm des geschlossenen Regelkreises

Aufgrund der Rückkopplung (Frey und Bossert, 2008, S.139) der Regelgröße ergibt sich folgende Übertragungsfunktion

$$G_w(s) = \frac{G_R \cdot G_S}{1 + G_R \cdot G_S} \tag{6.8}$$

für das Führungsverhalten. Mit dieser Beziehung kann das Führungsverhalten des Systems unmittelbar in MATLAB® simuliert werden. Die simulierten Führungssprungantworten sind in Abbildung 6.4 dargestellt. Tabelle 6.4 enthält die daraus bestimmten Parameter, maximales Überschwingen $e_{\max}(D)$, Ausregelzeit für ein Toleranzband von $t_{2\%}$ und die Anstiegszeit $T_{A,50}$. Diese Parameter wurden unter Abschnitt 4.1 definiert. Dort wurde auch in Tabelle 4.1 bei einer Dämpfung von $D = 0{,}5$ der erwartete Wert für das Überschwingen mit $e_{\max} = 16{,}3\,\%$ angegeben. Dieser Wert für das Überschwingen wird bei allen Simulationen in guter Näherung erreicht. Das Einschwingverhalten stimmt auch qualitativ mit der Erwartung überein. So gibt es einen Überschwinger in der erwarteten Höhe, anschließend einen kleinen Unterschwinger und dann nur noch ein leichtes Schwingen der Regelgröße. Daraus kann geschlossen werden, dass der Programmteil für die Reglersynthese hier gute Ergebnisse liefert, da in der Simulation das gewünschte Schwingverhalten für alle Streckenmodelle erreicht werden konnte.
In den Messreihen zur geregelten Sprungantwort („Measure3“, „Measure4“, „Measure6“), wäre bei der verwendeten Amplitude der Stellgröße $\hat{w} = 0{,}6\,\mathrm{V}$ ein Überschwingen

auf einen maximalen Wert von

$$\hat{y} = (e_{\max} + 1) \cdot \hat{w} = 116{,}3\% \cdot 0{,}6\text{V} = 0{,}69\text{V} \tag{6.9}$$

zu erwarten gewesen. Laut Tabelle 6.1 wurden Amplituden $\hat{y} = 0{,}68 \ldots 0{,}74$ gemessen. Es wird also auch hier, bei einem willkürlich aus einer der Identifikationen ausgewähltem Parametersatz (siehe oben), ein Überschwingen in der gewünschten Höhe erreicht. In den Abbildungen der Messreihen im Anhang (Abbildung A.6 bis A.8) ist zu sehen, dass der qualitative Verlauf hier ebenfalls erreicht wird.

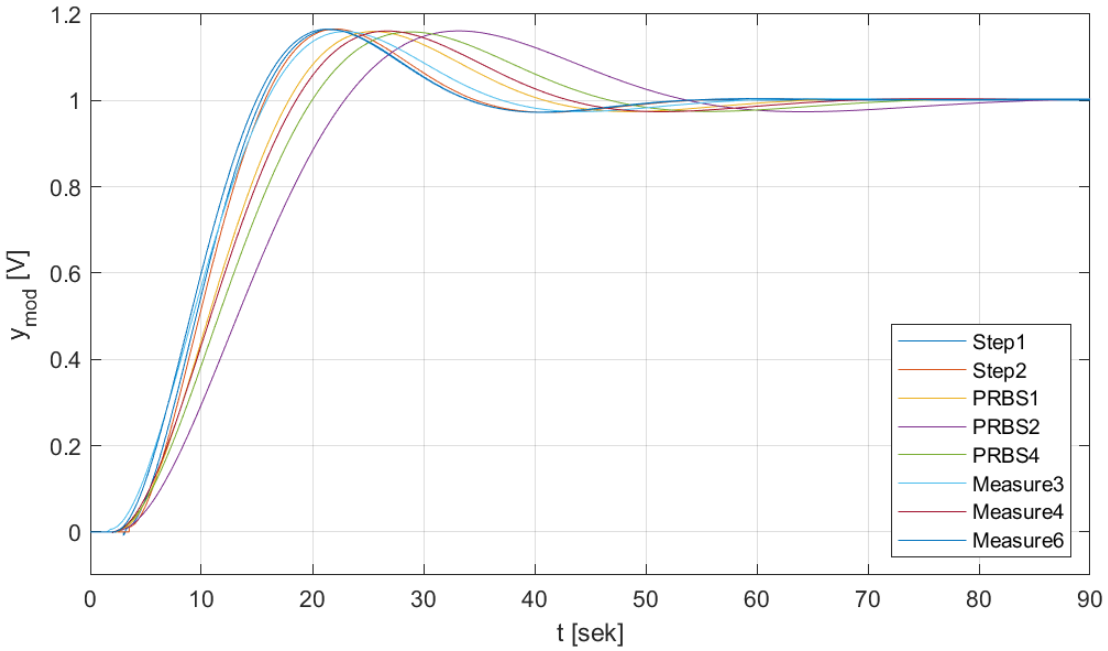

Abbildung 6.4: Führungssprungantworten der Labor-Regelstrecke mit Reglerparametern für eine Dämpfung D=0.5, simuliert mit einer Amplitude von $\hat{u} = 1\,\text{V}$

In der Simulation unterscheidet sich der zeitliche Horizont des Einschwingvorgangs jedoch je nach Streckenmodell erheblich. Dort kann die Messreihe „PRBS2“ mit einer sehr großen Ausregelzeit $t_{2\%}$ als Ausreißer nach oben identifiziert werden, was auf eine mangelhafte Systemidentifikation für diese Messung schließen lässt. Die sehr schnelle Anregung der Regelstrecke führt bei dieser Messreihe zu einer starken Dämpfung des Eingangssignals, was sich nicht zuletzt in der kleinen Ausgangsamplitude $\hat{y}$ (siehe Tabelle 6.1) ausdrückt. Selbst wenn dieser Ausreißer nicht berücksichtigt wird, bleibt eine erhebliche Schwankung des Zeitverhaltens bestehen.

	$T_{A,50}$ [sek]	$t_{2\%}$ [sek]	$e_{max}(D)$ [%]
Step1	9,53	45,20	16,29
Step2	9,84	46,82	16,21
PRBS1	10,68	55,38	15,53
PRBS2	13,27	73,46	15,65
PRBS4	11,56	62,49	15,65
Measure3	9,26	50,44	15,51
Measure4	10,85	57,91	15,84
Measure6	9,01	45,97	16,28

Tabelle 6.4: Kenngrößen der simulierten Führungssprungantworten

6.2 Strukturprüfung

In diesem Abschnitt soll die Ermittlung der Modellordnung n und der Totzeit n_d nach Abschnitt 3.3.2 analysiert werden. Wie in Tabelle 6.2 dargelegt, wurden für die verwendete Beispielregelstrecke Parameter von $n = 2 \ldots 4$ ermittelt. Da aber (wie oben erwähnt) neben den beiden größten Zeitkonstanten T_1 und T_2 nur noch sehr kleine Zeitkonstanten identifiziert ist, besteht der Verdacht, dass die Strecke bei den Identifikationen mit einer Modellordnung $n > 2$ übermodelliert sind, dass also mehr Parameter, als für eine gute Beschreibung des Systems notwendig, bestimmt werden. Diese Hypothese soll anhand der Messreihe „PRBS1“ überprüft werden. Diese Messreihe wird dafür ausgewählt, da diese (zusammen mit anderen) die höchste Modellordnung $n = 4$ aufweist. Trotzdem liegen die identifizierten Parameter hier im mittleren Bereich der ermittelten Parametersätze aller in Tabelle 6.2 aufgelisteten Messreihen, was auf eine hohe Güte der Identifikation schließen lässt.

In Abbildung 6.5 ist das Ausgangssignal der Messreihe „PRBS1“ nochmals dargestellt. Zusätzlich sind dort die simulierten Ausgangssignale für die ermittelte Totzeit von $n_d = 5$ (bzw. $T_t = n_d \cdot T_s = 2.5\,\text{sek}$) und die Modellordnungen $n = 1$ und $n = 2$ dargestellt. Beim Systemmodell erster Ordnung sind dort noch relativ große Abweichungen sichtbar. Beim Modell zweiter Ordnung sind die Linien für die Simulation und die Messung von y nahezu deckungsgleich. Das System kann also mit dem System zweiter Ordnung deutlich besser beschrieben werden. Wie eingangs vermutet, ergibt sich bei höheren Systemordnungen jedoch kaum eine Veränderung des Ausgangssignals. Auf das Darstellen entsprechender Simulationen in Abbildung 6.5 wird aus Gründen der Übersichtlichkeit verzichtet. Die Änderungen in der Simulation sind so minimal, dass sie in dieser Abbildung ohnehin nicht sichtbar würden. Dennoch wird bei der Systemidentifikation über den F-Test eine Modellordnung $n = 4$ ausgewählt.

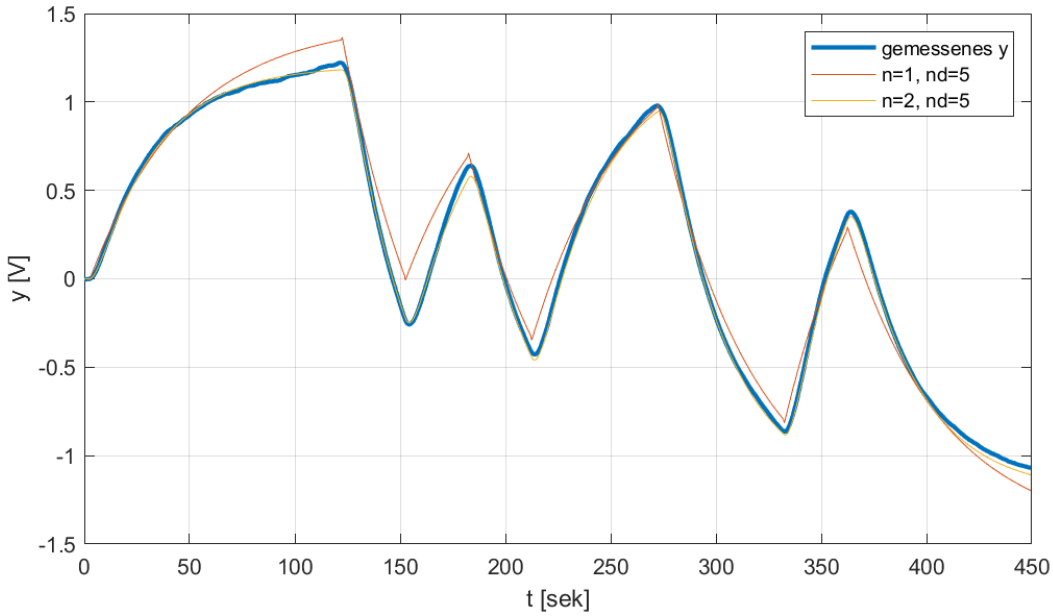

Abbildung 6.5: Ausgangssignal der Messreihe „PRBS1“ und simuliertes Ausgangssignal für verschiedene Modellordnungen n und der Totzeiten n_d

Um das Verhalten des gewählten Algorithmus genauer untersuchen zu können, sind in Tabelle 6.5 die signifikanten Parameter der einzelnen Iterationsschritte des Algorithmus zur Bestimmung der Modellordung aufgelistet. Wie in Abschnitt 3.3.2 beschrieben, vergleicht dieser jeweils zwei Systemmodelle der Ordnungen n_1 und n_2 über deren Gütefunktionale I_{v1} und I_{v2}. Aus diesen Parametern wird die Testgröße f gebildet und mit dem Schwellwert F verglichen. Ist die Testgröße f dabei kleiner als Schwellwert F, werden die zusätzlichen Parameter als nicht relevant angenommen.

Iteration	n_1	n_2	I_{v1}	I_{v2}	f	F	$f < F$
1	1	2	$1{,}33 \cdot 10^{-2}$	$8{,}21 \cdot 10^{-4}$	6769	19,49	Falsch
2	2	3	$8{,}21 \cdot 10^{-4}$	$5{,}98 \cdot 10^{-4}$	165,1	19,49	Falsch
3	3	4	$5{,}98 \cdot 10^{-4}$	$5{,}61 \cdot 10^{-4}$	29,83	19,49	Falsch
4	4	5	$5{,}61 \cdot 10^{-4}$	$5{,}48 \cdot 10^{-4}$	10,41	19,49	Wahr

Tabelle 6.5: Parameter des F-Tests beim Suchen der Modellordnung des Systemmodells zur Messreihe „PRBS1“. Jede Zeile entspricht einem Iterationsschritt des Algorithmus

Auch in dieser Tabelle ist zu sehen, dass die Modellordnung $n = 2$ ein deutlich besseres (um zwei Größenordnungen kleineres) Gütefunktional als die Modellordnung $n = 1$ lie-

fert. Außerdem wird deutlich, dass sich I_v hier auf einem sehr geringen Niveau befindet, wenn in Erwägung gezogen wird, dass es sich dabei um die halbe quadratische Summe der einzelnen Residuen handelt. Die Schwachstelle des verwendeten Verfahrens liegt hier darin, dass die Bestimmung der Testgröße nach

$$f(\hat{n}_2, \hat{n}_1) = \frac{I_{v1} - I_{v2}}{I_{v2}} \cdot \frac{\tilde{N} - 2\hat{n}_2}{2(\hat{n}_2 - \hat{n}_1)}$$

Gleichung (3.22) die Änderung des Gütefunktionals durch den Term

$$\frac{I_{v1} - I_{v2}}{I_{v2}} \tag{6.10}$$

ins Verhältnis zum absoluten Wert von I_v setzt. Die Systemordnung n wird durch den Algorithmus auch dann noch erhöht, wenn bereits ein sehr gutes Niveau der Übereinstimmung zwischen System und dessen Modell erreicht ist. Die absolute Höhe dieses Niveaus wird nicht weiter geprüft, sondern lediglich, ob eine weitere Verbesserung möglich ist.

Abschließend lässt sich sagen, dass die große identifizierte Modellordnung hier keinen Mehrwert bildet. Bei der Identifikation im Frequenzbereich wirken sich die zusätzlichen, sehr kleinen Zeitkonstanten kaum aus, da die resultierenden Eckfrequenzen sehr groß sind. Es kann hier also durchaus von einer Übermodellierung gesprochen werden. An dieser Stelle besteht noch Potential zur Verbesserung der Software.

6.3 Test der Identifikation an realen Strecken

Das erstellte Programm wird nun unter Praxisbedingungen untersucht. Dazu werden Versuche an verschiedenen Regelstrecken in einem Gebäudekomplex durchgeführt. Die Regelstrecken unterliegen hier realen Störungen. Außerdem wird das Programm in einem Netzwerk mit einigen zehntausend BACnet Datenpunkten und ca. einhundert PXC Automationscontrollern durchgeführt. Es wird also auch getestet, ob die angeforderten BACnet Daten auch unter realen Bedingungen ausreichend schnell und zuverlässig übertragen werden. Für diesen Bericht werden zwei Regelstrecken exemplarisch dargestellt.

6.3.1 Vorerhitzer-Register einer Lüftungsanlage

Bei der ersten Regelstrecke handelt es sich um ein Vorerhitzer-Register in einem Lüftungskanal einer raumlufttechnischen Anlage. Der Aufbau ist dabei ähnlich zu der Labormessstrecke aus dem vorherigen Abschnitt: Es gibt ein Heiz-Register, dem über ein Drosselventil Heißwasser zugeführt werden kann, um einen Luftstrom aufzuheizen. Das Ventil stellt dabei das Stellglied, der Durchfluss des Heizmediums die Regelgröße dar. Einige Meter hinter dem Heizregister wird dann die Lufttemperatur als Regelgröße gemessen. Bei dieser Regelstrecke gibt es jedoch eine Besonderheit: Es wird auf dem

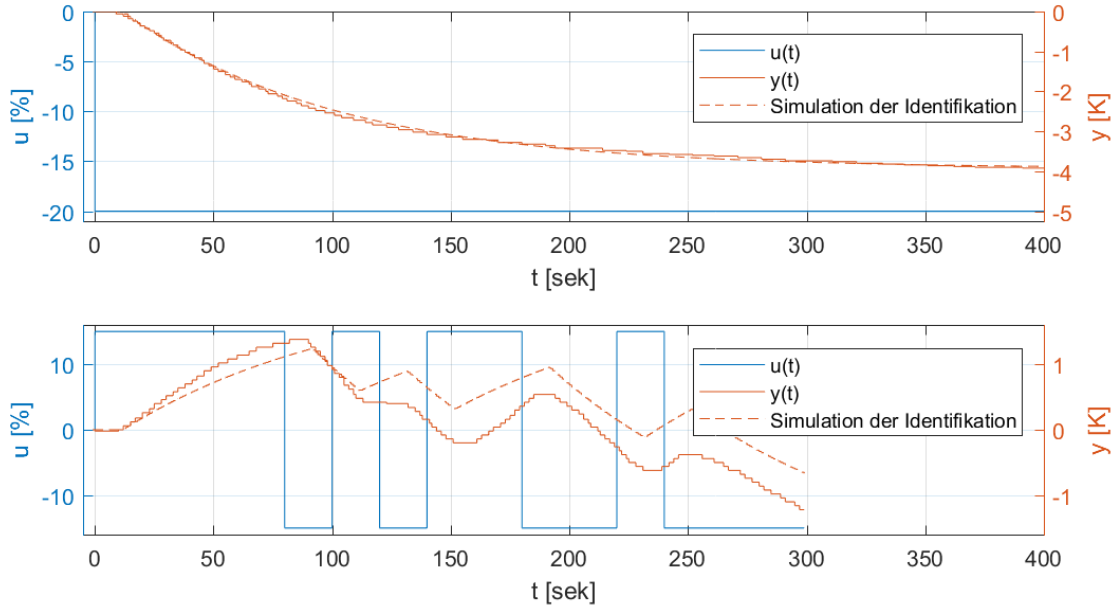

Abbildung 6.6: Messreihen und Simulation der Regelgröße mit einer sprungförmigen und einer PRBS-Anregung an einem Vorerhitzer-Register

Stellventil ein sogenannter „elektrohydraulischer Stellantrieb“ Typ *SKD60* der Firma Siemens eingesetzt. Dieser Antrieb hat die Eigenschaft, dass die Stellzeit für das Öffnen des Ventils (30 sek) doppelt so groß wie die Zeit für das Schließen (15 sek) ist (Siemens AG, 2018). Dieses Ventil stellt eine Nichtlinearität im System dar, weil ein Sprung in negativer Richtung durch die unterschiedlichen Stellzeiten zwangsläufig anders verläuft als ein positiver Sprung. Diese Strecke wird als negatives Beispiel für eine Identifikation in diesen Bericht aufgenommen. In Abbildung 6.7 ist zu sehen, dass das Streckenmodell bei einem Sprung gut identifiziert werden kann. Bei der PRBS-Anregung weicht das simulierte Streckenmodell jedoch stark von der Messung ab, was an der eingangs beschriebenen

Nichtlinearität liegt. Dennoch werden in beiden Messreihen ähnliche Zeitkonstanten ermittelt. Bei der ermittelten Streckenverstärkung beträgt die Abweichung der ermittelten Streckenverstärkung ΔK_s jedoch erhebliche 25% (siehe Tabelle 6.6).

Anregung	K_s [K/%]	T_1 [sek]	T_2 [sek]	T_t [sek]	n
sprungförmig	0,20	89,59	0,24	12	2
PRBS	0,15	96,17	–	12	1

Tabelle 6.6: Auswahl ermittelter Streckenparameter für das Vorerhitzer-Register einer Lüftungsanlage

6.3.2 Zulüfter einer Raumlüftungsanlage in einem Bürogebäude

Als zweite Strecke wird der Lüfter einer Raumlüftungsanlage in einem Bürogebäude untersucht. Die Stellgröße ist hier der Ausgang eines Frequenzumrichters, der einen Lüftermotor antreibt. Der Differenzdruck vom Lüftungskanal zum atmosphärischen Umgebungsdruck stellt dabei die Regelgröße dar. Dieser Druck treibt den Volumenstrom der Luft in ein angeschlossenes Netz aus Einzelräumen. Diese Strecke wird aufgrund ihrer hohen Geschwindigkeit ausgewählt, vor allem um die Leistungsfähigeit der implementierten BACnet Schnittstelle nachzuweisen. Die entsprechenden Messreihen befinden sich in Abbildung 6.7.

Tabelle 6.7 zeigt, dass bei diesem Versuch eine sehr gute Übereinstimmung der identifizierten Modelle besteht. Auch in Abbildung 6.7 ist jeweils eine sehr gute Übereinstimmung zwischen Messung und Simulation zu erkennen. Die Identifikation kann hier als erfolgreich angesehen werden.

	K_s [Pa/%]	T_1 [sek]	T_2 [sek]	T_t [sek]	n
sprungförmig	11,71	11,33	0,09	3,75	3
PRBS	11,76	10,52	0,09	3,5	3

Tabelle 6.7: Auswahl ermittelter Streckenparameter für den Zulüfter einer Raumlüftungsanlage

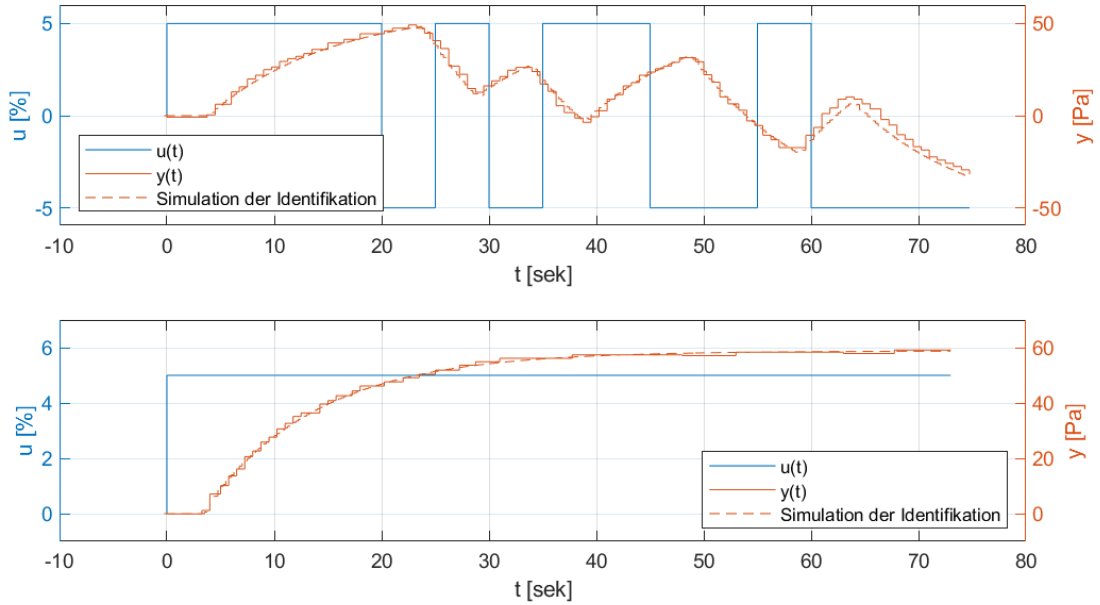

Abbildung 6.7: Messreihen und Simulation der Regelgröße mit einer PRBS- und einer sprungförmigen Anregung an einem Zulüfter einer Bürolüftungsanlage

7 Fazit und Ausblick

Um diese Arbeit zu erstellen, war die Bearbeitung eines breiten Themenfeldes notwendig. Es wurde aufgezeigt, auf welche Arten eine Regelstrecke in der Praxis angeregt werden kann, und wie ein anschließendes Identifikations- und Syntheseverfahren ablaufen kann. Die Ergebnisse wurden anhand drei verschiedener Strecken erfolgreich angewendet und verifiziert. Im Rahmen dieser Arbeit wurde damit die Erstellung eines praxistauglichen Reglersynthese-Tools erreicht. Durch den Einsatz der erstellten Software kann der Zeitaufwand für die Optimierung oder Inbetriebnahme einer Regelstrecke erheblich verringert werden. Das manuelle Aufnehmen und Auswerten von Sprungantworten, bzw. das empirische Ermitteln von Reglerparametern kann durch die Software nun weitestgehend ersetzt werden. Trotz vieler Möglichkeiten das erstellte Produkt weiter auszubauen, wurde hier ein solider Grundstein für weitere Arbeiten gelegt. Auf dieser Basis können in Zukunft weitere Verbesserungen und Erweiterungen entwickelt werden. Eine Auswahl wird im Folgenden dargelegt:

Verbesserung des Automatisierungsgrads

Eine automatisierte Identifikation und Reglersynthese wurde nicht vollständig erreicht. Der Benutzer muss das Syntheseverfahren zurzeit noch manuell parametrieren und je nach Anregungstyp eine Messung manuell überwachen und auch beenden. Zusätzlich wird in der jetzigen Version vom Benutzer verlangt, am Anfang der Messung zu bestätigen, dass sich das System in einer *Ruhelage* befindet. D.h., dass keine Änderungen der Zustandsgrößen des Systems auftreten (Unbehauen, 2007, S. 25). Ein Ansatz kann hier sein, dass die Änderung der Regelgröße überwacht wird und eine Messung automatisch gestartet bzw. gestoppt wird. Die Aufzeichnung der Messreihen könnte in Zukunft z.B. mit der maximal sicher erreichbaren Abtastfrequenz durchgeführt werden. Das Programm könnte anschließend automatisch ermitteln, ob ein nachträgliches Reduzieren der Abtastrate („downsampling“) sinnvoll ist.

Perspektivisch lassen sich auch Reglerparameter direkt in ein BACnet Regler-Object (siehe Kranz (2013), S. 230ff) schreiben. Dadurch wird die Bedienung der Software weiter vereinfacht.

Rekursive Lösung des Ausgleichsproblems

Im bisherigen Vorgehen wurde die Systemidentifikation erst nach Abschluss der Messung durchgeführt. Es ist allerdings möglich, die Streckenparameter mit einem rekursiven Lösungsverfahren bereits während der Aufnahme der Messreihen zu lösen (siehe Bohn und Unbehauen (2016) Kapitel „3.3.2 Rekursive Lösung“). Dabei werden Startwerte für die Streckenparameter angegeben, die über die Messdauer zum Identifikationsergebnis konvergieren. Dadurch kann eine ausreichende Messdauer bei der Identifikation durch die Änderung der Parameter ermittelt werden. Ändern sich diese kaum noch, kann die Messung und die Identifikation beendet werden (Bohn und Unbehauen, 2016, S. 508).

Verbesserung der Benutzerschnittstelle

In dieser Arbeit wurde der Fokus auf die Systemidentifikation und Reglersynthese gelegt. Die Benutzerschnittstelle wurde daher nur rudimentär ausgestaltet. Um die Bedienung der Software zu vereinfachen, wäre es z.B. möglich BACnet Objekte über deren Namen, statt über deren ID im Netzwerk zu finden. Dadurch kann das Risiko einer Fehlbedienung minimiert werden. Zusätzlich wäre es in Zukunft wünschenswert, eine graphische Benutzeroberfläche zu erstellen. Dort könnten auch simulierte Führungssprungantworten für verschiedene Reglerparameter dargestellt werden. Das würde zusätzlich die Implementierung eines Simulationsmoduls in der Software erfordern. So könnte das sich ergebende Zeitverhalten bereits vor dem Durchführen einer weiteren Messung begutachtet werden.

Validierung des Systemidentifikation- und Reglersyntheseverfahrens

In dieser Arbeit konnte eine Validierung der Ergebnisse nur stichprobenartig an einigen Regelstrecken durchgeführt werden. Eine breit angelegte Versuchsreihe zur Durchführung von Messreihen wäre hier wünschenswert, um die Leistungsfähigkeit des erstellten Programms zu überprüfen. Der in Abschnitt 2.2 erwähnte zeitliche Horizont der Zeitkonstanten konnte hier nicht vollständig ausgetestet werden. Der obere Bereich der möglichen Zeitkonstanten T_{max} wurde nicht untersucht. Für diesen Bereich wurden etwa Raumheizungsregelungen als mögliche Strecken anvisiert, die beim Erstellen der Arbeit aber aufgrund von Witterungsbedingungen nicht mit ausreichender Messzeit zur Verfügung standen.

Ein weiterer Punkt ist, dass die berechneten Reglerparameter bisher nicht durch eine Messung einer Sprungantwort mit dem neu ermittelten Parametersatz validiert werden. Auch dieser Schritt ließe sich in Zukunft in das Programm integrieren und automatisieren.

Literaturverzeichnis

[ALGLIB-Project 2018a] ALGLIB-PROJECT: *ALGLIB - C++ manual*. 2018. – URL http://www.alglib.net/translator/man/manual.cpp.html. – [Online; Zugriff 27.05.2018]

[ALGLIB-Project 2018b] ALGLIB-PROJECT: *ALGLIB - C++/C# numerical analysis library*. 2018. – URL http://www.alglib.net/. – [Online; Zugriff 27.05.2018]

[Ansorge und Oberle 2000] ANSORGE, Rainer ; OBERLE, Hans J.: *Mathematik für Ingenieure, Band 1, 3.Auflage*. WILEY-VCH Verlag, 2000. – ISBN 3-527-40309-4

[Bohn und Unbehauen 2016] BOHN, Christian ; UNBEHAUEN, Heinz: *Identifikation dynamischer Systeme*. Springer Vieweg, 2016. – ISBN 978-3-8348-1755-6

[cplusplus.com 2018] CPLUSPLUS.COM: *mutex - C++ Reference*. 2018. – URL http://www.cplusplus.com/reference/mutex/mutex/. – [Online; Zugriff 26.05.2018]

[Doxygen 2018] DOXYGEN: *Doxygen: Main Page*. 2018. – URL http://www.doxygen.org. – [Online; Zugriff 25.05.2018]

[Frey und Bossert 2008] FREY, Thomas ; BOSSERT, Martin: *Signal- und Systemtheorie*. Vieweg+Teubner Verlag, 2008. – ISBN 978-3-8351-0249-1

[von Grünigen 2014] GRÜNIGEN, Daniel C. von: *Digitale Signalverarbeitung*. Carl Hanser Verlag München, 2014. – ISBN 978-3-446-44079-1

[Karg 2017] KARG, Steve: *BACnet Stack*. 2017. – URL http://bacnet.sourceforge.net/. – [Online; Zugriff 26.05.2018]

[Kranz 2013] KRANZ, Hans R.: *BACnet Gebäudeautomation 1.12*. cci Dialog GmbH, 2013. – ISBN 978-3-922420-25-5

[Nithya u. a. 2015] NITHYA, R. ; PAVIYA, M. ; POORNIMADEVI.A1, A. ; SELVARAJ, Ravi: A performance comparision of low power LFSR structures. In: *International Journal of Recent Research in Science, Engineering and Technology* 1 (2015), Nr. 1, S. 23–35

[Otto 1989] OTTO, Michael: *Tafel der wichtigsten linearen Übertragungsglieder der Regelungstechnik*. 1989

[Siemens AG 2018] SIEMENS AG: *SKD60 Elektrohydraulischer Stellantrieb*. 2018. – URL `https://www.buildingtechnologies.siemens.com/bt/global/de/products/hlk-produkte/ventile-und-stellantriebe/stellantriebe-f%C3%BCr-hub--und-kombiventile/seiten/SKD60.aspx`. – [Online; Zugriff 30.05.2018]

[Siemens Schweiz AG 2015a] SIEMENS SCHWEIZ AG: *Desigo^{TM} Gebäudeautomationssystem 6.0*. Desigo V6, Systemdokumentation Engineering-Ausgabe, Dokument-ID: CM110664de. 2015

[Siemens Schweiz AG 2015b] SIEMENS SCHWEIZ AG: *Desigo^{TM} Vertiefungshandbuch Compound-Bibliotheken LED23*. Desigo V6, Systemdokumentation Engineering-Ausgabe, Dokument-ID: CM110748de. 2015

[Unbehauen 2007] UNBEHAUEN, Heinz: *Regelungstechnik I*. Vieweg & Sohn Verlag, 2007. – ISBN 978-3-8348-0230-9

[Wosnitza und Hilgerst 2012] WOSNITZA, Franz ; HILGERST, Hans G.: *Energieeffizienz und Energiemanagement*. Vieweg+Teubner Verlag, 2012. – ISBN 978-3-8348-1941-3

A Anhang

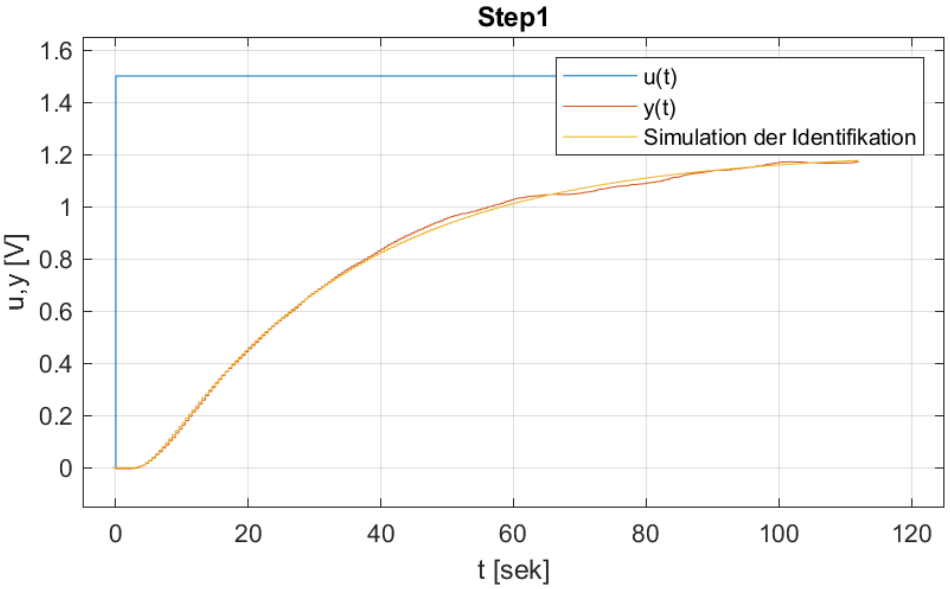

Abbildung A.1: Messreihe ’Step1’ Labor-Regelstrecke aus Abschnitt 6.1

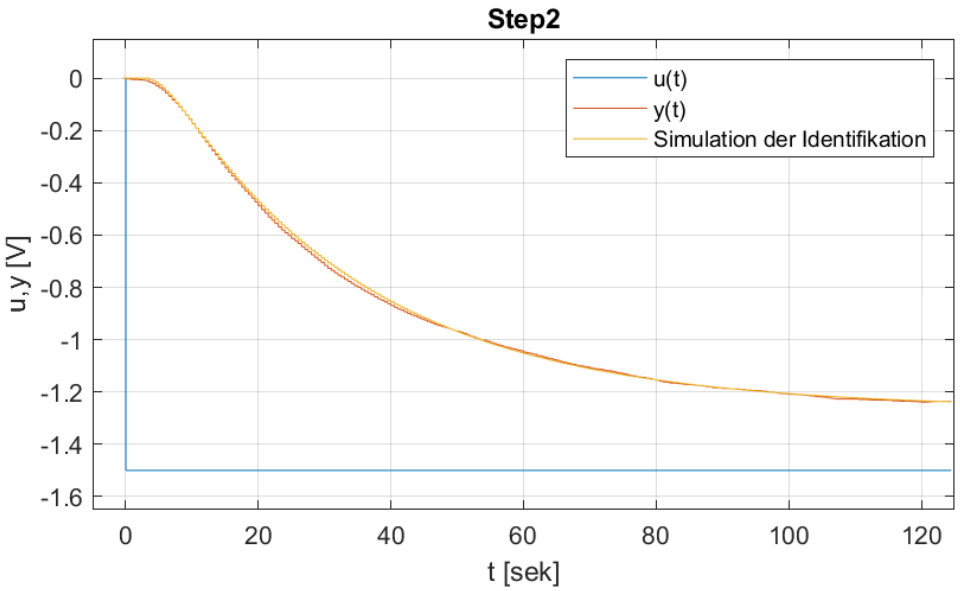

Abbildung A.2: Messreihe ’Step2’ an der Labor-Regelstrecke aus Abschnitt 6.1

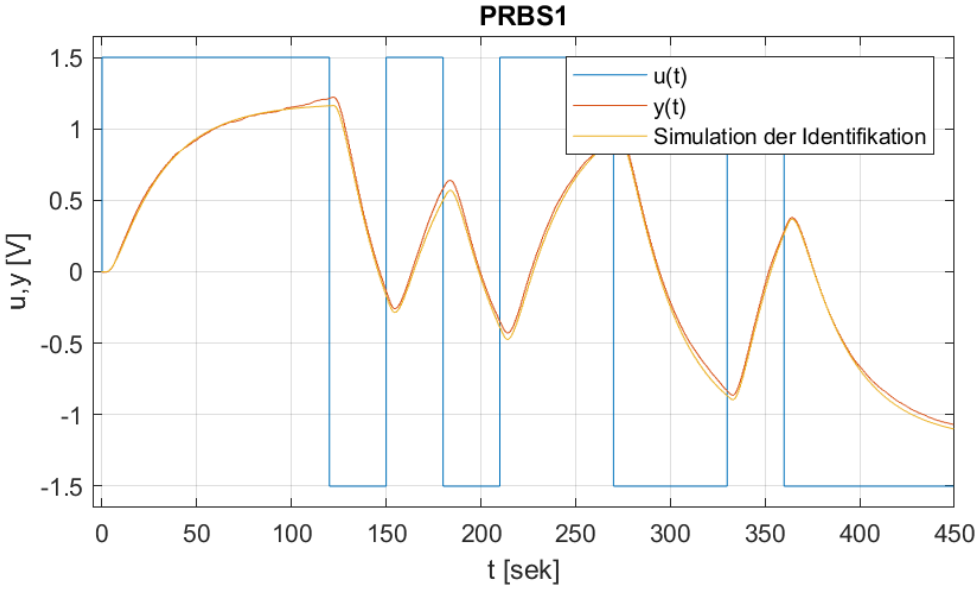

Abbildung A.3: Messreihe ’PRBS1’ an der Labor-Regelstrecke aus Abschnitt 6.1

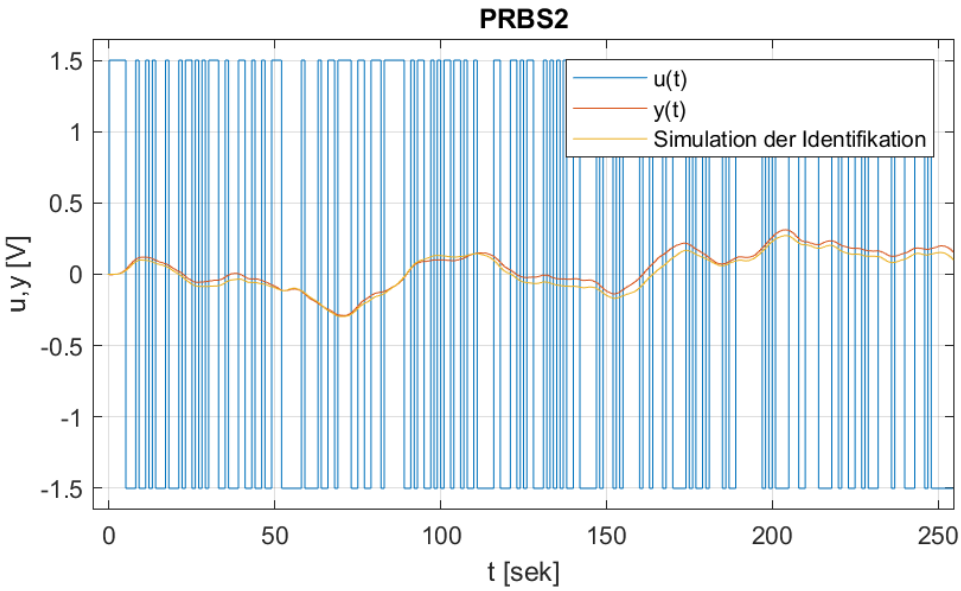

Abbildung A.4: Messreihe ’PRBS2’ an der Labor-Regelstrecke aus Abschnitt 6.1

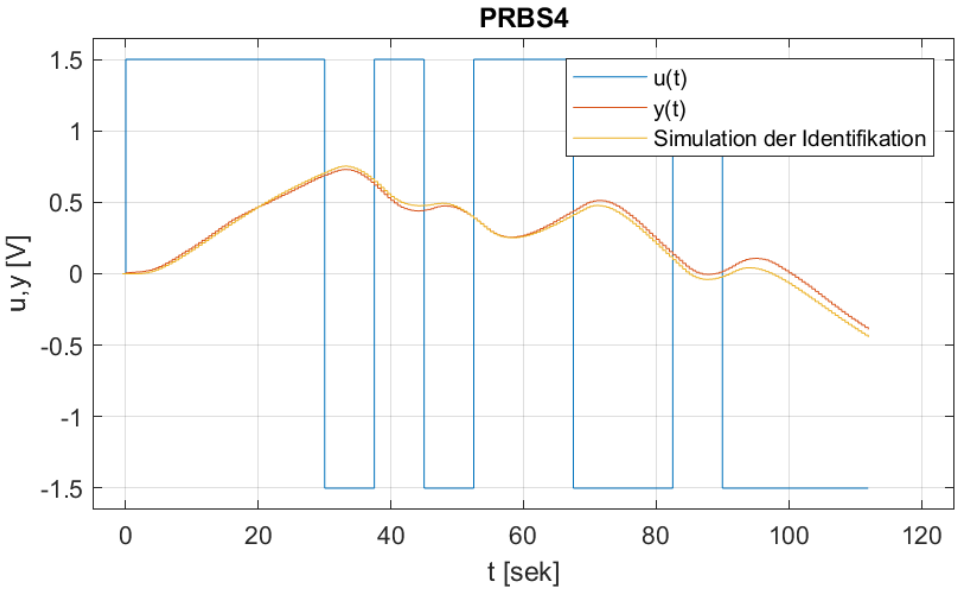

Abbildung A.5: Messreihe ’PRBS4’ an der Labor-Regelstrecke aus Abschnitt 6.1

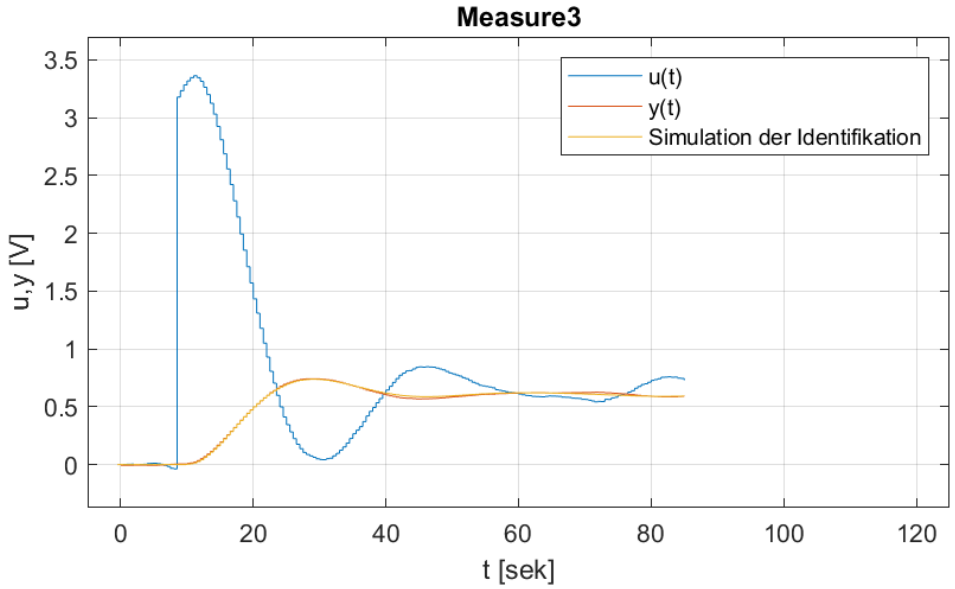

Abbildung A.6: Messreihe ’Measure3’ an der Labor-Regelstrecke aus Abschnitt 6.1

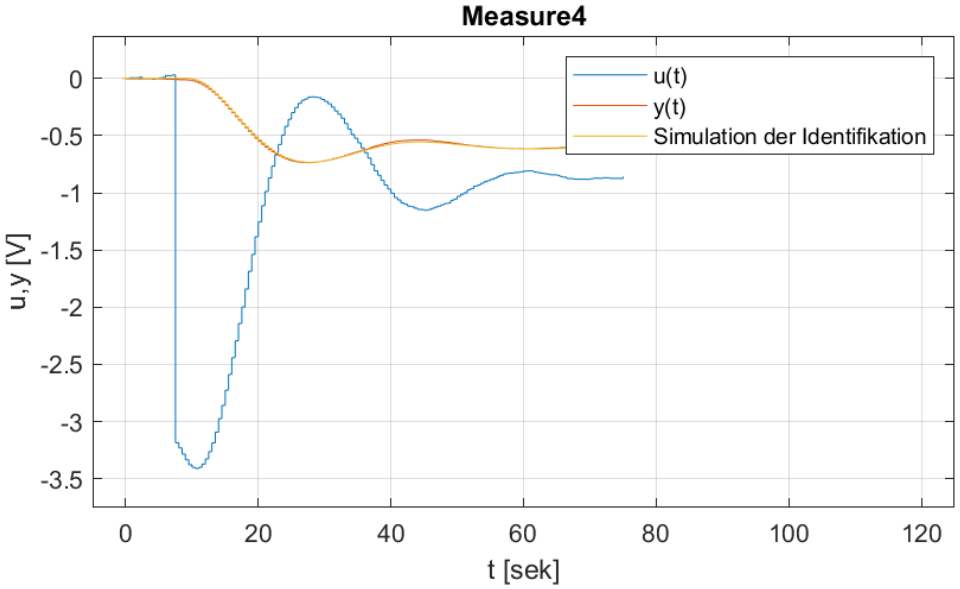

Abbildung A.7: Messreihe ’Measure4’ an der Labor-Regelstrecke aus Abschnitt 6.1

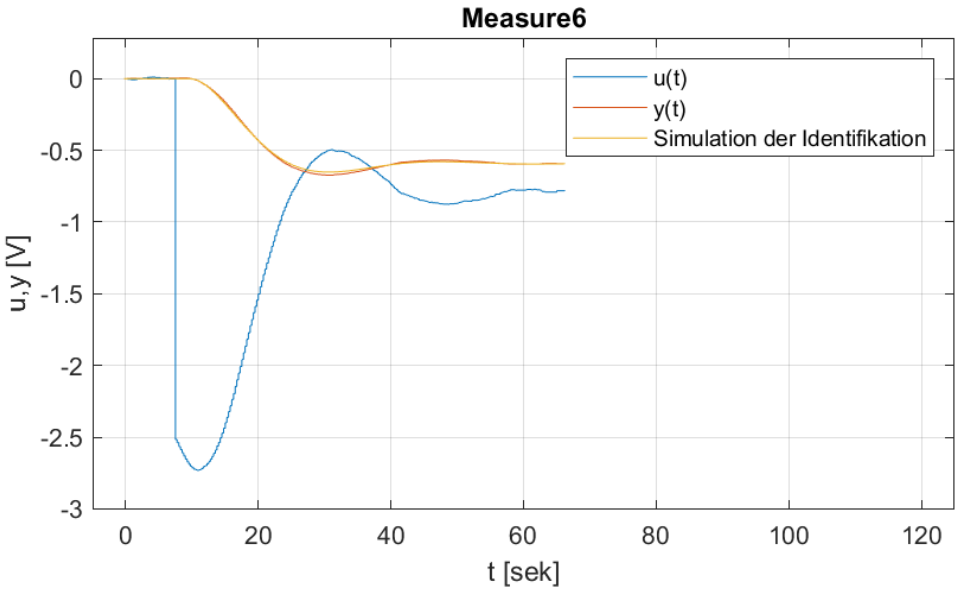

Abbildung A.8: Messreihe ’Measure6’ an der Labor-Regelstrecke aus Abschnitt 6.1

B Inhalt der DVD

Ein weiterer Anhang dieser Bachelorthesis befindet sich auf einer DVD, die beim Erstprüfer Prof. Dr. Michael Erhard zur Einsicht vorliegt. Es folgt eine Auflistung der wichtigsten enthaltenen Dateien und Ordner. Einzelne Dateien sind in *kursiv* gedruckt, Ordner in **fett**. Die Ebene der Auflistung spiegelt die Hierarchieebene im Dateisystem der DVD wieder.

- *Bachelorarbeit Tiede.pdf*
  diese Thesis in elektronischer Form
- *sysIdent Doxygen Doc.html*
  .html Datei, die die Doxygen Dokumentation von sysIdent öffnet
- **binaries**
  Ordner mit ausführbarem Programm
  - *sysident.exe*
  - benötigte .dll-Files
- **C-Code**
  Ordner mit verwendetem/erstelltem Quellcode
  - **alglib**
    ALGLIB-Software Bibliothek in der verwendeten Version
  - **bacnet-stack-0.84.**
    BACnet-stack Bibliothek in der verwendeten Version
  - **sysIdent**
    * Quellcode von sysIdent
    * Ordner **doc** mit Doxygen Dokumentation
- **Messreihen**
  Ordner mit aufgenommenen Messdaten
  - *README Datenformat.txt*
    enthält genaue Erklärung des Datenformats für die Messreihen in den Unterordnern

- *get_all_ident_data.m*
  MATLAB® Funktion zum Einlesen der Daten aus den Unterordnern
- **Measure3**
  Beispielhaft sind hier die Dateien für eine Messreihe gelistet. Die Struktur der Ordner für die folgenden Messreihen sind identisch.
  - *controller.log*
    .log-Datei mit den ermittelten Reglerparametern
  - *data.csv*
    CSV-Datei mit den Messwerten und auf Abtastschritte gerundete Messzeitpunkten
  - *fit.log*
    .log-Datei mit Informationen zu den durchgeführten Systemidentifikationen mit allen ermittelten Streckenmodellen
- **Measure4**
- **Measure6**
- **PRBS1**
- **PRBS2**
- **PRBS4**
- **Step1**
- **Step2**

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 7. Juni 2018	
Ort, Datum	Unterschrift