Jakob Rehrl, BSc

Methoden zur On-Line-Schätzung von Totzeiten in zeitdiskreten Regelkreisen

MASTERARBEIT

zur Erlangung des akademischen Grades
Diplom-Ingenieur

Masterstudium Elektrotechnik

eingereicht an der

TECHNISCHEN UNIVERSITÄT GRAZ

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

Institut für Regelungs- und Automatisierungstechnik

Mitbetreuer
Dipl.-Ing. Dr.techn. Georg Stettinger

Kompetenzzentrum - Das virtuelle Fahrzeug, Forschungsgesellschaft mbH

Graz, September 2015

Kurzfassung

Die vorliegende Arbeit beschäftigt sich mit Methoden zur On-Line-Schätzung von Totzeiten in zeitdiskreten Regelkreisen. Die Einleitung umfasst die Motivation mit dem zentralen Ziel: der Entwicklung einer Methode, welche weder in der Art der Signale, noch in der Charakteristik des betrachteten Systems beschränkt ist. Zudem sollen auch zeitlich variierende Totzeiten zuverlässig erkannt werden. Hierfür werden einführend Totzeit-Systeme im Allgemeinen und die Auswirkungen von Totzeiten in Regelkreisen im Speziellen untersucht. Es folgt eine Literaturrecherche zu bereits existierenden Ansätzen zur Schätzung von Totzeiten. Hier kommen zunächst drei signalbasierte Methoden zur Sprache, welche grundsätzlich die Fähigkeit zur On-Line-Totzeitschätzung aufweisen: Eine Methode, die auf der Sprungantwort eines dynamischen Systems basiert, ein korrelativer Ansatz sowie eine Herangehensweise mit Hilfe adaptiver Filterung. Es wird jedoch festgestellt, dass jede dieser drei Methoden entweder in der Art der Signale oder in der Charakteristik des Systems beschränkt ist. Die erste modellbasierte Methode, die vorgestellt wird, arbeitet mit dem Einsatz von Kalman-Filtern. Auch hier liegt jedoch eine Beschränkung in der Art der Signale vor. Ein weiterer Ansatz - die gradientenbasierte Methode - scheitert hauptsächlich an seinen schlechten Eigenschaften bezüglich zeitvariabler Totzeiten. Es folgen einige Methoden, welche auf der rekursiven Least-Squares-Schätzung der Koeffizienten eines linearen Systems beruhen. All diese Methoden besitzen grundsätzlich das Potenzial, den Ansprüchen der erwähnten Zielsetzung gerecht zu werden. Nach einer eingehenden Analyse anhand mehrerer Testbeispiele mit unterschiedlichen Anregungssignalen stellt sich der Elnaggar-Algorithmus als vielversprechendster Ansatz heraus. Dieser Ansatz wird weiterverfolgt und für die Anwendung innerhalb eines Koppelelements in einem zeitdiskreten Regelkreis adaptiert, wobei sowohl auf Strecken- als auch auf Reglerseite Totzeiten auftreten können. Es folgen vier Maßnahmen zur Verbesserung der Anwendbarkeit des Algorithmus: Die Berücksichtigung von Einschwingvorgängen der geschätzten Modellparameter, die Optimierung der Vergessensfaktoren, die Verringerung der Rauschsensibilität durch direktionales Vergessen und die Verwendung einer Sprungerkennung bei sprungförmigen Eingangsgrößen. Nach einer kurzen Skizzierung der Implementierung des Algorithmus inklusive der genannten zusätzlichen Maßnahmen in MATLAB/SIMULINK wird die Leistungsfähigkeit der entwickelten Methode abschließend anhand zweier Beispiele aufgezeigt.

Abstract

This thesis deals with recursive time-delay estimation approaches for discrete-time control loops. The introduction motivates the main goal: The development of a method which has no limitations in terms of signal types or the characteristic of the considered system. Moreover, time-variant time-delays should be recognized reliably. Therefore, time-delay systems in general and the effect of delays in control loops are analyzed introductory. Next, a literature research on existing strategies to estimate time-delays follows. Three signal-based methods capable of estimating time-delays online are addressed initially: A method based on the step response of a dynamic system, a correlative approach and a strategy by means of adaptive filtering. However, it is pointed out that each of these methods is limited either in the type of the signal or in the characteristic of the system. The first model-based method which is introduced is based on Kalman filters. There is also a limitation on the type of signals though. Another approach is the gradient-based method which fails mainly due to problems with time-varying delays. The literature research continues with several methods basing on recursive least-squares estimation of the coefficients of a linear system. All these methods offer the potential to fulfill the requirements of the mentioned goal. After a detailed analysis by means of several examples with different excitation signals the Elnaggar algorithm turns out to be the most promising approach. This approach is pursued in the following sections and an adaption for the use within a coupling element in a control loop (delays on plant-side and on controller-side) is made. Furthermore, four measures to improve usability of the chosen method are taken: a consideration of transient effects for the estimated model parameters, an optimization of the forgetting factors, a reduction of noise sensitivity by directional forgetting and the use of a step-detection for signals which mainly consist of jumps. After a brief explanation on the implementation of the algorithm including the mentioned additional measures in MATLAB/SIMULINK, the performance of the developed method is demonstrated with the help of two examples at last.

Danksagung

Diese Arbeit entstand am *Kompetenzzentrum - Das virtuelle Fahrzeug, Forschungsgesellschaft mbH* in Graz, Österreich. Ich bedanke mich für die Förderung im Rahmen des *COMET K2 - Competence Centers for Excellent Technologies* Programms des Österreichischen Bundesministeriums für Verkehr, Innovation und Technologie (*bmvit*), des Österreichischen Bundesministeriums für Wissenschaft, Forschung und Wirtschaft (*bmwfw*), der Österreichischen Forschungsförderungsgesellschaft mbH (*FFG*), des Landes Steiermark sowie der Steirischen Wirtschaftsförderung (*SFG*).

Ich möchte mich zudem bei Prof. Martin Horn und Dr. Georg Stettinger für die hervorragende wissenschaftliche Betreuung bedanken. Sie standen mir stets hilfsbereit und unterstützend zur Seite und gaben mir wichtige Denkanstöße bei der Entstehung dieser Arbeit. Des Weiteren möchte ich Markus Bachinger, Dr. Markus Neumayer, Dr. Michael Stolz und Markus Tranninger für ihre Hilfestellungen danken.

Nicht zuletzt danke ich meiner Freundin Viola für ihren großartigen und immer verständnisvollen Beistand während des Verfassens dieser Arbeit und meines gesamten Studiums. Abschließend bedanke ich mich bei meinen Eltern und Großeltern für ihre Geduld und ihre finanzielle Unterstützung, die mein Studium und diese Arbeit erst möglich gemacht haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

A Einleitung

1 Motivation

Ein moderner industrieller Entwicklungsprozess besteht oftmals aus einer Vielzahl von untergeordneten Prozessen, welche sich zunächst vergleichsweise eigenständig mit der Entwicklung einzelner Komponenten des gewünschten Produktes beschäftigen. In einer bestimmten Projektphase wird es jedoch wichtig, nicht nur die autarke Funktionsweise sondern auch das Zusammenspiel der einzelnen Bestandteile genauer zu betrachten. Dieser Schritt ist von immenser Bedeutung, um mögliche Probleme in einem Produktkonzept bereits in einem frühen Projektstadium erkennen und beheben zu können. Beteiligte Ingenieure sind also dazu angehalten, neben der Optimierung der Teilkomponenten auch ihre reibungslose Interaktion im Auge zu behalten, wobei hier Systeme aus unterschiedlichsten Fachbereichen zusammentreffen können. Als Beispiel sei hier ein Kraftfahrzeug genannt – dort interagieren elektronische, mechanische sowie hydraulische Teilsysteme. Im Mittelpunkt der Interaktion stehen zumeist mikrocontroller-basierte Steuergeräte, welche die einzelnen Komponenten über Kommunikationsmedien miteinander in Verbindung bringen. Nicht selten übernehmen die Steuergeräte dabei regelungstechnische Aufgaben, indem ein im Mikrocontroller hinterlegter Regelalgorithmus das Verhalten eines oder mehrerer Systeme (Regelstrecken) beeinflusst. Bekanntermaßen treten hierbei jedoch in den meisten Fällen Verzögerungen auf, beispielsweise durch die begrenzte Geschwindigkeit der verwendeten Kommunikationsmedien. Diese in der Regelungstechnik als Totzeit bezeichneten Verzögerungen stellen eine große Herausforderung für den Reglerentwurf dar, da sie das Verhalten des Regelkreises zumeist entscheidend beeinflussen. Bezüglich der Kompensation von Auswirkungen solcher Totzeiten existieren unterschiedlichste Lösungsansätze. Diese Lösungsansätze basieren oft auf der Annahme, dass die Totzeit bekannt ist und keiner zeitlichen Änderung unterliegt. In der Realität ist diese Annahme jedoch vielmals nicht gegeben und die Totzeit ist ein unbekannter Systemparameter, der sich mit der Zeit verändern kann. Da das Wissen über den Wert der Totzeit für den Reglerentwurf aber von entscheidender Bedeutung ist, benötigt man geeignete Methoden zur Abschätzung der auftretenden Verzögerung. Genau an diesem Punkt setzt die vorliegende Arbeit an.

2 Zielsetzung

Diese Arbeit soll zunächst auf in der Literatur bereits vorhandene Methoden zur On-Line-Totzeitschätzung für zeitdiskrete Systeme eingehen und diese analysieren. Dabei sollen sowohl signalbasierte als auch modellbasierte Ansätze in Betracht gezogen werden, wobei diese auf unterschiedlichste Eigenschaften hin untersucht werden sollen. So sind beispielsweise der Rechenaufwand, die Art der benötigten Anregungssignale oder auch die Anwendbarkeit bei unterschiedlichen Systemordnungen zu durchleuchten. Anschließend sollen geeignete Testbeispiele zum Einsatz kommen, um die Leistungsfähigkeit und das Potenzial der vorgestellten Ansätze aufzuzeigen. Zudem sollen die angeführten Ansätze bei Bedarf entsprechend optimiert bzw. adaptiert werden, um die Einsetzbarkeit für bestimmte Zwecke zu gewährleisten. Im Besonderen ist eine Verwendung innerhalb der am *Kompetenzzentrum – Das virtuelle Fahrzeug, Forschungsgesellschaft mbH*[1] entwickelten Co-Simulationsplattform *ICOS*[2] (*Independent Co-Simulation*) vorgesehen. Dieses automotive Entwicklungstool ermöglicht eine domänenübergreifende Simulation unterschiedlicher Teilkomponenten eines Kraftfahrzeugs. Innerhalb dieser Software kommt eine modellbasierte Kopplung zum Einsatz, welche unter anderem den Effekt von Kommunikationstotzeiten zu mindern versucht. Daher werden auch Messdaten aus bereits existierenden ICOS-Fallbeispielen zur Evaluierung der gewählten Totzeitschätzungsmethode verwendet.

3 Gliederung

Die Arbeit startet mit einer grundlegenden Betrachtung von Systemen mit auftretenden Totzeiten (Kapitel B), wobei sowohl auf den zeitkontinuierlichen, als auch auf den zeitdisktreten Fall eingegangen wird. Zudem wird der Einfluss von Totzeiten in Regelkreisen erläutert und mögliche Lösungsansätze kommen kurz zur Sprache. Im nächsten Kapitel werden mögliche signalbasierte Methoden zur Schätzung von Totzeiten aufgeführt (Kapitel C). Diese beinhalten einen Ansatz, der auf der Sprungantwort eines Systems beruht, einen korrelativen Ansatz, sowie eine Methode, die auf adaptiver Filterung beruht. Kapitel D beschäftigt sich anschließend eingehend mit modellbasierten Methoden. Zunächst wird hier ein Kalman-Filter-Ansatz und eine gradientenbasierte Herangehensweise besprochen. Es folgen einige rekursive Least-Squares-Methoden, wobei hier einführend auf den rekursiven Least-Squares-Algorithmus zur Identifizierung eines zeitdiskreten Übertragungsverhaltens näher eingegangen wird. Dieser bildet die Basis für alle in weiterer Folge erwähnten Ansätze. Im nächsten Schritt wird dann versucht, den leistungsfähigsten der vorgestellten Al-

[1] http://www.v2c2.at

[2] http://www.v2c2.at/icos

gorithmen anhand geeigneter Testbeispiele herauszufinden. Kapitel F beschäftigt sich damit, die gewählte Methode hinsichtlich ihrer Anwendbarkeit mittels geeigneter Maßnahmen zu verbessern. Hier wird auf eine ausreichende Einschwingphase, die Wahl geeigneter Vergessensfaktoren, die Verwendung einer alternativen Vergessensstrategie und eine Sprungerkennung bei sprungförmigen Eingangsgrößen eingegangen. Anschließend folgt in Kapitel G eine Beschreibung der Implementierung der erarbeiteten Ergebnisse in einem MATLAB/SIMULINK-Block. Kapitel H veranschaulicht daraufhin die Leistungsfähigkeit des verwendeten Algorithmus anhand zweier Anwendungsbeispiele. bevor ein Fazit und ein kurzer Ausblick die vorliegende Arbeit abrunden (Kapitel I).

B Grundlegendes zu Totzeit-Systemen

In diesem Kapitel sollen die grundlegenden Eigenschaften von totzeitbehafteten dynamischen Systemen aufgezeigt werden. Hierbei wird sowohl der zeitkontinuierliche als auch der zeitdiskrete Fall berücksichtigt. Zudem wird auf Auswirkungen von Totzeiten in geregelten Systemen Rücksicht genommen.

1 Zeitkontinuierliche Darstellung

In realen dynamischen Systemen kommt es immer wieder zum Auftreten von Totzeiten, also zu zeitlichen Verzögerungen bestimmter physikalischer Größen. Dies kann beispielsweise durch Transportvorgänge oder begrenzte Geschwindigkeiten in Übertragungsmedien hervorgerufen werden. Allgemein hat eine Totzeit T im zeitkontinuierlichen Fall auf ein Signal $u(t)$ folgenden Einfluss:

$$u\Big(\,t\,\Big) \xrightarrow{\text{Totzeit } T} u\Big(\,t-T\,\Big) \tag{B.1.1}$$

Bei Betrachtung der mathematischen Notation (B.1.1) kann man erkennen, dass das totzeitbehaftete Signal $u(t-T)$ zum Zeitpunkt t denjenigen Wert annimmt, den das totzeitfreie Signal $u(t)$ schon zum Zeitpunkt $t-T$ angenommen hätte.

1.1 Differentialgleichung

Tritt eine Totzeit in dynamischen Systemen auf, so hat diese natürlicherweise auch einen Einfluss auf das dynamische Verhalten der entsprechenden Größen. Nimmt man ein einfaches SISO-System (*Single Input Single Output*) mit Eingang $u(t)$ und Ausgang $y(t)$ an, so kann man das dynamische Verhalten des Systems mit Zählergrad m und Nennergrad n $(n \geq m)$ bei Annahme von Zeitinvarianz durch folgende Differentialgleichung beschreiben:

$$y^{(n)}\Big(\,t\,\Big) = f\Big(\,u\Big(\,t\,\Big), \dot{u}\Big(\,t\,\Big), \ldots, u^{(m)}\Big(\,t\,\Big), y\Big(\,t\,\Big), \dot{y}\Big(\,t\,\Big), \ldots, y^{(n-1)}\Big(\,t\,\Big)\Big) \tag{B.1.2}$$

Tritt nun am Eingang $u(t)$ eine konstante Totzeit T_u auf, dann verändert sich diese Beziehung wie folgt:

$$y^{(n)}\Big(\,t\,\Big) = f\Big(\,u\Big(\,t - T_u\,\Big),\ldots,u^{(m)}\Big(\,t - T_u\,\Big),y\Big(\,t\,\Big),\ldots,y^{(n-1)}\Big(\,t\,\Big)\Big) \tag{B.1.3}$$

Ebenfalls möglich ist der Fall, dass eine Totzeit nicht am Eingang, sondern am Ausgang auftritt. Dies kann dadurch beschrieben werden, dass das Ausgangssignal $y(t)$ und seine Ableitungen zum Zeitpunkt $t + T_y$ anstatt zum Zeitpunkt t herangezogen werden:

$$y^{(n)}\Big(\,t + T_y\,\Big) = f\Big(\,u\Big(\,t\,\Big),\ldots,u^{(m)}\Big(\,t\,\Big),y\Big(\,t + T_y\,\Big),\ldots,y^{(n-1)}\Big(\,t + T_y\,\Big)\Big) \tag{B.1.4}$$

Substituiert man $\tau \coloneqq t + T_y$, so erhält man:

$$y^{(n)}\Big(\,\tau\,\Big) = f\Big(\,u\Big(\,\tau - T_y\,\Big),\ldots,u^{(m)}\Big(\,\tau - T_y\,\Big),y\Big(\,\tau\,\Big),\ldots,y^{(n-1)}\Big(\,\tau\,\Big)\Big) \tag{B.1.5}$$

Hieraus wird deutlich, dass die Beziehungen (B.1.4) äquivalent zur Beziehung (B.1.3) ist. Es kann also rein anhand des Eingangs- und des Ausgangssignals nicht festgestellt werden, ob der Eingang oder der Ausgang totzeitbehaftet ist. Gleiches gilt für ein Auftreten von Totzeiten sowohl am Eingang, als auch am Ausgang. Der Einfachheit halber wird im weiteren Verlauf nur noch der Fall einer Eingangsverzögerung nach Beziehung (B.1.3) betrachtet, da dieser die beiden erläuterten anderen Fälle mit abdeckt.

1.2 Übertragungsfunktionen im Laplace-Bereich

Bekanntermaßen lassen sich lineare, zeitinvariante Differentialgleichungen in den Laplace-Bereich überführen. Hierfür benötigt man folgende Korrespondenz für Ableitungen im Zeitbereich:

$$\mathcal{L}\Big\{\,a^{(l)}\Big(\,t\,\Big)\Big\} = s^l A\Big(\,s\,\Big) \tag{B.1.6}$$

Unter Annahme einer linearen, zeitinvarianten Differentialgleichung (B.1.2) ergibt sich daher:

$$s^n Y\Big(\,s\,\Big) = f\Big(\,U\Big(\,s\,\Big),sU\Big(\,s\,\Big),\ldots,s^m U\Big(\,s\,\Big),Y\Big(\,s\,\Big),sY\Big(\,s\,\Big),\ldots,s^{n-1}Y\Big(\,s\,\Big)\Big) \tag{B.1.7}$$

Tritt nun eine konstante Totzeit auf, so lässt sich diese im Laplace-Bereich ebenfalls durch eine einfache Korrespondenz berücksichtigen:

$$\mathcal{L}\Big\{\,a\Big(\,t - T\,\Big)\Big\} = A\Big(\,s\,\Big)\mathrm{e}^{-sT} \tag{B.1.8}$$

Daher ergibt sich für ein totzeitbehaftetes zeitinvariantes System:

$$s^n Y\Big(\,s\,\Big) = f\Big(\,U\Big(\,s\,\Big)\mathrm{e}^{-sT},\ldots,s^m U\Big(\,s\,\Big)\mathrm{e}^{-sT},Y\Big(\,s\,\Big),\ldots,s^{n-1}Y\Big(\,s\,\Big)\Big) \tag{B.1.9}$$

Da man wie angemerkt von einem System mit linearem Übertragungsverhalten, also mit linearer Funktion $f(\cdot)$ ausgeht, vereinfacht sich die Beziehung wie folgt:

$$\begin{gathered} s^n Y\Big(\,s\,\Big) = \\ b_0 U\Big(\,s\,\Big)\mathrm{e}^{-sT} + \ldots + b_m s^m U\Big(\,s\,\Big)\mathrm{e}^{-sT} - a_0 Y\Big(\,s\,\Big) - \ldots - a_{n-1}s^{n-1}Y\Big(\,s\,\Big) \end{gathered} \tag{B.1.10}$$

In weiterer Folge lässt sich eine Übertragungsfunktion anschreiben:

$$G\Big(\,s\,\Big) = \frac{U\Big(\,s\,\Big)}{Y\Big(\,s\,\Big)} = \frac{b_m s^m + b_{n-1}s^{n-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \ldots + a_1 s + a_0}\,\mathrm{e}^{-sT} \tag{B.1.11}$$

Man kann erkennen, dass sich die Übertragungsfunktion eines linearen, zeitinvarianten Totzeit-Systems von der eines äquivalenten Systems ohne Totzeit nur durch den nachstehenden Term e^{-sT} unterscheidet.

2 Zeitdiskrete Darstellung

Da diese Arbeit auf eine digitale Implementierung eines Algorithmus zur Totzeitschätzung abzielt, wird hier der Einfluss von Totzeiten in zeitdiskreten Systemen ebenfalls erläutert. Durch eine Totzeit d verzögert sich ein Signal u_k ganz allgemein wie folgt:

$$u_k \xrightarrow{\text{Totzeit } d} u_{k-d} \tag{B.2.1}$$

Das totzeitbehaftete Signal u_{k-d} nimmt also zum Zeitschritt k denjenigen Wert an, den es ohne Totzeit schon zum Zeitschritt $k-d$ angenommen hätte.

2.1 Differenzengleichung

Zur Beschreibung dynamischer Systeme werden im zeitdiskreten Fall Differenzengleichungen folgender Form verwendet:

$$y_k = f\Big(u_{k+m-n}, u_{k+m-n-1}, \ldots, u_{k-n}, y_k, y_{k-1}, \ldots, y_{k-n} \Big) \tag{B.2.2}$$

Dabei kennzeichnet auch hier m den Zählergrad und n den Nennergrad $(n \geq m)$. Bei konstanter Totzeit d verändert sich die Systembeschreibung wie folgt:

$$y_k = f\Big(u_{k+m-n-d}, u_{k+m-n-1-d}, \ldots, u_{k-n-d}, y_{k-1}, y_{k-2}, \ldots, y_{k-n} \Big) \tag{B.2.3}$$

Wie schon für den zeitkontinuierlichen Fall beschrieben macht dabei der „Ort“ der Totzeit (Auftreten am Eingang oder am Ausgang) keinen Unterschied.

2.2 Übertragungsfunktionen im z-Bereich

Für den Übergang in den z-Bereich benötigt man hier folgende Korrespondenz:

$$\mathcal{Z}\Big\{ a_{k-l} \Big\} = z^{-l} A\Big(z \Big) \tag{B.2.4}$$

Liegt eine lineare, zeitinvariante Differenzengleichung (B.2.2) vor, so sieht diese im Bildbereich also wie folgt aus:

$$Y\Big(z \Big) = f\Big(z^{m-n-d} U\Big(z \Big), \ldots, z^{-n-d} U\Big(z \Big), z^{-1} Y\Big(z \Big), \ldots, z^{-n} Y\Big(z \Big) \Big) \tag{B.2.5}$$

Aufgrund der linearen Funktion $f(\cdot)$ erhält man:

$$\begin{gathered} Y\Big(z \Big) = \\ b_m z^{m-n} z^{-d} U\Big(z \Big) + \ldots + b_0 z^{-n} z^{-d} U\Big(z \Big) - a_{n-1} z^{-1} Y\Big(z \Big) - \ldots - a_0 z^{-n} Y\Big(z \Big) \end{gathered} \tag{B.2.6}$$

Nach kurzer Umformung erhält man daraus folgende Übertragungsfunktion:

$$\begin{aligned} G\Big(z \Big) &= \frac{Y\Big(z \Big)}{U\Big(z \Big)} \\ &= \frac{b_m z^{m-n} + b_{m-1} z^{m-n-1} \ldots + b_0 z^{-n}}{1 + a_{n-1} z^{-1} + \ldots + a_0 z^{-n}} \, z^{-d} \\ &= \frac{b_m z^{m} + b_{m-1} z^{m-1} \ldots + b_0}{z^{n} + a_{n-1} z^{n-1} + \ldots + a_0} \, z^{-d} \end{aligned} \tag{B.2.7}$$

Auch hier zeigt sich, dass sich die Übertragungsfunktion eines linearen, zeitinvarianten Totzeit-Systems von der eines äquivalenten Systems ohne Totzeit nur durch den nachstehenden Term z^{-d} unterscheidet.

3 Totzeit in Regelkreisen

Abschließend soll in diesem Kapitel der Einfluss von Totzeiten in Regelkreisen dargestellt werden, wobei der Einfachheit halber nur der zeitkontinuierliche Fall betrachtet wird. Hierfür wird eine im Vorwärtszweig eines Standardregelkreises auftretende Totzeit T angenommen. Abbildung 1 zeigt das entsprechende Strukturbild:

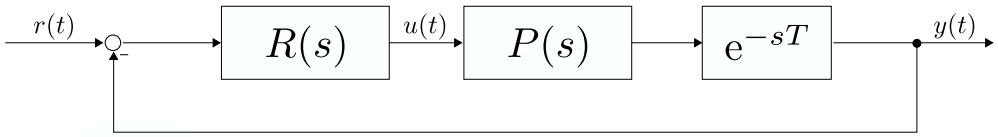

Abbildung 1: Standard-Regelkreis mit Totzeit

3.1 Einfluss auf den Frequenzgang

Am besten zeigt sich der Einfluss der Totzeit in Regelkreisen, wenn man ihre Auswirkung auf den Frequenzgang des offen Kreises analysiert und entsprechende Stabilitätsbetrachtungen unternimmt. Hierfür nimmt man die Totzeit T wiederum als konstant an und untersucht alleine das Frequenzverhalten des Totzeitterms e^{-sT}. Der Betragsgang ergibt sich wie folgt:

$$\left| \mathrm{e}^{-j\omega T} \right| = 1 \quad ; \quad \arg\left\{ \mathrm{e}^{-j\omega T} \right\} = -\omega T \tag{B.3.1}$$

Der Totzeitterm e^{-sT} wirkt also als Allpass und hat keinen Einfluss auf den Betragsgang des offenen Kreises. Jedoch führt er zu einer frequenzabhängigen Phasensenkung um $-\omega T$ im offenen Kreis. Es wird klar, dass es dadurch zu einer Verringerung der Phasenreserve und womöglich sogar zum Verlust der BIBO-Stabilität (*Bounded Input Bounded Output*) im geschlossenen Kreis kommen kann. Als Beispiel sei folgender offene Kreis (ohne Totzeit) angeführt:

$$L\left(s\right) = R\left(s\right) P\left(s\right) = \frac{10}{\left(1 + \frac{s}{10}\right)\left(1 + \frac{s}{10}\right)} \tag{B.3.2}$$

Diese Übertragungsfunktion ist vom „einfachen Typ“ [1], d.h. sie erfüllt folgende Kriterien:

- Alle Pole haben negativen Realteil, bis auf maximal einen Pol bei $s = 0$ (hier zwei Pole bei $s = -10$)
- $V > 0$ (hier: $V = 4$)
- Die Betragskennlinie $|\, L(j\omega)\,|_{\mathrm{dB}}$ verläuft für $\omega \to \infty$ unterhalb der 0 db - Linie, sie hat also Tiefpasscharakter
- Es gibt genau eine Durchtrittsfrequenz ω_c

Für Übertragungsfunktionen vom „einfachen Typ“ kann zur Überprüfung der BIBO-Stabilität das vereinfachte Nyquist-Kriterium (Schnittpunktkriterium) angewendet werden. Dazu überprüft man den Frequenzgang des offenen Kreises auf positive Phasenreserve. Abbildung 2 zeigt ebendiesen:

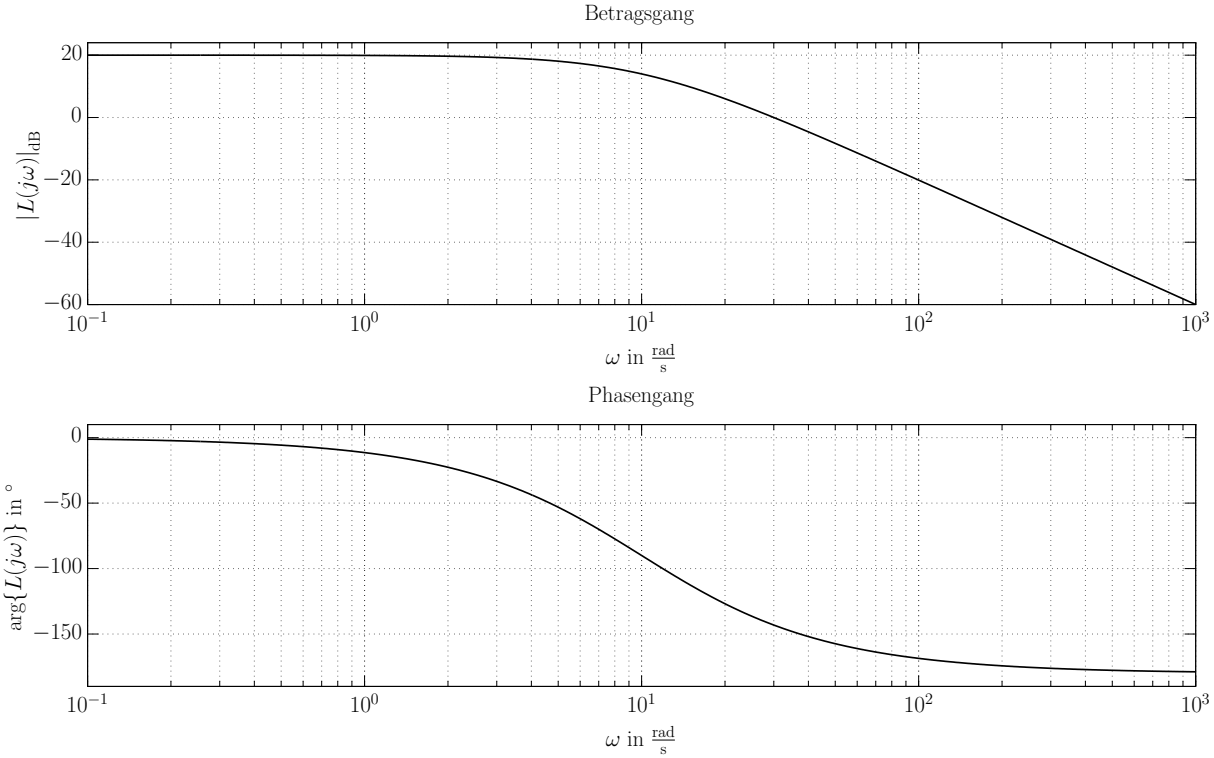

Abbildung 2: Frequenzgang des offenen Kreises ohne Totzeit

Ohne Totzeit ergibt sich bei der Durchtrittsfrequenz von $\omega_c = 30$ rad/s eine Phasenreserve von $\phi_R = 36.9°$. Es sei nun im Standardregelkreis laut Abbildung 1 eine Totzeit von $T = 25$ ms angenommen. Es ist möglich, das vereinfachte Schnittpunktkriterium auch auf den Regelkreis inklusive Totzeit anzuwenden. Hierzu zeigt Abbildung 3 zunächst nur den Frequenzgang des Totzeitglieds alleine:

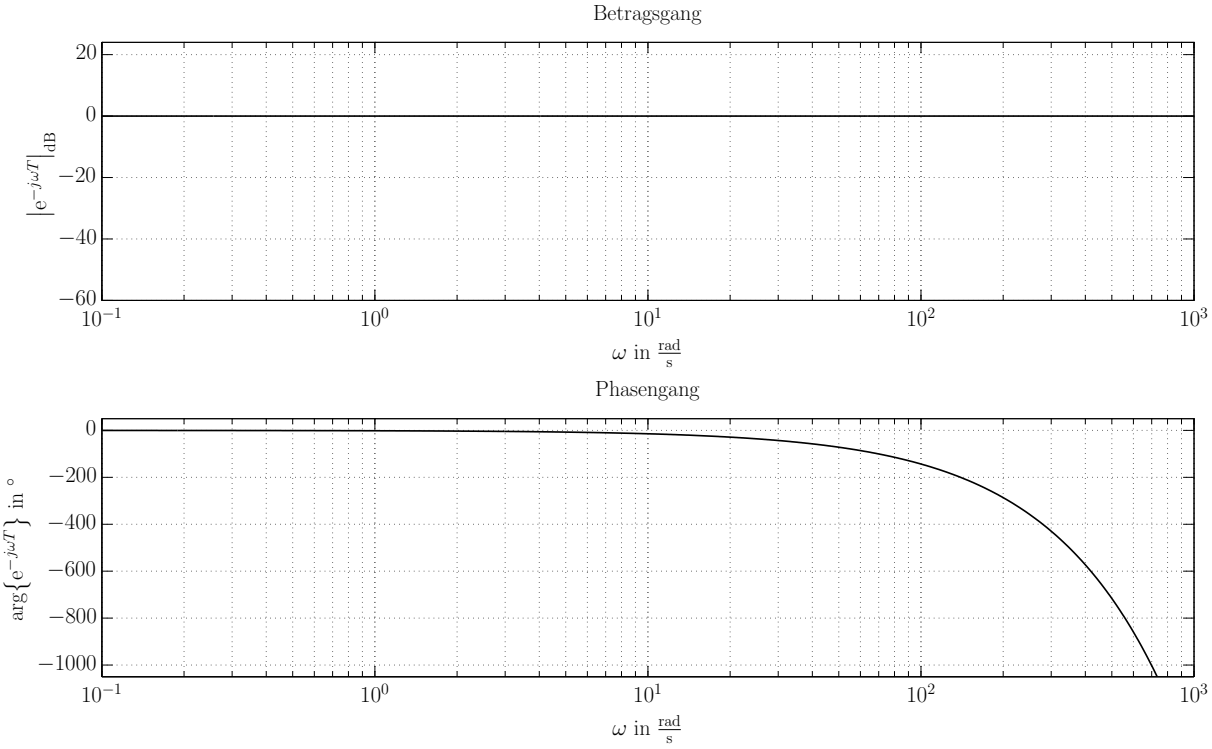

Abbildung 3: Frequenzgang des Totzeitglieds

Man erkennt, dass die Phase durch das Totzeitglied wie beschrieben mit steigender Frequenz immer stärker gesenkt wird (man beachte die logarithmische Frequenzskala). Für den offenen Kreis inklusive Totzeit ergibt sich damit folgender Frequenzgang:

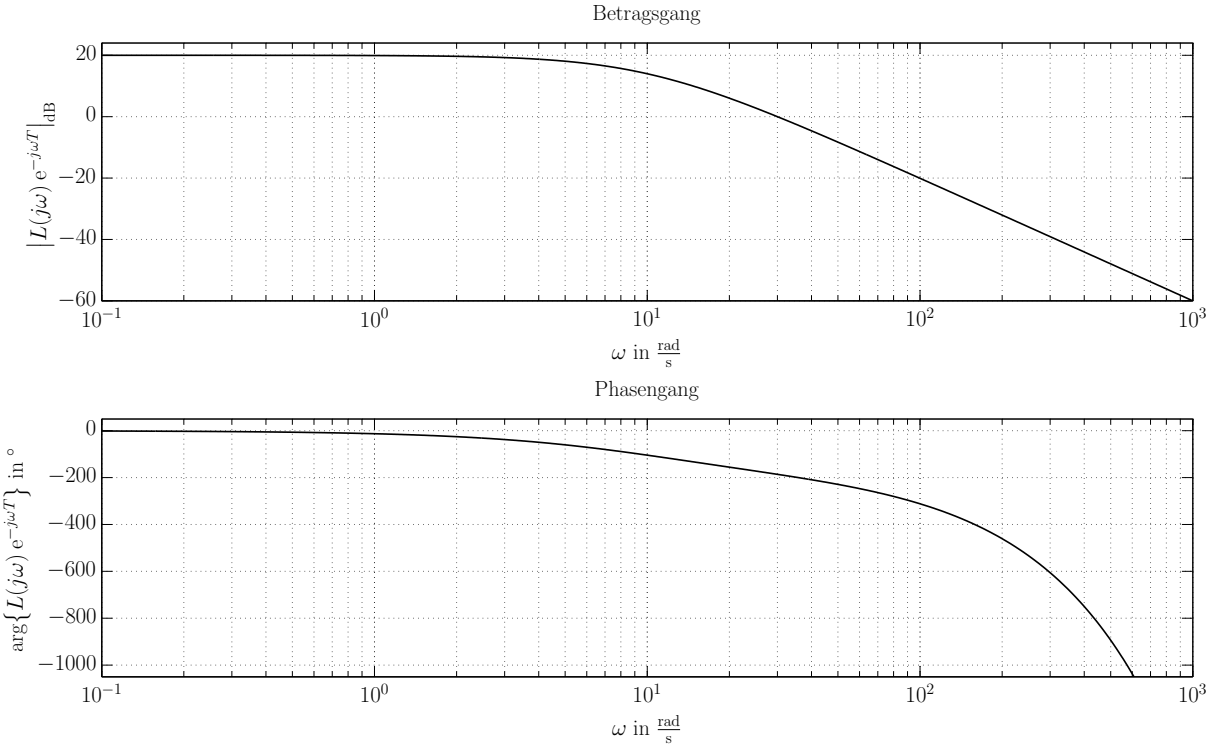

Abbildung 4: Frequenzgang des offenen Kreises mit Totzeit

Die Phasenreserve beträgt nun $\phi_r = -6.1°$ bei gleicher Durchtrittsfrequenz $\omega_c = 30\ \mathrm{rad/s}$, womit das System nicht mehr stabil ist. Interessant ist auch eine kurze Analyse der zugehörigen Ortskurve, welche ausschnittsweise in Abbildung 5 zu sehen ist:

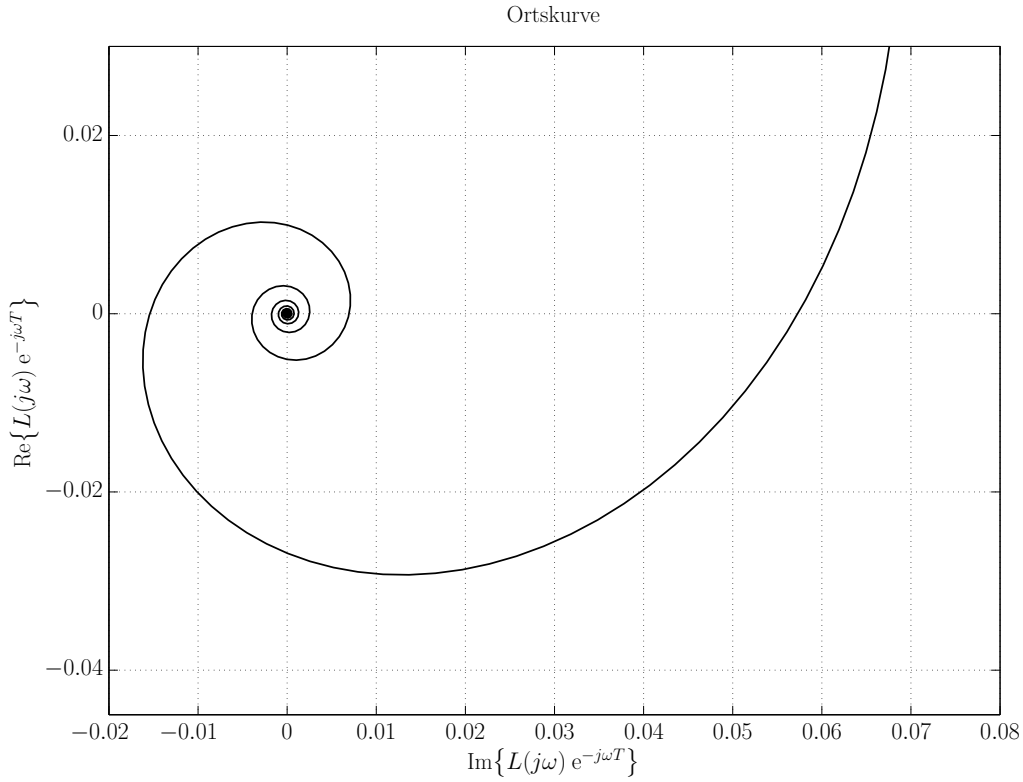

Abbildung 5: Ortskurve des offenen Kreises mit Totzeit (Ausschnitt)

Man erkennt, dass sich die Ortskurve für steigende Frequenzen spiralförmig dem Nullpunkt annähert. Dieses Verhalten ist ebenfalls erklärbar – wie aus dem Phasengang bekannt sinkt die Phase mit zunehmender Frequenz kontinuierlich ab, was bei näherer Betrachtung genau eine solche Form der Ortskurve nach sich zieht.

Dieses Beispiel zeigt also deutlich, dass Totzeiten in Regelkreisen zu erheblichen Stabilitätsproblemen führen können und daher schon beim Reglerentwurf in jedem Fall berücksichtigt werden sollten.

3.2 Mögliche Lösungsansätze

Wie im vorigen Abschnitt angemerkt, ist es für den Reglerentwurf von essentieller Bedeutung, mögliche auftretende Totzeiten zu berücksichtigen. Hierzu existiert eine große Fülle an möglichen Lösungsansätzen, wovon zwei an dieser Stelle kurz erläutert werden.

3.2.1 Padé-Approximation

Ein weit verbreiteter Ansatz zur Behandlung von Totzeiten ist die Approximation des Totzeit-Glieds e^{-sT} durch eine rationale Übertragungsfunktion. Sehr gebräuchlich ist dabei die Verwendung der Padé-Approximation [2], bei welcher rationale Funktionen als Annäherung dienen. Bricht man die Approximation für das Totzeitglied schon nach der ersten Ordnung ab (Padé-Approximation erster Ordnung), dann erhält man:

$$\mathrm{e}^{-sT} \approx \frac{1 - s\dfrac{T}{2}}{1 + s\dfrac{T}{2}} \tag{B.3.3}$$

Zu beachten ist, dass diese Approximation nicht minimalphasig ist. Selbiges gilt für die Approximation der Totzeit als Padé-Approximation zweiter Ordnung, welche ebenfalls häufig verwendet wird. Sie lautet:

$$\mathrm{e}^{-sT} = \frac{1 - s\dfrac{T}{2} + s^2\dfrac{T^2}{12}}{1 - s\dfrac{T}{2} + s^2\dfrac{T^2}{12}} \tag{B.3.4}$$

Anhand dieser Approximationen wird es möglich, das Totzeitglied explizit als Teil einer linearen Strecke zu interpretieren. Man nimmt also beispielsweise bei Verwendung der Padé-Approximation erster Ordnung folgende Strecken-Übertragungsfunktion an:

$$\tilde{P}\Big(\, s \,\Big) = P\Big(\, s \,\Big)\mathrm{e}^{-sT} \approx P\Big(\, s \,\Big)\frac{1 - s\dfrac{T}{2}}{1 + s\dfrac{T}{2}} \tag{B.3.5}$$

Dies hat den Vorteil, dass sämtliche Entwurfsmethoden für die Regelung linearer Strecken wie üblich angewendet werden können. Als Beispiel sei hier das klassische Frequenzkennlinien-Verfahren genannt. Ein wesentlicher Nachteil dieser Vorgangsweise zeigt sich jedoch bei Betrachtung des Frequenzganges. Da die Approximation der Totzeit wie die exakte Formulierung einen Allpass darstellt, gibt es im Betragsgang keine Unterschiede. Im Gegensatz dazu unterscheidet sich der Phasengang jedoch sehr wohl. Bei einer Totzeit von $T = 250\,\mathrm{ms}$ zeigt die Phase der Padé-Approximation des Totzeitglieds nach Beziehung (B.3.3) im Vergleich zum exakten Totzeitglied folgendes Frequenzverhalten:

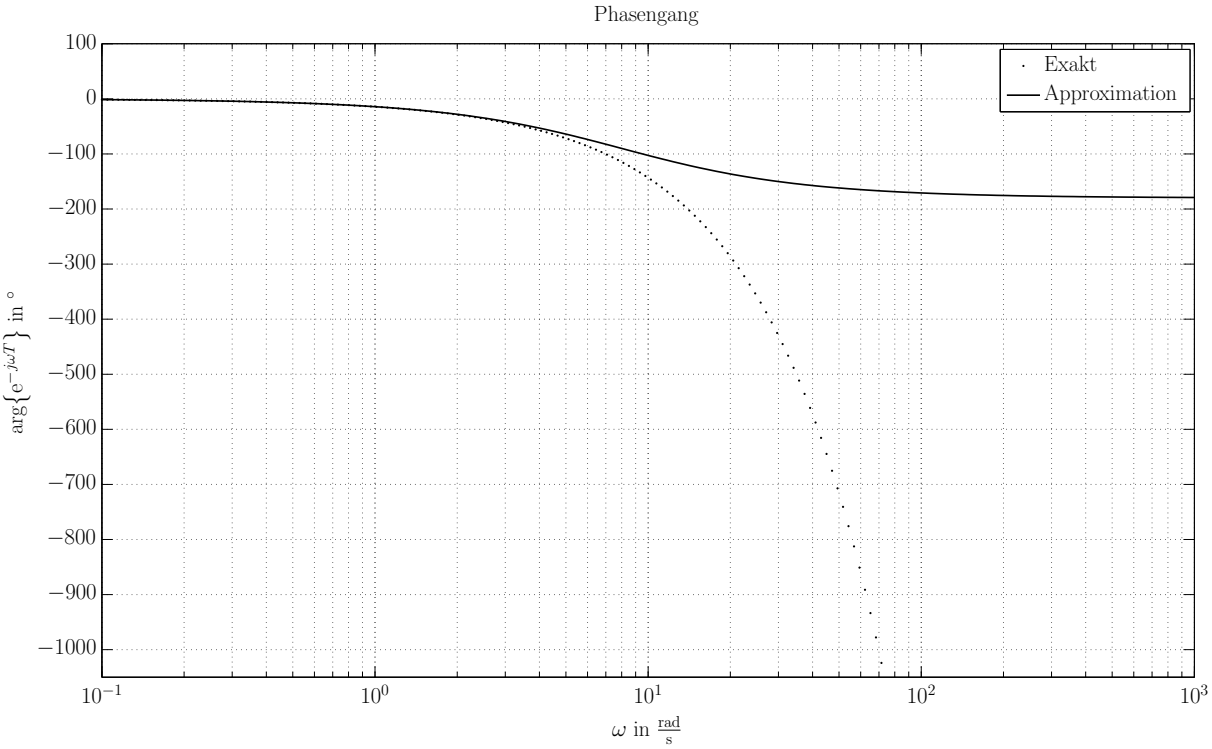

Abbildung 6: Vergleich des Phasengangs eines Totzeitglieds (exakt - approximiert)

Man erkennt, dass der Phasengang der Approximation nur für niedrige Frequenzen mit dem Phasengang eines exakten Totzeitglieds übereinstimmt. Dies bedeutet, dass jegliche Reglerentwürfe unter Verwendung der Padé-Approximation eines Totzeitglieds nur im unteren Frequenzbereich das gewünschte Verhalten aufweisen werden. In der Praxis ist also darauf zu achten, dass die tatsächliche Bandbreite einer auf diese Art und Weise entworfenen Regelung geringer sein wird als durch die Padé-Approximation suggeriert.

3.2.2 Smith-Prädiktor

Eine weiterer Ansatz zur Berücksichtigung von Totzeiten ist der Smith-Prädiktor [3]. Dieser verfolgt das Ziel, einen Regler zu entwerfen, der bewirkt, dass sich ein geregeltes System mit einer Totzeit im offenen Kreis gleich verhält wie ein äquivalentes System, bei welchem die Totzeit nicht im offenen Kreis, sondern erst im Ausgangszweig auftritt. Das Verhalten des Regelkreises nach Abbildung 1 soll also identisch zum Verhalten des folgenden Regelkreises sein:

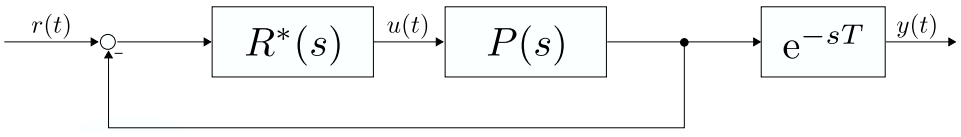

Abbildung 7: Äquivalente Regelkreis-Struktur für Smith-Prädiktor

Diese Struktur hat den großen Vorteil, dass hier das Totzeitglied kein Teil des offenen Kreises ist. Sämtliche Nachteile, die eine Totzeit innerhalb des offenen Kreises bezüglich der Stabilität mit sich bringt, sind in dieser Struktur also nicht mehr von Bedeutung. Zudem kann man den Regler $R^*(s)$ im Zuge dieser Herangehensweise so entwerfen, als ob keine Totzeit vorhanden wäre. Für die Berechnung von $R^*(s)$ berücksichtigt man also nur $P(s)$ und lässt den Totzeitterm zunächst außer Acht. Damit die beiden Strukturen (Abbildung 1 und Abbildung 7) in ihrem Verhalten, also bezüglich der Führungsübertragungsfunktion $T(s)$ zueinander äquivalent sind, muss anschließend gelten:

$$\frac{R\left(s\right)P\left(s\right)\mathrm{e}^{-sT}}{1+R\left(s\right)P\left(s\right)\mathrm{e}^{-sT}} = \frac{R^*\left(s\right)P\left(s\right)}{1+R^*\left(s\right)P\left(s\right)}\mathrm{e}^{-sT} \tag{B.3.6}$$

Die Beziehung (B.3.6) kann man nun auf $R(s)$ umformen, um den gewünschten Regler zu erhalten. Man erhält:

$$R\left(s\right) = \frac{R^*\left(s\right)}{1+R^*\left(s\right)P\left(s\right)\left(1-\mathrm{e}^{-sT}\right)} \tag{B.3.7}$$

Diese Berechnungsvorschrift für den Regler bezeichnet man als Smith-Prädiktor. Wird ein solcher Prädiktor zur Regelung einer realen, totzeitbehafteten Strecke eingesetzt, dann erhält man nach kurzer Umformung folgende Regelstruktur:

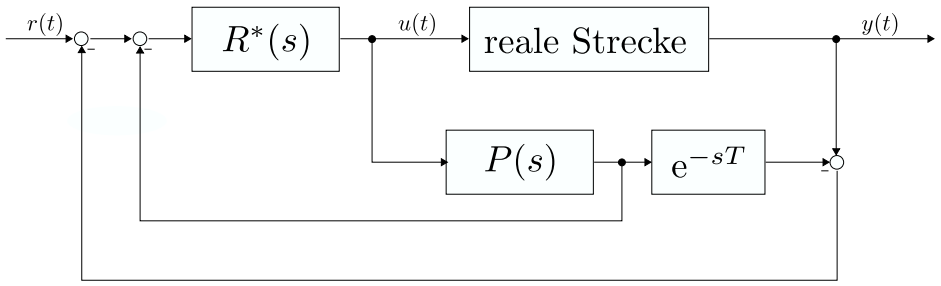

Abbildung 8: Regelstruktur mit Smith-Prädiktor

Man kann erkennen, dass der Term $P(s)\,\mathrm{e}^{-sT}$ in dieser Struktur einem Modell der realen, totzeitbehafteten Strecke entspricht. Stimmt dieses Modell exakt mit der Realität überein, so wird deutlich, dass der äußere Rückkopplungszweig in Abbildung 8 wegfällt, da die Differenz zwischen realem Streckenausgang und Modellausgang in diesem Fall zu jeder Zeit gleich null ist. Der Wegfall der angesprochenen Rückkopplung hat zur Folge, dass die reale Strecke zwingend BIBO-stabil sein muss. Dies stellt eine wesentliche Einschränkung für die Anwendbarkeit des Smith-Prädiktors dar.

Wie schon angedeutet, gibt es neben Padé-Approximation und Smith-Prädiktor noch zahlreiche weitere Ansätze zur Berücksichtigung von Totzeiten in geregelten Systemen – diese basieren auf unterschiedlichsten Herangehensweisen. Die allermeisten dieser Methoden stimmen jedoch in einem Punkt überein: Sie gehen von einer konstanten und bekannten Totzeit aus. Nun kann die Totzeit in einigen realen Systemen tatsächlich als konstant angenommen wären – die Voraussetzung, dass die Totzeit zudem bekannt ist, ist jedoch oft nicht gegeben. Deshalb ist es bei allen Regelentwurfs-Methoden für Totzeit-Systeme notwendig, die auftretende Totzeit in geeigneter Art und Weise zu bestimmen. Ist die Annahme einer konstanten Totzeit gerechtfertigt, so kann dies durch entsprechende vorangehende Experimente vollzogen werden. Für den Fall, dass solche Experimente aus bestimmten Gründen nicht möglich sind oder dass die Totzeit nicht konstant ist, besteht jedoch die Notwendigkeit einer Abschätzung der Totzeit während des laufenden Betriebes. Genau an diesem Punkt setzt die vorliegende Arbeit an: Die Entwicklung einer Methode zur On-Line-Totzeitschätzung.

C Signalbasierte Methoden

In diesem Kapitel werden drei signalbasierte Ansätze zur On-Line-Schätzung von Totzeiten in zeitdiskreten Systemen vorgestellt. Die erste Methode basiert auf der Sprungantwort eines dynamischen Systems, anschließend wird ein korrelativer Ansatz erläutert und zuletzt kommt eine Methode beruhend auf adaptiver Filterung zur Sprache. Abschließend folgt ein kurzes Fazit.

1 Methode basierend auf der Sprungantwort

Eine einfache Methode zur Identifikation der Totzeit eines dynamischen Systems basiert auf der Analyse der Sprungantwort [4]. Hierfür geht man davon aus, dass sich das betrachtete, totzeitbehaftete System in Ruhe befindet und am Eingang ein Sprung angelegt wird. Aus dem Zeitverzug zwischen dem Anlegen des Eingangssprungs und erstmaliger Ausgangsänderung erhält man unter Berücksichtigung des Systemcharakters (Modellordnung) die Totzeit des Systems. Diese Vorgehensweise ist grundsätzlich für zeitdiskrete Systeme aller Art möglich, bei nicht BIBO-stabilen Systemen sollte jedoch statt einem Eingangssprung einen Rechteckpuls endlicher Länge am Eingang angelegt werden. Als einfaches Beispiel zur Veranschaulichung dieser Methode seien folgende drei zeitdiskrete Systeme angenommen, welche jeweils eine Totzeit von $d = 6$ aufweisen:

$$P_0\Big(\, z \,\Big) = 1 z^{-6}$$

$$P_1\Big(\, z \,\Big) = \frac{0.1}{z - 0.9}\, z^{-6} \tag{C.1.1}$$

$$P_2\Big(\, z \,\Big) = \frac{0.01}{\Big(\, z - 0.9 \,\Big)\Big(\, z - 0.9 \,\Big)}\, z^{-6}$$

Augenscheinlich handelt es sich um lineare Systeme nullter, erster und zweiter Ordnung, wobei alle drei Systeme BIBO-Eigenschaft aufweisen. Am Eingang wird nun ein Einheitssprung zum Zeitschritt $k = 20$ angelegt:

$$u_k = \begin{cases} 0 \text{ für } k < 20 \\ 1 \text{ für } k \geq 20 \end{cases} \tag{C.1.2}$$

Abbildung 9 zeigt den Eingangsverlauf und die zugehörigen Ausgangsverläufe für die Systeme P_0, P_1 und P_2:

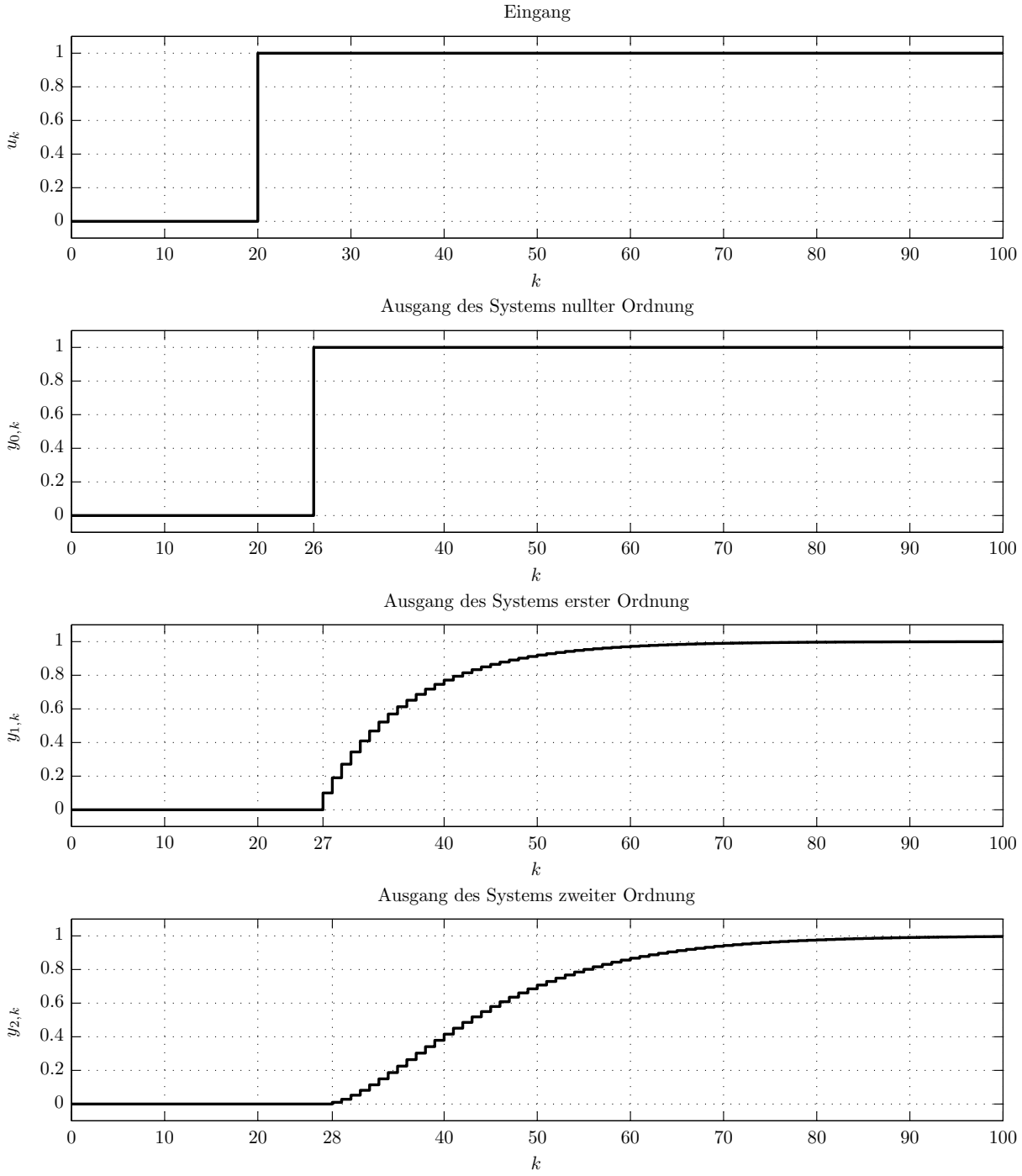

Abbildung 9: Ausgangsverläufe verschiedener totzeitbehafteter Systeme

Man erkennt, dass sich der Ausgang beim System nullter Ordnung zum ersten Mal bei $k = 26$, beim System erster Ordnung zum ersten Mal bei $k = 27$ und beim System zweiter Ordnung zum ersten Mal bei $k = 28$ ändert. Die Zeitverzüge zwischen Eingangssprung und Ausgangsänderung betragen also $D = 6$, $D = 7$ bzw. $D = 8$. Die Detektion dieser Zeitverzüge könnte durch eine entsprechende Schwellwerterken-

nung sowohl am Eingang als auch am Ausgang relativ einfach implementiert werden. Die angeführten Differenzzeiten lassen jedoch schon erahnen, dass zur Ermittlung der Totzeit auch die Systemordnung mit berücksichtigt werden muss. Genauer gesagt muss von der errechneten Zeitdifferenz noch der Differenzgrad $n - m$ gemäß der Systembeschreibung nach (B.2.7) abgezogen werden. Die Berechnungsvorschrift zur Bestimmung der geschätzten Totzeit $\hat{d}$ nach dieser Methode lautet also:

$$\hat{d} = D - \Big(n - m \Big) \tag{C.1.3}$$

Anhand dieser Formel könnte die Totzeit im obigen Beispiel für alle drei Systeme korrekt identifiziert werden. Anzumerken ist, dass in (C.1.3) bereits Modellwissen zur Anwendung kommt. Diese eigentlich signalbasierte Methode kommt also nicht ohne die Hinzunahme von Modellwissen aus.

Aus praktischer Sicht ergeben sich für die sprungantwort-basierte Methode jedoch einige Schwierigkeiten. So ist man in nahezu allen praktischen Anwendungen mit Ausgangsrauschen konfrontiert, dass durch verschiedenste Rauschquellen hervorgerufen wird. Es ist leicht einzusehen, dass dies eine einfache Schwellwertdetektion deutlich schwieriger macht, da die Detektion bei zu gering gewähltem Schwellwert nicht nur beim Auftreten einer Ausgangsänderung bedingt durch den Eingangssprung, sondern auch bei rauschbedingten Ausgangsänderungen anschlagen würde. Abhilfe würde hier ein Schwellwert S schaffen, der sich an der Standardabweichung σ_y des Ausgangsrauschens orientiert, also z.B. $S = 10\,\sigma_y$. Der Wert für σ_y müsste hierbei entweder durch Vorwissen festgelegt werden oder in rekursiver Form zusätzlich mitgeschätzt werden. In [5] geschieht dies beispielsweise durch Verwendung exponentieller Filter. Klar ist jedoch auch, dass diese Methode zur Totzeitschätzung beim Auftreten von Ausgangsrauschen keine derart eindeutigen Ergebnisse wie noch im anfangs erwähnten Beispiel liefern kann. Die Methode wird eher zu einer zu groß geschätzten Totzeit neigen, da die erste feststellbare Reaktion des Ausgangssignals auf den Eingangssprung möglicherweise im Rauschen „untergeht“.

Ein weiteres Problem der sprungantwort-basierten Methode ist ihre Beschränkung auf sprungförmige Eingangsgrößen. Die Methode könnte also nur bei Systemen verwendet werden, bei denen vereinzelt Eingangssprünge auftreten und das System zwischen den Eingangssprüngen immer einen stationären Zustand erreicht. Dies stellt einen schwerwiegenden Nachteil dieser Herangehensweise dar, da sie die Zielsetzung einer universell einsetzbaren Methode für jegliche Eingangsgrößen nicht erfüllt.

2 Korrelative Methode

Eine sehr gängige Methode zur Ermittlung zeitlicher Verzögerungen ist die Schätzung mit Hilfe der Kreuzkorrelation [6]. Diese Methode kommt insbesondere in der Audiotechnik und in der schallbasierten Abstandsmessung zum Einsatz, wo man die Zeitverzögerung zwischen einem Signal und einem proportionalen, jedoch zeitlich verzögerten Signal ermitteln möchte - diese Verzögerung wird oftmals als *Time-Delay of Arrival (TDOA)* bezeichnet. Es sei darauf hingewiesen, dass die in Folge angeführten Betrachtungen in zeitkontinuierlicher Form stattfinden, eine zeitdiskrete Implementierung ist jedoch ohne Probleme möglich. Die angesprochene Verzögerung kann man als Totzeit eines Systems nullter Ordnung interpretieren. Das Ausgangssignal $y(t)$ ist eine um die Totzeit T verzögerte, proportionale Version des Eingangssignals $u(t)$. Tritt zudem sowohl am Eingang als auch am Ausgang Rauschen ($n_u(t)$ bzw. $n_y(t)$) auf, dann liegen die beiden folgenden Signale zur Korrelationsberechnung vor:

$$\begin{aligned} x_1\big(t\big) &= u\big(t\big) + n_u\big(t\big) \\ x_2\big(t\big) &= y\big(t\big) + n_y\big(t\big) = Ku\big(t-T\big) + n_y\big(t\big) \end{aligned} \tag{C.2.1}$$

Für die Korrelation $R_{x_1x_2}(\tau)$ zwischen $x_1(t)$ und $x_2(t)$ gilt dann unter Annahme von gegenseitiger Unkorreliertheit zwischen $u(t)$, $n_u(t)$ und $n_y(t)$:

$$\begin{aligned} R_{x_1x_2}\big(\tau\big) &= \mathrm{E}\Big\{ x_1\big(t-\tau\big)x_2\big(t\big)\Big\} = \int_{t=-\infty}^{+\infty} x_1\big(t-\tau\big)x_2\big(t\big)dt \\ &= \int_{t=-\infty}^{+\infty} \Big(u\big(t-\tau\big) + n_u\big(t-\tau\big)\Big)\Big(Ku\big(t-T\big) + n_y\big(t\big)\Big)dt \\ &= \int_{t=-\infty}^{+\infty} u\big(t-\tau\big)Ku\big(t-T\big)dt + \underbrace{\int_{t=-\infty}^{+\infty} u\big(t-\tau\big)n_y\big(t\big)dt}_{=0} + \\ &\quad \underbrace{\int_{t=-\infty}^{+\infty} n_u\big(t-\tau\big)Ku\big(t-T\big)dt}_{=0} + \underbrace{\int_{t=-\infty}^{+\infty} n_u\big(t-\tau\big)n_y\big(t\big)dt}_{=0} \\ &= K\int_{t=-\infty}^{+\infty} u\big(t-\tau\big)u\big(t-T\big)dt \end{aligned} \tag{C.2.2}$$

Dieser Ausdruck ist jedoch äquivalent zu einer Faltung der Korrelation $R_{uu}(\tau)$ mit einem Delta-Impuls $\delta(\tau - T)$:

$$R_{x_1x_2}\left(\tau\right) = K \int_{t=-\infty}^{+\infty} u\left(t-\tau\right) u\left(t-T\right) dt = K R_{uu}\left(\tau\right) * \delta\left(\tau - T\right) \tag{C.2.3}$$

Da $R_{uu}(\tau)$ für nicht periodische Funktionen sein Maximum bei $\tau = 0$ annimmt, kann man in weiterer Folge leicht zeigen, dass die Korrelation $R_{x_1x_2}(\tau)$ ihr Maximum bei $\tau = T$, also genau bei der gesuchten Totzeit aufweist. In der Praxis kann die Korrelation $R_{x_1x_2}(\tau)$ jedoch nur über einen gewissen Zeitraum geschätzt werden:

$$\hat{R}_{x_1x_2}(\tau) = \int_{t=0}^{t_1} x_1\left(t-\tau\right) x_2\left(t\right) dt \tag{C.2.4}$$

Dies lässt sich realisieren, indem man das Signal $x_1(t)$ um die Zeit τ verzögert, mit $x_2(t)$ multipliziert und anschließend über den Zeitraum $t = 0$ bis $t = \tau$ integriert. Führt man diese Schritte für verschiedene τ im Bereich $[\tau_{\min}, \tau_{\max}]$ durch, so ergibt sich die geschätzte Totzeit wie folgt:

$$\hat{T} = \arg\max \hat{R}_{x_1x_2}\left(\tau\right) \;\forall\; \tau \in \left[\,\tau_{\min}, \tau_{\max}\,\right] \tag{C.2.5}$$

Die Variation der Zeit τ zur Maximierung von $\hat{R}_{x_1x_2}(\tau)$ lässt sich mit Hilfe einer geeigneten Optimierung bewerkstelligen. Dies wird in Abbildung 10 verdeutlicht:

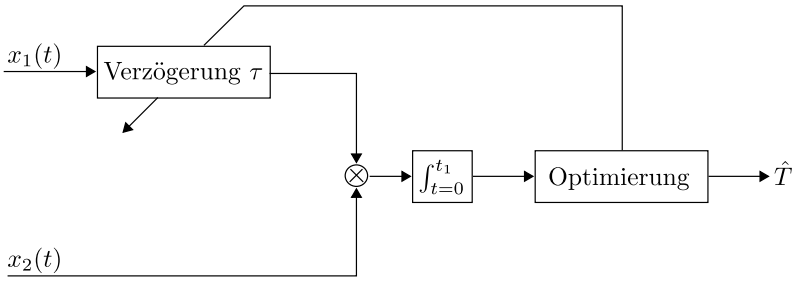

Abbildung 10: Totzeitschätzung mittels Korrelation

Aus Beziehung (C.2.3) kann man ableiten, dass der Delta-Impuls $\delta(\tau - T)$ durch die Autokorrelation $R_{uu}(\tau)$ sozusagen „verschmiert“ wird. Auf die Lage des Maximums von $R_{x_1x_2}(\tau)$ hätte dies bei exakter Berechnung keinen Einfluss. Da $R_{x_1x_2}(\tau)$ jedoch wie erwähnt nur durch Beziehung (C.2.4) approximiert werden kann, kann dies zu dem Fall führen, dass das Maximum nicht mehr eindeutig an der Stelle $\tau = T$ identifizierbar ist. Zur Abhilfe kann man sich in diesem Fall die Definition der Kreuzkorrela-

tion $\hat{R}_{x_1x_2}(\tau)$ als inverse Fourier-Transformation des Kreuzleistungsdichtespektrums $\hat{G}_{x_1x_2}(\omega)$ zu Nutze machen:

$$\hat{R}_{x_1x_2}(\tau) = \int_{\omega=-\infty}^{+\infty} \hat{G}_{x_1x_2}\Big(\,\omega\,\Big)\mathrm{e}^{j\omega\tau}\,d\omega \tag{C.2.6}$$

Durch die Verwendung geeigneter Gewichtsfunktionen $W(\omega)$ kann man über diesen Umweg erreichen, dass sich bei $\tau = T$ wieder ein scharfes und klar definiertes Maximum der Kreuzkorrelation $\hat{R}_{x_1x_2}(\tau)$ ergibt – für Einzelheiten hierzu sei auf [6] verwiesen. Die geschätzte Totzeit $\hat{T}$ ergibt sich in diesem Fall wie folgt:

$$\begin{aligned} \tilde{R}_{x_1x_2}(\tau) &= \int_{\omega=-\infty}^{+\infty} W\Big(\,\omega\,\Big)\hat{G}_{x_1x_2}\Big(\,\omega\,\Big)\mathrm{e}^{j\omega\tau}\,d\omega \\ \hat{T} &= \arg\max \tilde{R}_{x_1x_2}\Big(\,\tau\,\Big)\;\forall\;\tau \in \Big[\,\tau_{\min},\tau_{\max}\,\Big] \end{aligned} \tag{C.2.7}$$

Bezüglich der Wahl der Gewichtungsfunktion $W(\omega)$ existieren in der Literatur verschiedenste Vorschläge, [6] schafft hier ebenfalls einen guten Überblick.

Bezüglich der vorgestellten korrelativen Methode zur Totzeitschätzung ist anzumerken, dass diese für Systeme nullter Ordnung bzw. für reine Verzögerungsschätzungen zwar sehr brauchbar sein kann, jedoch kann man diese Herangehensweise bei dynamischen Systemen höherer Ordnung nicht mehr sinnvoll verwenden. Auch dieser Ansatz erfüllt also nicht die Anforderung der universellen Einsetzbarkeit.

3 Methode mittels adaptiver Filterung

Eine Methode, die für die Schätzung der reinen Verzögerungszeit eines Signals ebenfalls denkbar ist, ist die Verwendung eines adaptiven Filters [7]. Als adaptiven Filter versteht man in dieser Anwendung ein digitales FIR-Filter mit Länge L, dessen Koeffizienten $w_0,\ldots,w_{L-1}$ durch einen geeigneten Algorithmus adaptiv eingestellt werden, also zeitlich variabel sind. Als Berechnungsgrundlagen erhält der Algorithmus dabei die Eingangsgröße des adaptiven Filters und den Fehler zwischen gewünschtem Filterausgang und tatsächlichem Filterausgang. Der Algorithmus stellt die Koeffizienten des Filters dann genau so ein, dass eben dieser Fehler zu null wird. Bei Verwendung eines solchen Filters für eine Verzögerungsschätzung dient das nicht verzögerte Signal u_k als Filtereingang und das verzögerte Signal $y_k = u_{k-d}$ stellt das gewünschte Signal am Ausgang dar. Es ergibt sich also folgende Struktur:

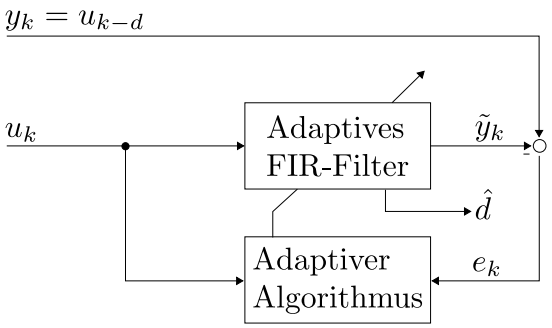

Abbildung 11: Adaptives Filter zur Totzeitschätzung

Ein korrekt funktionierender adaptiver Algorithmus wird in obiger Struktur alle der L Koeffizienten des Filtervektors $\mathbf{w} := (\ w_0 \quad \cdots \quad w_{L-1}\ )^T$ bis auf genau einen an der Stelle d sukzessive auf Null setzen und der Koeffizient an der Stelle d wird den Wert Eins annehmen. Dadurch tritt am Ausgang des adaptiven Filters ebenfalls $y_k = u_{k-d}$ auf, wodurch der Fehler $e_k = y_k - \tilde{y}_k$ verschwindet. Der Filtervektor $\mathbf{w}$ wird nach ausreichender Zeit also folgendes Aussehen haben:

$$\mathbf{w} \approx (\ \underset{\substack{\uparrow \\ 0}}{0} \quad \ldots \quad 0 \quad \underset{\substack{\uparrow \\ d}}{1} \quad 0 \quad \ldots \quad \underset{\substack{\uparrow \\ L-1}}{0}\ )^T \tag{C.3.1}$$

Es kann also von der Position des Koeffizienten mit dem Wert von annähernd Eins auf die Totzeit $\hat{d}$ geschätzt werden. Das Entscheidende an dieser Herangehensweise ist ein funktionstüchtiger adaptiver Algorithmus. Hierfür betrachtet man zunächst das Fehlersignal e_k, dass sich wie folgt berechnet:

$$e_k = y_k - \tilde{y}_k = y_k - \sum_{i=0}^{L-1} w_i u_{k-i} = u_{k-d} - \sum_{i=0}^{L-1} w_i u_{k-i} \tag{C.3.2}$$

Wie erläutert, besteht das Ziel des Algorithmus darin, das Fehlersignal zu minimieren. Dies erreicht man durch Minimierung folgender Gütefunktion:

$$J = \mathrm{E}\left\{ e_k^2 \right\} = \mathrm{E}\left\{ \left(u_{k-d} - \sum_{i=0}^{L-1} w_i u_{k-i} \right)^2 \right\} \tag{C.3.3}$$

Das Minimum der Gütefunktion (C.3.3) liegt an der Stelle, an der die Ableitungen nach den Filterkoeffizienten gleich Null sind. Damit folgt:

$$\frac{\partial \mathrm{E}\left\{ e_k^2 \right\}}{\partial w_j} = 0 \quad \forall \quad j = 0,\ldots,L-1$$

$$\begin{aligned}
\mathrm{E}\left\{ 2e_k \frac{\partial}{\partial w_j} \left(u_{k-d} - \sum_{i=0}^{L-1} w_i u_{k-i} \right) \right\} &= 0 \quad \forall \quad j = 0,\ldots,L-1 \\
2\,\mathrm{E}\left\{ e_k \left(-u_{k-j} \right) \right\} &= 0 \quad \forall \quad j = 0,\ldots,L-1 \\
\mathrm{E}\left\{ e_k u_{k-j} \right\} &= 0 \quad \forall \quad j = 0,\ldots,L-1
\end{aligned} \tag{C.3.4}$$

Dies bedeutet, dass bei einem optimalen adaptiven Filter das Fehlersignal e_k orthogonal und damit unkorreliert zum Eingangssignal u_k ist (Orthogonalitätsprinzip). Zudem gilt:

$$\begin{aligned}
\mathrm{E}\left\{ e_k \tilde{y}_k \right\} &= \mathrm{E}\left\{ e_k \sum_{i=0}^{L-1} w_i u_{k-i} \right\} \\
&= \sum_{i=0}^{L-1} w_i \,\mathrm{E}\left\{ e_k u_{k-i} \right\}
\end{aligned} \tag{C.3.5}$$

Somit ist das Fehlersignal bei optimalen Filterkoeffizienten auch zum Ausgangssignal $\tilde{y}_k$ orthogonal und unkorreliert. Daher bezeichnet man ein optimales Schätzfilter dieser Art auch oft als Dekorrelations-Filter. Für die Berechnung des optimalen Filtervektors $\mathbf{w}_{\mathrm{opt}}$ bietet sich die Vektorschreibweise des Signals u_k , d.h. $\mathbf{u}_k = (\; u_k \quad u_{k-1} \quad \cdots \quad u_{k-(L-1)} \;)^T$ an. Für die Gütefunktion J ergibt sich damit:

$$\begin{aligned}
J &= \mathrm{E}\left\{ e_k^2 \right\} \\
&= \mathrm{E}\left\{ \left(u_{k-d} - \mathbf{w}^T \mathbf{u}_k \right)^2 \right\} \\
&= \mathrm{E}\left\{ u_{k-d}^2 \right\} - 2\,\mathrm{E}\left\{ u_{k-d} \mathbf{w}^T \mathbf{u}_k \right\} + \mathrm{E}\left\{ \mathbf{w}^T \mathbf{u}_k \mathbf{u}_k^T \mathbf{w} \right\} \\
&= \mathrm{E}\left\{ u_{k-d}^2 \right\} - 2\mathbf{w}^T\,\mathrm{E}\left\{ u_{k-d} \mathbf{u}_k \right\} + \mathbf{w}^T\,\mathrm{E}\left\{ \mathbf{u}_k \mathbf{u}_k^T \right\} \mathbf{w} \\
&= \sigma_u^2 - 2\mathbf{w}^T \mathbf{r}_{uu} + \mathbf{w}^T \mathbf{R}_{uu} \mathbf{w}
\end{aligned} \tag{C.3.6}$$

Hier kennzeichnet σ_u^2 die Varianz des Signals u_k, $\mathbf{r}_{uu}$ den zugehörigen Autokorrelationsvektor und $\mathbf{R}_{uu}$ die Autokorrelationsmatrix. Um nun den optimalen Koeffizientenvektor $\mathbf{w}_{\text{opt}}$ für das FIR-Filter zu erhalten, bildet man wiederum die entsprechende Ableitung der Gütefunktion und setzt diese gleich null:

$$\begin{aligned} \left.\frac{\partial J}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}_{\text{opt}}} &= \mathbf{0} \\ \left.\frac{\partial}{\partial \mathbf{w}}\Big(\sigma_u^2 - 2\mathbf{w}^T\mathbf{r}_{uu} + \mathbf{w}^T\mathbf{R}_{uu}\mathbf{w} \Big)\right|_{\mathbf{w}=\mathbf{w}_{\text{opt}}} &= \mathbf{0} \\ -2\mathbf{r}_{uu} + 2\mathbf{R}_{uu}\mathbf{w}_{\text{opt}} &= \mathbf{0} \\ \mathbf{w}_{\text{opt}} &= \mathbf{R}_{uu}^{-1}\mathbf{r}_{uu} \end{aligned} \tag{C.3.7}$$

Diese Berechnungsvorschrift entspricht der Wiener-Lösung bzw. der zeitdiskreten Form der Wiener-Hopf-Gleichung [8]. Notwendige Bedingung ist eine reguläre Autokorrelationsmatrix $\mathbf{R}_{uu}$. Die Schwierigkeit in Beziehung (C.3.7) liegt nun darin, dass der Autokorrelationsvektor $\mathbf{r}_{uu}$ und die Autokorrelationsmatrix $\mathbf{R}_{uu}$ in den meisten Fällen nicht bekannt sind. Durch einen geeigneten Algorithmus wird es aber möglich, den optimalen Koeffizientenvektor $\mathbf{w}_{\text{opt}}$ iterativ zu bestimmen. Hierfür ist die Methode des steilsten Abstiegs [9] eine gängige Wahl. Dabei initialisiert man den Koeffizientenvektor mit $\mathbf{w}_0$ (z.B. $\mathbf{w}_0 = \mathbf{0}$) und iteriert dann Schritt für Schritt in die Richtung des steilsten Abstiegs der Gütefunktion J - diese Richtung entspricht dem negativen Gradienten:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \frac{\partial J}{\partial \mathbf{w}_k} = \mathbf{w}_k - \mu \frac{\partial \mathrm{E}\left\{ e_k^2 \right\}}{\partial \mathbf{w}_k} \tag{C.3.8}$$

Hier entspricht μ der Schrittweite des Algorithmus. Diese kann entweder fix festgelegt oder durch einen weiterführenden Algorithmus adaptiv angepasst werden. Da der Erwartungswert von e_k^2 ebenfalls nicht bekannt ist, versucht man diesen geeignet zu ersetzen. Im Falle des LMS-Algorithmus (*Least Mean Squares*) [8] geschieht dies durch die ersatzweise Verwendung des jeweils aktuellen Wertes e_k^2. Dadurch ergibt sich mit Hilfe der Kettenregel:

$$\begin{aligned}\mathbf{w}_{k+1} &= \mathbf{w}_k - \mu \frac{\partial e_k^2}{\partial \mathbf{w}_k} \\ &= \mathbf{w}_k - \mu 2 e_k \frac{\partial \Big(u_{k-d} - \mathbf{w}_k^T \mathbf{u}_k \Big)}{\partial \mathbf{w}_k} \\ &= \mathbf{w}_k + 2\mu e_k \mathbf{u}_k\end{aligned} \tag{C.3.9}$$

Der Koeffizientenvektor des adaptiven Filters zum Zeitschritt $k+1$ ergibt sich also durch Addition des Koeffizientenvektors $\mathbf{w}_k$ zum vorherigen Zeitschritt mit dem Produkt aus der doppelten Schrittweite μ, dem Fehler e_k und dem Datenvektor $\mathbf{u}_k$. Bei stationären Signalen u_k und y_k konvergiert dass adaptive FIR-Filter gegen das Wiener-Filter $\mathbf{w}_{\text{opt}} = -\mathbf{R}_{uu}^{-1}\mathbf{r}_{uu}$, es gilt also $\lim_{k\to\infty} \mathbf{w}_k = \mathbf{w}_{\text{opt}}$. Bei nicht stationären Signalen u_k und y_k folgt das Filterausgangssignal $\tilde{y}_k$ dem Signal y_k, wodurch sich im Anwendungsfall der Totzeitschätzung nach ausreichender Zeit $k >> k_0$ der ebenfalls Koeffizientenvektor $\mathbf{w}$ nach (C.3.1) ergibt.

Die Methode der adaptiven Filterung stellt also ein gutes Werkzeug zur Schätzung von Verzögerungszeiten zwischen zwei ansonsten identischen Signalen dar. Für dynamische Systeme erster oder höherer Ordnung ist sie jedoch nicht brauchbar, weshalb auch hier der Anspruch einer universell einsetzbaren Methode nicht erfüllt wird.

4 Zusammenfassung

In diesem Kapitel wurden drei ausgewählte signalbasierte Methoden zur Totzeitschätzung näher erläutert. Hierbei wurden ausschließlich Methoden, welche on-line, also während des laufenden Betriebes einsetzbar sind, berücksichtigt. Alle drei vorgestellten Ansätze weisen jedoch entscheidende Nachteile in Bezug auf die Zielsetzung dieser Arbeit auf, d.h. keiner von ihnen erfüllt das Ziel einer universell einsetzbaren Methode für beliebige Systeme und Anregungssignale. Die Methode basierend auf der Sprungantwort dynamischer Systeme kann zwar für unterschiedlichste Systemcharakteristiken eingesetzt werden, sie ist jedoch – wie der Name schon vermuten lässt – auf Eingangssprünge als Anregungssignal beschränkt. Die korrelative Methode weist dieses Problem zwar nicht auf, jedoch kann sie nur bei Systemen nullter Ordnung, also reinen Verstärkungen sinnvoll angewendet werden. Eine korrelative Methode, die auch bei Systemen höherer Ordnung von Nutzen ist, wird von Elnaggar et al. in [10] und [11] vorgeschlagen. Hier wird für alle möglichen Totzeiten $d \in [d_{\min}, d_{\max}]$ eine Gütefunktion ausgewertet, die sich aus der Korrelation zwischen dem um d Samples

verzögerten Eingangswert u_{k-d} und der aktuellen Ausgangsänderung $y_k - y_{k-1}$ berechnet. Diese Herangehensweise wurde in mehreren Experimenten jedoch ebenfalls als nicht universell einsetzbar befunden – sie liefert nur bei Systemdynamiken mit viel Phasenreserve und vergleichsweise kleiner Abtastzeit gute Schätzergebnisse. Zuletzt wurde noch eine Methode mittels adaptiver Filterung dargelegt. Diese ist jedoch ebenfalls nur eingeschränkt von Nutzen, da sie ausschließlich für die Schätzung reiner Verzögerungszeiten zwischen zwei identischen Signalen in Frage kommt. Es konnte also keine signalbasierte Methode gefunden werden, welche die gestellten Ansprüche vollständig erfüllt.

D Modellbasierte Methoden

Dieses Kapitel dient einer eingehenden Betrachtung modellbasierter Methoden zur On-Line-Totzeitschätzung für zeitdiskrete Systeme. Es kommen drei unterschiedliche Ansätze zur Sprache: Zunächst wird eine Methode besprochen, bei welcher die Totzeit anhand des Verhaltens zweier Typen von Kalman-Filtern in Folge sprungförmiger Eingangssignale identifiziert wird. Anschließend folgt eine gradientenbasierte Methode, welche versucht, Modellparameter und Totzeit als gemeinsame Bestandteile eines Schätzvektors im Verbund zu schätzen. Der letzte Ansatz basiert auf Systemidentifikation mit Hilfe rekursiver Least-Squares-Algorithmen, wobei hier unterschiedlichste Varianten zur Bestimmung der Totzeit existieren.

1 Methode basierend auf Kalman-Filtern

Wie schon kurz erwähnt, versucht diese modellbasierte Methode die Totzeit eines zeitdiskreten Systems anhand zweier unterschiedlicher Varianten des Kalman-Filters zu bestimmen [12]. Die erste Variante des Kalman-Filters (1) dient dabei der Festlegung einer Obergrenze der Totzeit, während mehrere Ausführungen der zweiten Variante (2) zur anschließenden Schätzung der Totzeit verwendet werden. Es ist anzumerken, dass zur Sicherstellung der Funktionstüchtigkeit dieser Methode ein sprungförmiges Eingangssignal angelegt werden muss.

1.1 Kalman-Filter

Da bei der Totzeitschätzung nach dieser Methode Kalman-Filter zum Einsatz kommen, wird die grundlegende Theorie hier der Vollständigkeit halber kurz wiedergegeben. Das lineare, zeitinvariante System mit Eingang u_k und Ausgang y_k, dessen Zustände $\mathbf{x}_k$ geschätzt werden sollen, hat folgende Struktur:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{\Phi}\mathbf{x}_k + \mathbf{h}u_k + \mathbf{w}_k \\ y_k &= \mathbf{c}^T\mathbf{x}_k + v_k \end{aligned} \tag{D.1.1}$$

Hier kennzeichnet $\mathbf{\Phi}$ die Dynamikmatrix, $\mathbf{h}$ den Eingangsvektor, $\mathbf{w}_k$ das Zustandsrauschen, $\mathbf{c}$ den Ausgangsvektor und v_k das Messrauschen. Die stochastischen Prozesse $\mathbf{w}_k$ und v_k seien als weiß, mittelwertfrei und unkorreliert angenommen:

$$\mathrm{E}\Big\{\,\mathbf{w}_k\,\Big\} = \mathbf{0} \quad ; \quad \mathrm{E}\Big\{\,\mathbf{w}_k\mathbf{w}_j^T\,\Big\} = \begin{cases} \mathbf{Q} & \text{für } k = j \\ \mathbf{0} & \text{für } k \neq j \end{cases}$$

$$\mathrm{E}\Big\{\,v_k\,\Big\} = 0 \quad ; \quad \mathrm{E}\Big\{\,v_k v_j\,\Big\} = \begin{cases} r & \text{für } k = j \\ 0 & \text{für } k \neq j \end{cases} \tag{D.1.2}$$

$$\mathrm{E}\Big\{\,\mathbf{w}_k v_j\,\Big\} = 0$$

$\mathbf{Q}$ ist die Kovarianzmatrix des Zustandsrauschens $\mathbf{w}_k$ und r ist die Varianz des Ausgangsrauschens v_k. Die fünf Kalman-Filter-Gleichungen lauten dann:

$$\begin{aligned} \mathbf{x}_k^* &= \mathbf{\Phi}\hat{\mathbf{x}}_{k-1} + \mathbf{h}u_{k-1} \\ \mathbf{P}_k^* &= \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{Q} \\ \mathbf{k}_k &= \mathbf{P}_k^*\mathbf{c}\Big(\,\mathbf{c}^T\mathbf{P}_k^*\mathbf{c} + r\,\Big)^{-1} \\ \hat{\mathbf{x}}_k &= \mathbf{x}_k^* + \mathbf{k}_k\Big(\,y_k - \mathbf{c}^T\mathbf{x}_k^*\,\Big) \\ \mathbf{P}_k &= \Big(\,\mathbf{E} - \mathbf{k}_k\mathbf{c}^T\,\Big)\mathbf{P}_k^* \end{aligned} \tag{D.1.3}$$

Für den Rekursionsbeginn müssen der Erwartungswert des a-priori Schätzwerts von $\mathbf{x}_0$, also $\mathbf{x}_0^* = \mathrm{E}\{\mathbf{x}_0\}$ und die Kovarianzmatrix des Fehlers des prädizierten Schätzwerts, also $\mathbf{P}_0^* = \mathrm{E}\{(\mathbf{x}_0^* - \mathbf{x}_0)(\mathbf{x}_0^* - \mathbf{x}_0)^T\}$ geeignet initialisiert werden. Dann können in jeder Iteration schrittweise der prädizierte Schätzvektor $\mathbf{x}_k^*$, die Kovarianzmatrix des Fehlers des prädizierten Schätzvektors $\mathbf{P}_k^*$, der Korrekturvektor $\mathbf{k}_k$, der Schätzvektor $\hat{\mathbf{x}}_k$ und die Schätzfehler-Kovarianzmatrix $\mathbf{P}_k$ ermittelt werden.

1.2 Ermittlung der Obergrenze für die Totzeit

Zur Ermittlung der Obergrenze $d_{\max}$ für die Totzeit wird ein Kalman-Filter (1) verwendet, für welches das folgende einfache Modell angenommen wird:

$$\begin{aligned} x_{k+1}^{(1)} &= x_k^{(1)} \\ y_k^{(1)} &= x_k^{(1)} + v_k^{(1)} \end{aligned} \tag{D.1.4}$$

Das Modell hat also nur einen Zustand x_k und entsprechend der Systembeschreibung nach (D.1.1) gilt $\phi^{(1)} = 1\,,\ h^{(1)} = 0$ und $c^{(1)} = 1\,.$ Es tritt außerdem kein Zustandsrauschen $\mathbf{w}_k^{(1)}$ auf und für das Ausgangsrauschen $v_k^{(1)}$ sei $r^{(1)} = \sigma^2_{v^{(1)}}$ angenommen. Damit ergibt sich für die Kalman-Filter-Gleichungen (D.1.3) folgendes:

$$\begin{aligned} x_k^{(1)^*} &= \hat{x}_{k-1}^{(1)} \\ P_k^{(1)^*} &= P_{k-1}^{(1)} \\ k_k^{(1)} &= \frac{P_k^{(1)^*}}{P_k^{(1)^*} + \sigma^2_{v^{(1)}}} \\ \hat{x}_k^{(1)} &= x_k^{(1)^*} + k_k^{(1)} \Big(y_k - x_k^{(1)^*} \Big) \\ P_k^{(1)} &= \Big(1 - k_k^{(1)} \Big) P_k^{(1)^*} \end{aligned} \tag{D.1.5}$$

Nach einigen Umformungen stellt sich heraus, dass der Kalman-Filter für dieses einfache System durch eine einzige Gleichung beschrieben kann, in die neben dem aktuellen Ausgangswert $y_k^{(1)}$ und dem letzten Schätzwert $\hat{x}_{k-1}^{(1)}$ lediglich die Varianz $\sigma^2_{v^{(1)}}$ des Ausgangsrauschen und die Schätzfehler-Varianz $P_0^{(1)}$ eingeht:

$$\hat{x}_k^{(1)} = \frac{kP_0^{(1)} + \sigma^2_{v^{(1)}}}{\Big(k+1 \Big) P_0^{(1)} + \sigma^2_{v^{(1)}}}\, \hat{x}_{k-1}^{(1)} + \frac{P_0^{(1)}}{\Big(k+1 \Big) P_0^{(1)} + \sigma^2_{v^{(1)}}}\, y_k \tag{D.1.6}$$

Sowohl $\sigma^2_{v^{(1)}}$ als auch $P_0^{(1)}$ können beispielsweise durch eine Initialisierungsphase vor dem Anliegen des Eingangssprungs zum Zeitpunkt d_0 auf geeignete Art und Weise geschätzt werden. Als Startwert für den Schätzwert $\hat{x}_k^{(1)}$ wählt man den Mittelwert des Ausgangssignals $y_k^{(1)}$ während der Initialisierungsphase. Um nun eine Obergrenze $d_{\max}$ für die Totzeit festzustellen, lässt man das obige Kalman-Filter ab dem Zeitpunkt des Eingangssprungs d_0 solange laufen, bis sich die Modellannahme eines konstanten Zustands $x_k^{(1)}$ als falsch erweist. Dies ist wiederum durch geeignete statistische Methoden möglich, bei welchen man prüft, wann das normalisierte Residuum, also der normalisierte Schätzfehler des Kalman-Filters einen bestimmten Vertrauensbereich verlässt. Genaueres hierzu findet man in [12].

1.3 Schätzung der Totzeit

Ab dem Zeitpunkt d_0 des Auftretens des Eingangssprungs startet man in jedem Zeitschritt ein Kalman-Filter der zweiten Variante (2) – so lange bis die durch ein Kalman-Filter erster Variante (1) ermittelte Obergrenze $d_{\max}$ für die zu schätzende Totzeit erreicht ist. Die zweite Variante modelliert ein System mit zwei Zuständen, wobei man $x_1^{(2)}$ als „Position“ und $x_2^{(2)}$ als „Steigung“ des Systems bezeichnen könnte:

$$\left.\begin{array}{c} x_{1,k+1}^{(2)} = x_{1,k}^{(2)} + h x_{2,k}^{(2)} \\ x_{2,k+1}^{(2)} = x_{2,k}^{(2)} \end{array}\right\} \quad \mathbf{x}_{k+1}^{(2)} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \mathbf{x}_k^{(2)}$$
$$y_k^{(2)} = x_{1,k}^{(2)} + v_k^{(2)} \tag{D.1.7}$$

Hier tritt wie beim Kalman-Filter des ersten Typs kein Zustandsrauschen $\mathbf{w}_k^{(2)}$ auf und für das Ausgangsrauschen $v_k^{(2)}$ sei auch hier $r^{(2)} = \sigma_{v^{(2)}}^2$ angenommen. Neben der abzulesenden Dynamikmatrix $\mathbf{\Phi}^{(2)}$ ist der Eingangsvektor durch $\mathbf{h}^{(2)} = \mathbf{0}$ und der Ausgangsvektor durch $\mathbf{c}^{(2)T} = \begin{pmatrix} 1 & 0 \end{pmatrix}$. Außerdem kennzeichnet h die Abtastzeit. Die Kalman-Filter-Gleichungen lauten somit:

$$\mathbf{x}_k^{(2)*} = \begin{pmatrix} \hat{x}_{1,k-1}^{(2)} + h\hat{x}_{2,k-1}^{(2)} \\ \hat{x}_{2,k-1}^{(2)} \end{pmatrix}$$

$$\mathbf{P}_k^{(2)*} = \begin{bmatrix} p_{11,k-1}^{(2)} + h p_{21,k-1}^{(2)} + h p_{12,k-1}^{(2)} + h^2 p_{22,k-1}^{(2)} & p_{12,k-1}^{(2)} + h p_{22,k-1}^{(2)} \\ p_{21,k-1}^{(2)} + h p_{22,k-1}^{(2)} & p_{22,k-1}^{(2)} \end{bmatrix}$$

$$\mathbf{k}_k^{(2)} = \begin{pmatrix} \dfrac{p_{11,k}^{(2)*}}{p_{11,k}^{(2)*} + \sigma_{v^{(2)}}^2} \\ \dfrac{p_{21,k}^{(2)*}}{p_{11,k}^{(2)*} + \sigma_{v^{(2)}}^2} \end{pmatrix} \tag{D.1.8}$$

$$\hat{\mathbf{x}}_k^{(2)} = \begin{pmatrix} x_{1,k}^{(2)*} + k_{1,k}^{(2)} \left(y_k - x_{1,k}^{(2)*} \right) \\ x_{2,k}^{(2)*} + k_{2,k}^{(2)} \left(y_k - x_{1,k}^{(2)*} \right) \end{pmatrix}$$

$$\mathbf{P}_k^{(2)} = \begin{bmatrix} \left(1 - k_{1,k}^{(2)}\right) p_{11,k}^{(2)*} & \left(1 - k_{1,k}^{(2)}\right) p_{12,k}^{(2)*} \\ -k_{2,k}^{(2)} p_{11,k}^{(2)*} + p_{21,k}^{(2)*} & -k_{2,k}^{(2)} p_{12,k}^{(2)*} + p_{22,k}^{(2)*} \end{bmatrix}$$

Wie erwähnt, wird nun zu jedem Abtastschritt beginnend mit dem Zeitpunkt des Eingangssprungs ein solches Kalman-Filter zweiten Typs (2) gestartet. Die Initialisierungswerte orientieren sich dabei an den aktuellen Werten des bereits laufenden Kalman-Filters ersten Typs (1) :

$$\begin{pmatrix} \hat{x}_{1,0}^{(2)} \\ \hat{x}_{2,0}^{(2)} \end{pmatrix} = \begin{pmatrix} \hat{x}_k^{(1)} \\ 0 \end{pmatrix} \quad ; \quad \mathbf{P}_0^{(2)*} = \begin{bmatrix} P_k^{(1)*} & 0 \\ 0 & p_{22,0}^{(2)*} \end{bmatrix} \tag{D.1.9}$$

Die Wahl von $p_{22,0}^{(2)*}$ gestaltet sich etwas komplizierter, da dieser Wert abhängig von Varianz $\sigma_{v^{(2)}}^2$ sein sollte. Für Einzelheiten sei hier wiederum auf [12] verwiesen. Um nun zu einer Totzeitschätzung zu gelangen, bewertet man alle bis zur Obergrenze der Totzeit gestarteten Kalman-Filter der zweiten Variante hinsichtlich ihrer Modellgüte, wobei die Modellgüte des Kalman-Filters erster Variante bis zum Start des jeweiligen Kalman-Filters ebenfalls berücksichtigt wird. Hierfür bietet sich folgende Kostenfunktion an:

$$J(d) = \sum_{k=d_0}^{d_0+d-1} \Big(y_k - \hat{x}_k^{(1)} \Big)^2 + \sum_{k=d_0+d}^{d_{\max}} \Big(y_k - \hat{x}_{1,k}^{(2)} \Big)^2, \quad d = 0, \ldots, d_{\max} - d_0 \tag{D.1.10}$$

Die geschätzte Totzeit $\hat{d}$ erhält man dann aus der minimalen zugehörigen Gütefunktion $J(d)$:

$$\hat{d} = \arg\min \Big\{ J\Big(d \Big) \Big\} \tag{D.1.11}$$

Durch diese Methode lässt sich also die Totzeit eines Systems bei Anregung durch einen Eingangssprung schätzen, wobei dies bei mehreren aufeinanderfolgenden Eingangssprüngen auch während des laufenden Betriebs möglich ist. Dies stellt jedoch wie schon bei der sprungantwort-basierten signalbasierten Methode (Kapitel C1) eine Einschränkung dar: Dadurch, dass ein Sprung der einzig mögliche Eingangssignalverlauf für die Anwendbarkeit dieser Methode ist, kann sie die Forderung nach einer universellen, für jegliche Anregungssignale anwendbaren Methode nicht erfüllen.

2 Gradientenbasierte Methode

Eine weitere Möglichkeit zur On-Line-Totzeitschätzung bei zeitdiskreten Systemen ist die gradientenbasierte Methode nach Bedoui et al. [13]. Diese beruht darauf, dass Modellparameter und Totzeit als gemeinsame Bestandteile eines Schätzvektors im Verbund geschätzt werden. Dies geschieht iterativ, wobei die Iterationsrichtung je-

weils durch den negativen Gradienten einer geeigneten Gütefunktion bestimmt ist. Zur Herleitung dieser Methode betrachtet man zunächst die Differenzengleichung eines mit einer variablen Totzeit behafteten linearen zeitinvarianten Systems. Diese ergibt sich aus der Übertragungsfunktion (B.2.7) wie folgt:

$$y_k = b_m u_{k+m-n-d_k} + b_{m-1} u_{k+m-1-n-d_k} \ldots + b_0 u_{k-n-d_k} - a_{n-1} y_{k-1} - \ldots - a_0 y_{k-n} \quad \text{(D.2.1)}$$

Es lässt sich schreiben:

$$y_k = \sum_{i=0}^{m} b_i u_{k+i-n-d_k} - \sum_{i=0}^{n-1} a_i y_{k+i-n} \quad \text{(D.2.2)}$$

Zur Vereinfachung fasst man die Messwerte und die Modellparameter zusammen:

$$\phi_k = \begin{pmatrix} -y_{k-1} \\ \vdots \\ -y_{k-n} \\ u_{k+m-n-d_k} \\ \vdots \\ u_{k-n-d_k} \end{pmatrix} \quad ; \quad \boldsymbol{\theta} = \begin{pmatrix} a_{n-1} \\ \vdots \\ a_0 \\ b_m \\ \vdots \\ b_0 \end{pmatrix} \quad \text{(D.2.3)}$$

Somit erhält man:

$$y_k = \phi_k^T \boldsymbol{\theta} \quad \text{(D.2.4)}$$

Wie angesprochen beinhaltet der Schätzvektor bei dieser Methode nicht nur die Schätzwerte der Modellparameter, sondern auch die geschätzte Totzeit $\hat{d}$. Er wird daher als generalisierter Schätzvektor $\hat{\boldsymbol{\theta}}_G$ bezeichnet und hat folgende Struktur:

$$\hat{\boldsymbol{\theta}}_G = \begin{pmatrix} \hat{a}_{n-1} \\ \vdots \\ \hat{a}_0 \\ \hat{b}_m \\ \vdots \\ \hat{b}_0 \\ \hat{d} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\theta}} \\ \hat{d} \end{pmatrix} \quad \text{(D.2.5)}$$

Um nun den optimalen Schätzvektor iterativ zu bestimmen, bewegt man sich in jedem Iterationsschritt k in Richtung des negativen Gradienten einer zu minimierenden Kostenfunktion J_k:

$$\hat{\boldsymbol{\theta}}_{G,k} = \hat{\boldsymbol{\theta}}_{G,k-1} - \mu_k \frac{\partial J_k}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \tag{D.2.6}$$

Die Variable μ_k wird als Schrittweite bezeichnet – näheres zur geeigneten Wahl von μ_k wird später beschrieben. Für die Wahl einer geeigneten Kostenfunktion bietet sich der Schätzfehler e_k des Modells zum jeweiligen Zeitpunkt k an. Man wählt:

$$J_k = \frac{1}{2} e_k^2 = \frac{1}{2}\Big(y_k - \hat{y}_k \Big)^2 = \frac{1}{2}\Bigg(y_k - \begin{pmatrix} \hat{\boldsymbol{\phi}}_k^T & 0 \end{pmatrix} \hat{\boldsymbol{\theta}}_{G,k-1} \Bigg)^2 \tag{D.2.7}$$

Man erkennt, dass für den Schätzwert $\hat{y}_k$ nicht der wahre Messvektor $\boldsymbol{\phi}_k$, sondern $\hat{\boldsymbol{\phi}}_k$ verwendet wird. Dies liegt daran, dass die für die Bildung von $\boldsymbol{\phi}_k$ nach (D.2.3) die wahre Totzeit d_k nicht zur Verfügung steht – diese wird ja gerade geschätzt. Daher muss man diese durch die zuletzt geschätzte Totzeit $\hat{d}_{k-1}$ ersetzen. Die Schreibweise $\hat{\boldsymbol{\phi}}_k$ ergibt sich also aus $\hat{\boldsymbol{\phi}}_k = \boldsymbol{\phi}_k \Big|_{d_k = \hat{d}_{k-1}}$. Für die Ableitung der Gütefunktion (D.2.7) nach $\hat{\boldsymbol{\theta}}_{G,k-1}$ erhält man damit:

$$\begin{aligned}
\frac{\partial J_k}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} &= \frac{1}{2} \frac{\partial e_k^2}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} = e_k \frac{\partial e_k}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \\
&= e_k \frac{\partial}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \Bigg(y_k - \begin{pmatrix} \hat{\boldsymbol{\phi}}_k^T & 0 \end{pmatrix} \hat{\boldsymbol{\theta}}_{G,k-1} \Bigg) \\
&= e_k \frac{\partial}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \Big(y_k - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big) \\
&= -e_k \frac{\partial}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \Bigg(\sum_{i=0}^{m} \hat{b}_{i,k-1} u_{k+i-n-\hat{d}_{k-1}} - \sum_{i=0}^{n-1} \hat{a}_{i,k-1} y_{k+i-n} \Bigg) \\
&= -e_k \begin{pmatrix} \hat{\boldsymbol{\phi}}_k \\ \dfrac{\partial}{\partial \hat{d}_{k-1}} \Bigg(\displaystyle\sum_{i=0}^{m} \hat{b}_{i,k-1} u_{k+i-n-\hat{d}_{k-1}} - \sum_{i=0}^{n-1} \hat{a}_{i,k-1} y_{k+i-n} \Bigg) \end{pmatrix}
\end{aligned} \tag{D.2.8}$$

Der letzte Eintrag des obigen Vektors, also die Ableitung des Schätzwertes $\hat{y}_k$ nach der zuletzt geschätzten Totzeit $\hat{d}_{k-1}$ kann wegen den diskret vorliegenden Messwerten nicht exakt angegeben werden. Es muss also eine geeignete Approximation ermittelt werden. Diese könnte beispielsweise wie folgt aussehen:

$$\frac{\partial}{\partial \hat{d}_{k-1}}\left(\sum_{i=0}^{m} \hat{b}_{i,k-1} u_{k+i-n-\hat{d}_{k-1}} - \sum_{i=0}^{n-1} \hat{a}_{i,k-1} y_{k+i-n} \right) = \frac{\partial}{\partial \hat{d}_{k-1}}\left(\sum_{i=0}^{m} \hat{b}_{i,k-1} u_{k+i-n-\hat{d}_{k-1}} \right)$$

$$\approx \sum_{i=0}^{m} \hat{b}_{i,k-1} \left(\frac{u_{k+i-n-\hat{d}_{k-1}-1} - u_{k+i-n-\hat{d}_{k-1}}}{\left(\hat{d}_{k-1} + 1 \right) - \hat{d}_{k-1}} \right) \tag{D.2.9}$$

$$= \sum_{i=0}^{m} \hat{b}_{i,k-1} \left(u_{k+i-n-\hat{d}_{k-1}-1} - u_{k+i-n-\hat{d}_{k-1}} \right)$$

In die Iterationsvorschrift (D.2.6) eingesetzt erhält man:

$$\hat{\boldsymbol{\theta}}_{G,k} = \hat{\boldsymbol{\theta}}_{G,k-1} - \mu_k \frac{\partial J_k}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}}$$

$$= \hat{\boldsymbol{\theta}}_{G,k-1} - \mu_k \left(-e_k \left(\begin{array}{c} \hat{\boldsymbol{\phi}}_k \\ \sum_{i=0}^{m} \hat{b}_{i,k-1} \left(u_{k+i-n-\hat{d}_{k-1}-1} - u_{k+i-n-\hat{d}_{k-1}} \right) \end{array} \right) \right) \tag{D.2.10}$$

$$= \hat{\boldsymbol{\theta}}_{G,k-1} + \mu_k e_k \left(\begin{array}{c} \hat{\boldsymbol{\phi}}_k \\ \sum_{i=0}^{m} \hat{b}_{i,k-1} \left(u_{k+i-n-\hat{d}_{k-1}-1} - u_{k+i-n-\hat{d}_{k-1}} \right) \end{array} \right)$$

Abschließend stellt sich noch die Frage nach der Wahl von μ_k. In [13] wird gezeigt, dass bei konstanter Totzeit $d_k = d$ durch folgende Wahl von μ_k Konvergenz des Schätzvektors $\hat{\boldsymbol{\theta}}_G$ garantiert ist:

$$0 < \mu_k < \frac{2}{\hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\phi}}_k} \tag{D.2.11}$$

Daher wird in weiterer Folge $\mu_k = \frac{1}{\hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\phi}}_k}$ festgelegt. Eingesetzt in (D.2.10) folgt:

$$\hat{\boldsymbol{\theta}}_{G,k} = \hat{\boldsymbol{\theta}}_{G,k-1} + \frac{1}{\hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\phi}}_k} e_k \left(\begin{array}{c} \hat{\boldsymbol{\phi}}_k \\ \sum_{i=0}^{m} \hat{b}_{i,k-1} \left(u_{k+i-n-\hat{d}_{k-1}-1} - u_{k+i-n-\hat{d}_{k-1}} \right) \end{array} \right) \tag{D.2.12}$$

Für den Rekursionsbeginn muss lediglich noch $\hat{\boldsymbol{\theta}}_{G,k}$ geeignet initialisiert werden. Für den Fall, dass a priori kein Wissen über die Modellparameter vorliegt, wird der Schätzvektor oft mit $\hat{\boldsymbol{\theta}}_{G,0} = \mathbf{0}$ initialisiert [14] [15]. Zudem ist darauf zu achten, dass $\hat{d}_k$ eine ganze Zahl sein muss, was durch den Algorithmus an sich nicht gewährleistet ist. Daher rundet man $\hat{d}_k$ einfach in jedem Schritt auf den nächstgelegenen Integer-Wert. Als Beispiel zur Demonstration der Funktionstüchtigkeit dieser Methode betrachtet man nun folgendes System erster Ordnung mit konstanter Totzeit:

$$G\Big(\, z \,\Big) = \frac{0.5}{z - 0.8}\, z^{-5} \tag{D.2.13}$$

Gemäß (B.2.7) beträgt hier der Nennergrad $n = 1$, der Zählergrad $m = 0$ und die konstante Totzeit $d_k = d = 5$. Als Anregungssignal wird weißes Rauschen mit Standardabweichung $\sigma_u = 1$ verwendet und der Algorithmus wird mit den Initialwerten $\hat{\boldsymbol{\theta}}_{G,0} = (\ \hat{a}_{0,0} \quad \hat{b}_{0,0} \quad \hat{d}_0\ )^T = (\ 0 \quad 0 \quad 0\ )^T$ gestartet. Folgende Verläufe für die geschätzten Parameter und die geschätzte Totzeit ergeben sich:

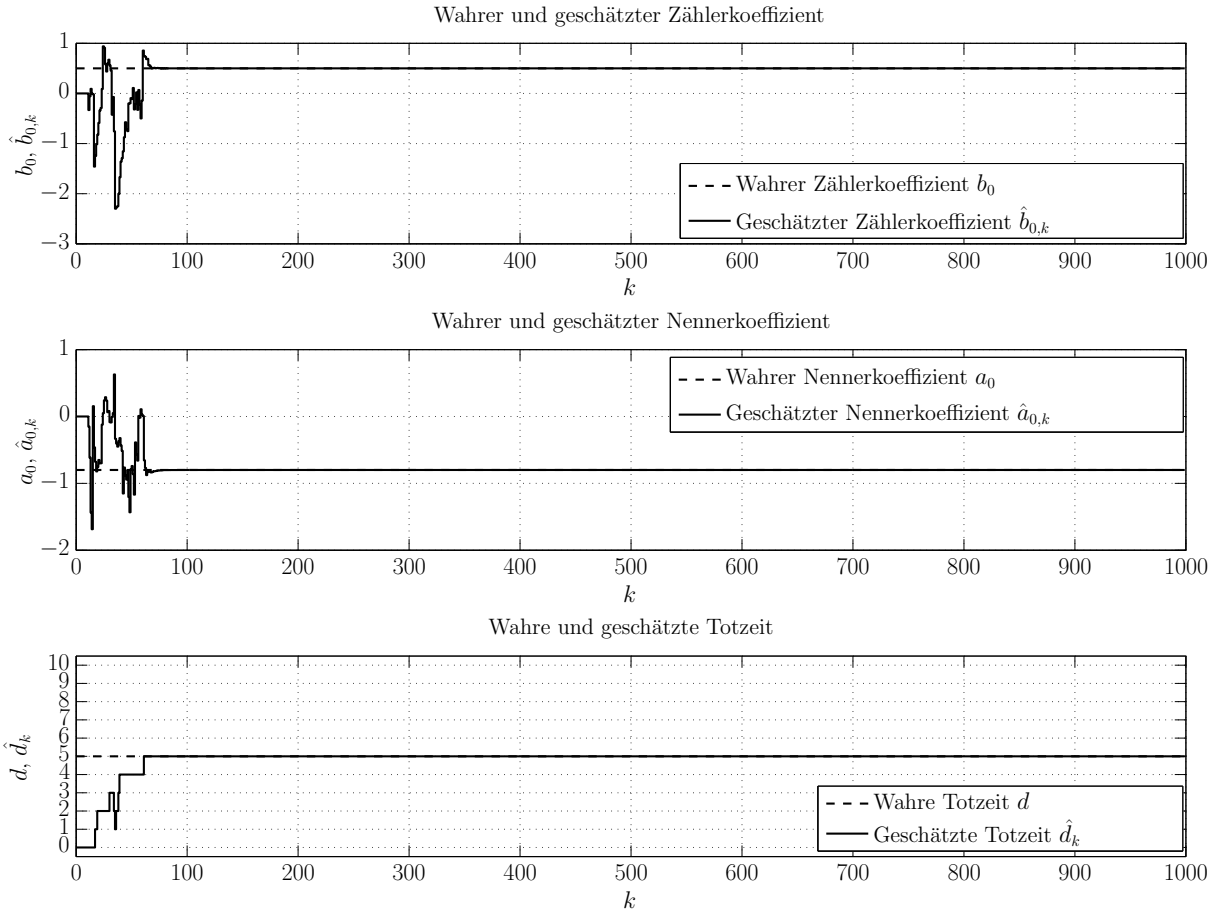

Abbildung 12: Beispiel gradientenbasierte Methode - konstante Totzeit

Aus Abbildung 12 ist ersichtlich, dass sowohl die Modellparameter $\hat{b}_k$ und $\hat{a}_k$ als auch die Totzeit $\hat{d}_k$ schon nach kurzer Zeit richtig geschätzt werden. In einem weiteren Beispiel soll nun das gleiche lineare zeitdiskrete System, nun jedoch mit zeitvariabler Totzeit untersucht werden. Die Totzeit d_k steigt hier treppenförmig bis auf den Wert 6 und sinkt anschließend wieder ab (gestrichelter Verlauf in Abbildung 13):

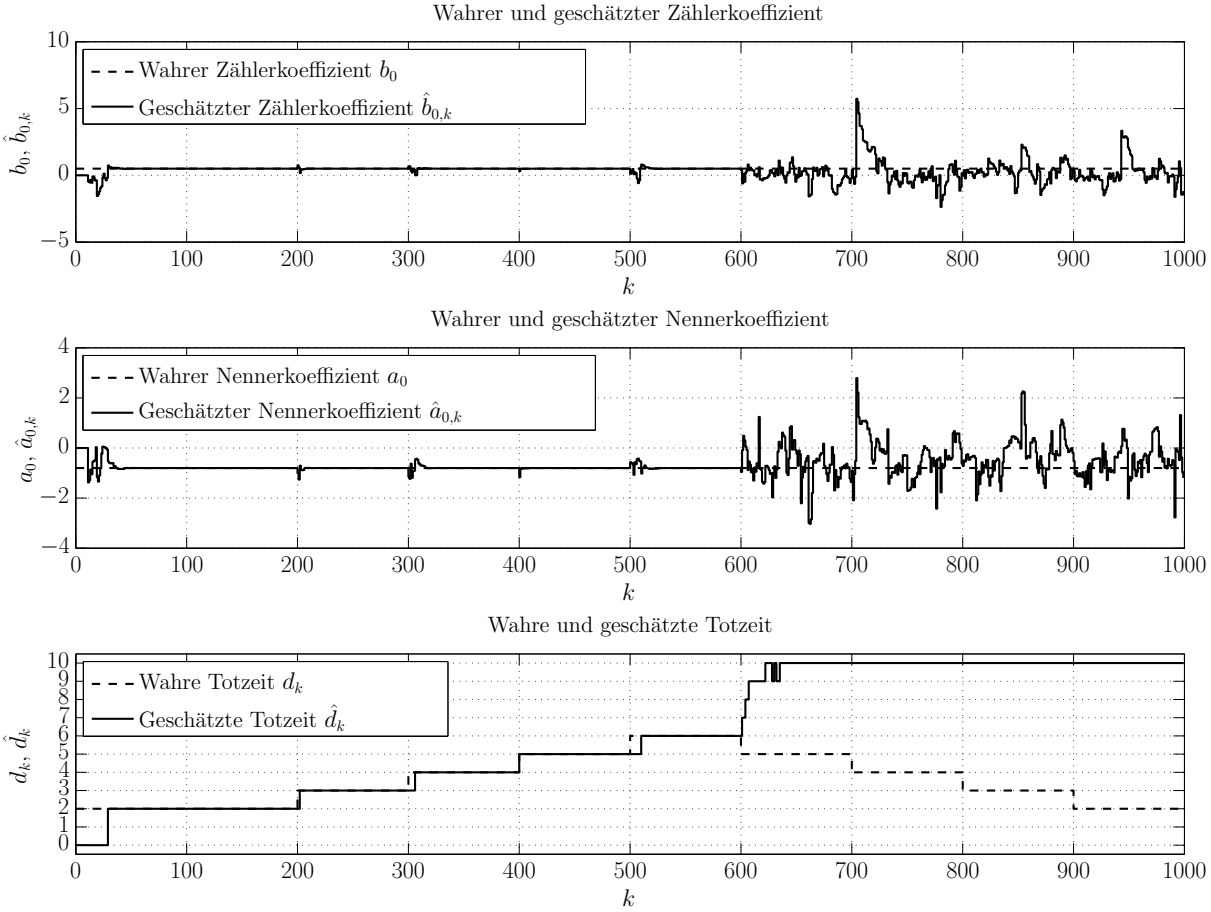

Abbildung 13: Beispiel gradientenbasierte Methode - variable Totzeit

Dieses Beispiel mit zeitvariabler Totzeit zeigt, dass die Methode für eine steigende Totzeit zwar gut funktioniert, jedoch schlägt die Totzeitschätzung bei sinkender Totzeit fehl. Dieses Verhalten wurde in zahlreichen Versuchen immer wieder festgestellt. Eine Begründung könnte in der diskreten approximierten Ableitung des prädizierten Werts $\hat{y}_k$ nach der zuletzt geschätzten Totzeit $\hat{d}_{k-1}$ liegen (D.2.9). Diese verhält sich im Falle sinkender Totzeiten ungünstig. Bei Zählerordnung $m = 0$ wird beispielsweise die Differenz $u_{k-n-\hat{d}_{k-1}-1} - u_{k-n-\hat{d}_{k-1}}$ herangezogen, obwohl es intuitiv sinnvoller wäre, bei sinkender Totzeit $\hat{d}_k = \hat{d}_{k-1} - 1$ auch den Wert $u_{k-n-\hat{d}_{k-1}+1}$ miteinzubeziehen. Dies könnte man beispielsweise (wieder für $m = 0$) durch Approximation der Ableitung mit der Differenz $u_{k-n-\hat{d}_{k-1}} - u_{k-n-\hat{d}_{k-1}+1}$ oder $0.5(u_{k-n-\hat{d}_{k-1}-1} - u_{k-n-\hat{d}_{k-1}+1})$ erreichen. Diese und auch weitere Approximationen wurden anhand mehrerer Beispiel ausgiebig getes-

tet, jedoch brachte keine der verwendeten Varianten gute Ergebnisse sowohl bei steigenden als auch bei sinkenden Totzeiten.

Zusammenfassend lässt sich sagen, dass die hier vorgestellte gradientenbasierte Methode eine sinnvolle Herangehensweise bei linearen Systemen mit konstanten Totzeiten ist, jedoch ergeben sich bei zeitvariablen Totzeiten, insbesondere bei sinkenden Totzeiten erhebliche Probleme. Außerdem ist anzumerken, dass für eine gute Qualität der Totzeitschätzung ein Anregungssignal nötig ist, dass möglichst viele Frequenzen enthält, also beispielsweise weißes Rauschen. Dies ist jedoch in der Praxis nicht realisierbar. Bei sinusförmigen Anregungen (eine Frequenz) konnten auch konstante Totzeiten nicht zuverlässig identifiziert werden. Diese beiden Nachteile – die Einschränkung auf konstante Totzeiten und breitbandige Anregung – stellen ein Ausschlusskriterium für diese Methode dar, da sie damit ebenfalls nicht universell einsetzbar ist.

3 Rekursive Least-Squares-Methoden

In diesem Kapitel wird eine Fülle an modellbasierten Methoden zur On-Line-Totzeitschätzung beschrieben, wobei alle diese Methoden auf dem rekursiven Least-Squares-Algorithmus beruhen. Daher wird zunächst die allgemeine Least-Squares-Identifikation und ihre rekursive Ausführung erläutert. Basierend auf dieser Grundlage wird anschließend auf unterschiedliche Ansätze zur Schätzung der Totzeit eingegangen.

3.1 Die Least-Squares-Identifikation

Den ersten Schritt in der Herleitung der Least-Squares-Identifikation [16] bilden einige zusammenfassende Schreibweisen. Der Ausgangsvektor $\mathbf{y}_k$ beinhaltet alle Ausgangswerte vom Zeitpunkt $k = 1$ bis zum Zeitpunkt $k = N$ und für den Fehlervektor $\mathbf{e}_k$ gilt analoges. Der Zeitpunkt $k = N$ ist dabei derjenige Zeitpunkt, bis zu welchem Messdaten vorliegen. Es werden also alle verfügbaren Daten miteinbezogen. Außerdem benötigt man den schon aus (D.2.3) bekannten Parametervektor $\boldsymbol{\theta}$:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \quad ; \quad \mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{pmatrix} \quad ; \quad \boldsymbol{\theta} = \begin{pmatrix} a_{n-1} \\ \vdots \\ a_0 \\ b_m \\ \vdots \\ b_0 \end{pmatrix} \tag{D.3.1}$$

Zudem führt man noch eine Messmatrix $\mathbf{\Phi}$ ein, die alle Messvektoren ϕ_k nach (D.2.3) bis zum Zeitpunkt $k = N$ in Zeilenform enthält:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_N \end{bmatrix}^T = \begin{bmatrix} -y_0 & -y_1 & \cdots & -y_{N-1} \\ \vdots & \vdots & \vdots & \vdots \\ -y_{1-n} & -y_{2-n} & \cdots & -y_{N-n} \\ u_{1+m-n} & u_{2+m-n} & \cdots & u_{N+m-n} \\ \vdots & \vdots & \vdots & \vdots \\ u_{1-n} & u_{2-n} & \cdots & u_{N-n} \end{bmatrix}^T \tag{D.3.2}$$

Es sei darauf hingewiesen, dass für u_k und y_k Kausalität angenommen wird, d.h. $u_k = 0 \;\forall\; k < 0$ und $y_k = 0 \;\forall\; k < 0$. Anhand der obigen Zusammenfassungen ist es nun möglich, das Eingangs-Ausgangs-Verhalten eines linearen zeitdiskreten Systems anhand folgender Gleichung für alle Zeitpunkte $k = 1$ bis $k = N$ zu beschreiben:

$$\mathbf{y} = \mathbf{\Phi\theta} + \mathbf{e} \tag{D.3.3}$$

Will man nun den Parametervektor $\boldsymbol{\theta}$ anhand des Ausgangsvektors $\mathbf{y}$ und der Messmatrix $\mathbf{\Phi}$ schätzen, so erhält man das folgende überbestimmte Gleichungssystem:

$$\mathbf{y} = \mathbf{\Phi}\hat{\boldsymbol{\theta}} + \mathbf{e} \tag{D.3.4}$$

Der Schätzvektors $\hat{\boldsymbol{\theta}}$ beinhaltet die Schätzwerte für die Modellparameter, also $\hat{\boldsymbol{\theta}} = \begin{pmatrix} \hat{a}_{n-1} & \cdots & \hat{a}_0 & \hat{b}_m & \cdots & \hat{b}_0 \end{pmatrix}^T$. Da das überbestimmte Gleichungssystem (D.3.4) nicht eindeutig lösbar ist, versucht man den Fehlervektor $\mathbf{e}$ in geeigneter Art und Weise zu minimieren. Im Falle der Least-Squares-Identifikation geschieht dies durch die Minimierung der Fehlerquadrate:

$$J = e_1^2 + e_2^2 + \ldots + e_N^2 = \mathbf{e}^T\mathbf{e} \quad \rightarrow \quad \min \tag{D.3.5}$$

In die Gütefunktion J setzt man an dieser Stelle den Fehlervektor nach Beziehung (D.3.3) ein:

$$\begin{aligned} J &= \mathbf{e}^T\mathbf{e} \\ &= \Big(\mathbf{y} - \mathbf{\Phi\theta}\Big)^T \Big(\mathbf{y} - \mathbf{\Phi\theta}\Big) \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{\Phi\theta} + \boldsymbol{\theta}^T\mathbf{\Phi}^T\mathbf{\Phi\theta} \end{aligned} \tag{D.3.6}$$

Zur Auffindung des optimalen Schätzvektors $\hat{\boldsymbol{\theta}}$ minimiert man nun die Gütefunktion (D.3.6). Hierzu muss ihre Ableitung nach dem Parametervektor $\boldsymbol{\theta}$ an der Stelle $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ elementweise gleich null sein:

$$\begin{aligned}
\left.\frac{\partial J}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} &= \mathbf{0} \\
\left.\frac{\partial}{\partial \boldsymbol{\theta}}\Big(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\boldsymbol{\Phi}\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\theta} \Big)\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} &= \mathbf{0} \\
\left. -2\boldsymbol{\Phi}^T\mathbf{y} + 2\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\theta}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} &= \mathbf{0} \qquad \text{(D.3.7)} \\
\boldsymbol{\Phi}^T\boldsymbol{\Phi}\hat{\boldsymbol{\theta}} &= \boldsymbol{\Phi}^T\mathbf{y} \\
\hat{\boldsymbol{\theta}} &= \Big(\boldsymbol{\Phi}^T\boldsymbol{\Phi} \Big)^{-1} \boldsymbol{\Phi}^T\mathbf{y}
\end{aligned}$$

Der optimale Schätzvektor $\hat{\boldsymbol{\theta}}$ im Sinne der kleinsten Fehlerquadrate (Least-Squares-Lösung) berechnet sich also durch Multiplikation der Matrix $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T$ mit dem Ausgangsvektor $\mathbf{y}$. Der Ausdruck $(\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T$ wird auch oft als Pseudo-Inverse der Messmatrix $\boldsymbol{\Phi}$ bezeichnet.

3.2 Der rekursive Least-Squares-Algorithmus (RLS-Algorithmus)

Da diese Arbeit auf einen während des laufenden Betriebes einsetzbaren Algorithmus abzielt, kann die Least-Squares-Identifikation nach (D.3.7) nicht direkt eingesetzt werden. Der Grund hierfür liegt darin, dass bei einer On-Line-Implementierung immer nur die Eingangs- und Ausgangswerte bis zum jeweiligen Zeitpunkt, jedoch nicht aus der Zukunft zur Verfügung stehen. Daher muss die Least-Squares-Identifikation zuerst in eine rekursive Form [16] gebracht werden. Hierfür führt man eine zeitliche Indizierung des Schätzvektors, der Messmatrix und des Ausgangsvektors ein, wodurch sich die jeweils aktuellsten Einträge (Zeitschritt k) einfach abspalten lassen. Die geschätzten Modellparameter zum Zeitschritt k berechnen sich damit wie folgt:

$$\hat{\boldsymbol{\theta}}_k = \Big(\boldsymbol{\Phi}_k^T\boldsymbol{\Phi}_k \Big)^{-1} \boldsymbol{\Phi}_k^T\mathbf{y}_k = \left(\begin{bmatrix} \boldsymbol{\Phi}_{k-1} \\ \boldsymbol{\phi}_k^T \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Phi}_{k-1} \\ \boldsymbol{\phi}_k^T \end{bmatrix} \right)^{-1} \begin{bmatrix} \boldsymbol{\Phi}_{k-1} \\ \boldsymbol{\phi}_k^T \end{bmatrix}^T \begin{pmatrix} \mathbf{y}_{k-1} \\ y_k \end{pmatrix} \qquad \text{(D.3.8)}$$

Für die weiteren Betrachtungen führt man folgende Abkürzung ein:

$$\mathbf{P}_k = \Big(\mathbf{\Phi}_k^T \mathbf{\Phi}_k \Big)^{-1} \tag{D.3.9}$$

Für die Matrix $\mathbf{P}_k$ ist im RLS-Algorithmus die Bezeichnung Kovarianzmatrix gebräuchlich. Die Inverse $\mathbf{P}_k^{-1}$ trägt den Namen Informationsmatrix. Für sie gilt:

$$\begin{aligned} \mathbf{P}_k^{-1} &= \begin{bmatrix} \mathbf{\Phi}_{k-1} \\ \boldsymbol{\phi}_k^T \end{bmatrix}^T \begin{bmatrix} \mathbf{\Phi}_{k-1} \\ \boldsymbol{\phi}_k^T \end{bmatrix} \\ &= \mathbf{\Phi}_{k-1}^T \mathbf{\Phi}_{k-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \\ &= \mathbf{P}_{k-1}^{-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \end{aligned} \tag{D.3.10}$$

Eingesetzt in (D.3.8) ergibt sich:

$$\begin{aligned} \hat{\boldsymbol{\theta}}_k &= \mathbf{P}_k \begin{bmatrix} \mathbf{\Phi}_{k-1} \\ \boldsymbol{\phi}_k^T \end{bmatrix}^T \begin{pmatrix} \mathbf{y}_{k-1} \\ y_k \end{pmatrix} \\ &= \mathbf{P}_k \Big(\mathbf{\Phi}_{k-1}^T \mathbf{y}_{k-1} + \boldsymbol{\phi}_k y_k \Big) \\ &= \mathbf{P}_k \Big(\mathbf{P}_{k-1}^{-1} \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{\phi}_k y_k \Big) \\ &= \mathbf{P}_k \Big(\Big(\mathbf{P}_k^{-1} - \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \Big) \hat{\boldsymbol{\theta}}_{k-1} + \boldsymbol{\phi}_k y_k \Big) \\ &= \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{P}_k \boldsymbol{\phi}_k \Big(y_k - \boldsymbol{\phi}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big) \end{aligned} \tag{D.3.11}$$

Damit liegt bereits eine rekursive Berechnungsvorschrift für den Schätzvektor $\hat{\boldsymbol{\theta}}_k$ vor. Er berechnet sich aus dem zuletzt geschätzten Parametervektor $\hat{\boldsymbol{\theta}}_{k-1}$ und einer Korrektur bestehend aus dem um $\mathbf{P}_k \boldsymbol{\phi}_k$ verstärkten Schätzfehler $y_k - \boldsymbol{\phi}_k^T \hat{\boldsymbol{\theta}}_{k-1}$. Aus diesem Grund führt man die Abkürzung $\mathbf{k}_k := \mathbf{P}_k \boldsymbol{\phi}_k$ ein, wobei $\mathbf{k}_k$ für den Korrekturvektor steht. Ein Problem besteht jetzt noch darin, dass die Matrix $\mathbf{P}_k^{-1}$ aus (D.3.10) zur Berechnung von $\hat{\boldsymbol{\theta}}_{k+1}$ in jedem Zeitschritt invertiert werden müsste. Dies birgt Nachteile in der Numerik und der Recheneffizienz. Abhilfe schafft jedoch die *Woodbury-Matrix-Identität* [17] (auch bekannt als *Matrix Inversion Lemma*), durch die sich die Matrix $\mathbf{P}_k$ in jedem Zeitschritt direkt berechnen lässt:

$$\begin{aligned}\mathbf{P}_k &= \left(\mathbf{P}_{k-1}^{-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \right)^{-1} \\ &= \mathbf{P}_{k-1} - \frac{\mathbf{P}_{k-1}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}}{1 + \boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k}\end{aligned} \tag{D.3.12}$$

Für den Korrekturvektor erhält man damit:

$$\begin{aligned}\mathbf{k}_k &= \left(\mathbf{P}_{k-1} - \frac{\mathbf{P}_{k-1}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}}{1 + \boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k} \right)\boldsymbol{\phi}_k \\ &= \mathbf{P}_{k-1}\boldsymbol{\phi}_k - \frac{\mathbf{P}_{k-1}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k}{1 + \boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k} \\ &= \frac{\mathbf{P}_{k-1}\boldsymbol{\phi}_k + \mathbf{P}_{k-1}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k - \mathbf{P}_{k-1}\boldsymbol{\phi}_k\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k}{1 + \boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k} \\ &= \frac{\mathbf{P}_{k-1}\boldsymbol{\phi}_k}{1 + \boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k}\end{aligned} \tag{D.3.13}$$

Dadurch vereinfacht sich Beziehung (D.3.12) wie folgt:

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \mathbf{k}_k\boldsymbol{\phi}_k^T\mathbf{P}_{k-1} \tag{D.3.14}$$

Zusammengefasst ergeben sich also die folgenden drei Gleichungen für den rekursiven Least-Squares-Algorithmus (RLS-Algorithmus):

$$\begin{aligned}\mathbf{k}_k &= \frac{\mathbf{P}_{k-1}\boldsymbol{\phi}_k}{1 + \boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\boldsymbol{\phi}_k} \\ \hat{\boldsymbol{\theta}}_k &= \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k\left(y_k - \boldsymbol{\phi}_k^T\hat{\boldsymbol{\theta}}_{k-1} \right) \\ \mathbf{P}_k &= \mathbf{P}_{k-1} - \mathbf{k}_k\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}\end{aligned} \tag{D.3.15}$$

Die drei Gleichungen aus (D.3.15) bilden eine exakte rekursive Form der Least-Squares-Identifikation, d.h. zum Zeitschritt $k = N$ entspricht der Schätzvektor $\hat{\boldsymbol{\theta}}_N$ genau dem Schätzvektor $\hat{\boldsymbol{\theta}}$ aus (D.3.7).

Es ist das Ziel dieser Arbeit, eine Methode zu adaptieren, die auch zeitlich variable Totzeiten bzw. zeitlich variable Modellparameter schätzen kann. Daher stellt sich der Algorithmus nach (D.3.15) schnell als ungünstig heraus. Hier werden alle Messvektoren $\boldsymbol{\phi}_k$ zu allen Zeitschritten gleich gewichtet, d.h. ein sehr lange zurückliegender Messvektor hat den gleichen Einfluss auf den Schätzvektor $\hat{\boldsymbol{\theta}}_k$ wie der aktuelle Messvektor. Damit wird es sehr lange dauern, bis Änderungen der Modellparameter vom Algorithmus identifiziert werden, da die Information über geänderte Parameter nur in den Messvektoren nach der Parameteränderung enthalten ist – nicht aber in denjenigen Messvektoren, die schon davor auftraten. Erwünscht ist also ein Algorithmus, bei dem aktuelle Messvektoren stärker gewichtet werden als weit zurückliegende. Am gebräuchlichsten ist dabei die Methode des exponentiellen Vergessens. Hier wird die Matrix $\mathbf{P}_{k-1}^{-1}$ in Beziehung (D.3.10) mit einem positiven Faktor λ im Bereich von $0 \leq \lambda \leq 1$ gewichtet:

$$\mathbf{P}_k^{-1} = \lambda \mathbf{P}_{k-1}^{-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \tag{D.3.16}$$

Dadurch wird der aktuelle Messvektor in jedem Zeitschritt stärker gewichtet als die vorherigen. Auf die drei Gleichungen aus (D.3.15) wirkt sich das in weiterer Folge wie folgt aus:

$$\begin{aligned}
\mathbf{k}_k &= \frac{\mathbf{P}_{k-1}\boldsymbol{\phi}_k}{\lambda + \boldsymbol{\phi}_k^T \mathbf{P}_{k-1} \boldsymbol{\phi}_k} \\
\hat{\boldsymbol{\theta}}_k &= \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k \Big(y_k - \boldsymbol{\phi}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big) \\
\mathbf{P}_k &= \frac{1}{\lambda} \Big(\mathbf{P}_{k-1} - \mathbf{k}_k \boldsymbol{\phi}_k^T \mathbf{P}_{k-1} \Big)
\end{aligned} \tag{D.3.17}$$

Der RLS-Algorithmus nach (D.3.17) ermöglicht es also, auch zeitlich veränderlichen Modellparametern zu folgen (*tracking*). Diese Eigenschaft ermöglicht – wie sich zeigen wird – in weiterer Folge auch die Schätzung zeitvariabler Totzeiten.

3.3 Algorithmen zur Totzeitschätzung

Basierend auf dem RLS-Algorithmus nach (D.3.17) existieren mehrere Möglichkeiten, aus den geschätzten Modellparametern in $\hat{\boldsymbol{\theta}}_k$ auf eine Totzeit $\hat{d}_k$ zu schließen. Diese werden anschließend erläutert.

3.3.1 Methode nach Normey-Rico

Normey-Rico [4] schlägt eine Überparametrierung des Zählers der Übertragungsfunktion vor, um auf die Totzeit schließen zu können. Der Ansatz beruht auf der Überlegung, dass alle Zählerkoeffizienten, welche zu Eingangswerten gehören, die aktueller als die wahre Totzeit sind, gleich null sind. Daher schätzt man im Zähler um die Differenz zwischen maximaler und minimaler Totzeit (im Vorhinein festgelegt), also $d_{\max} - d_{\min}$ Koeffizienten mehr als durch die Modellordnung angenommen. Der für den RLS-Algorithmus (D.3.17) verwendete Messvektor $\boldsymbol{\phi}_k$ und der zugehörige Schätzvektor $\hat{\boldsymbol{\theta}}_k$ ergeben sich zu:

$$\boldsymbol{\phi}_k = \begin{pmatrix} -y_{k-1} \\ \vdots \\ -y_{k-n} \\ u_{k+m-n-d_{\min}} \\ \vdots \\ u_{k-n-d_{\min}} \\ u_{k-n-\left(d_{\min}+1\right)} \\ \vdots \\ u_{k-n-d_{\max}} \end{pmatrix} \quad ; \quad \hat{\boldsymbol{\theta}}_k = \begin{pmatrix} \hat{a}_{n-1,k} \\ \vdots \\ \hat{a}_{0,k} \\ \hat{b}_{m-d_{\min},k} \\ \vdots \\ \hat{b}_{-d_{\min},k} \\ \hat{b}_{-\left(d_{\min}+1\right),k} \\ \vdots \\ \hat{b}_{-d_{\max},k} \end{pmatrix} \tag{D.3.18}$$

Die geschätzte Übertragungsfunktion $\hat{G}(z)$ hat dann folgende Form:

$$\hat{G}\left(z\right) = \frac{\hat{b}_{m-d_{\min}} z^{m-d_{\min}} + \ldots + \hat{b}_{-d_{\min}} z^{-d_{\min}} + \hat{b}_{-\left(d_{\min}+1\right)} z^{-\left(d_{\min}+1\right)} + \ldots + \hat{b}_{-d_{\max}} z^{-d_{\max}}}{z^n + \hat{a}_{n-1} z^{n-1} + \ldots + \hat{a}_0} \tag{D.3.19}$$

Als Vergleich zieht man eine Übertragungsfunktion mit konstanter Totzeit d heran:

$$G\left(z\right) = \frac{b_m z^m + b_{m-1} z^{m-1} \ldots + b_0}{z^n + a_{n-1} z^{n-1} + \ldots + a_0} z^{-d} = \frac{b_m z^{m-d} + b_{m-1} z^{m-1-d} \ldots + b_0 z^{-d}}{z^n + a_{n-1} z^{n-1} + \ldots + a_0} \tag{D.3.20}$$

Nimmt man nun an, dass das System (D.3.20) durch die Überparametrierung (D.3.19) zum Zeitpunkt k korrekt geschätzt wird, so wird klar, dass die ersten $d - d_{\min}$ geschätzten Zählerkoeffizienten in (D.3.19) den Wert Null annehmen:

$$\hat{b}_{m-d_{\min},k} = \hat{b}_{m-1-d_{\min},k} = \ldots = \hat{b}_{m-\left(d-1\right),k} = 0 \tag{D.3.21}$$

Zur Schätzung der Totzeit $\hat{d}_k$ zum Zeitschritt k müsste man also theoretisch nur feststellen, welcher der erste Koeffizient ist, der von null abweicht:

$$\hat{d}_k = j \quad \Leftrightarrow \quad \left(\hat{b}_{m-i,k} = 0 \ \forall \ i = d_{\min}, d_{\min}+1 \ldots, j-1 \right) \ \wedge \ \left(\hat{b}_{m-j,k} \neq 0 \right) \quad \text{(D.3.22)}$$

In der Praxis ist es jedoch so, dass die ersten d Zählerkoeffizienten des Zählerpolynoms aufgrund von Messrauschen niemals exakt den Wert Null annehmen werden. In [18] und [19] wird vorgeschlagen die Zählerkoeffizienten mit einer im Vorhinein festgelegten Schwelle zu vergleichen. Dies birgt jedoch die Schwierigkeit, dass für jede Anwendung zunächst eine passende Schwelle gefunden werden müsste. Als Abhilfe für dieses Problem wird in [4] vorgeschlagen, die Zählerkoeffizienten miteinander zu vergleichen und die Totzeit anhand der Verhältnisse der Koeffizienten zu ermitteln:

$$\hat{d}_k = j \quad \Leftrightarrow \quad \frac{\hat{b}_{m-i,k}}{b_{m-j,k}} < C_1 \ \forall \ i = d_{\min}, d_{\min}+1, \ldots, j-1 \quad \text{(D.3.23)}$$

Für die Konstante C_1 wird der Wert $C_1 = 0.15$ (empirisch ermittelt) vorgeschlagen. Es muss also keine explizite Schwelle für die Koeffizienten mehr angegeben werden, da die Totzeit einzig und allein aus dem Verhältnis der geschätzten Zählerkoeffizienten geschätzt wird. Dies kann lediglich dann zu Problemen führen, wenn die voranstehenden der Zählerkoeffizienten eines zeitdiskreten Systems tatsächlich kleiner als ein nachfolgender Koeffizient multipliziert mit der Schwelle $C_1 = 0.15$ sind. In diesem Fall würde die Totzeit zu hoch geschätzt werden.

3.3.2 Methode nach De Keyser

Einen Ansatz, der ebenfalls auf der Überparametrierung des Zählerpolynoms beruht, verfolgt De Keyser [20]. Um den Rechenaufwand zu begrenzen (Anzahl der geschätzten Parameter geht quadratisch ein), wird jedoch nur ein einziger zusätzlicher Zählerkoeffizient mitgeschätzt:

$$\hat{G}\left(z \right) = \frac{\hat{b}_m z^{m-d} + \hat{b}_{m-1} z^{m-1-d} + \ldots + \hat{b}_0 z^{-d} + \hat{b}_{-1} z^{-1-d}}{z^n + \hat{a}_{n-1} z^{n-1} + \ldots + \hat{a}_0} \quad \text{(D.3.24)}$$

Das Problem, dass hier zur Schätzung der Modellparameter die Totzeit d bekannt sein müsste, wird dadurch gelöst, dass anstatt von d die zuletzt geschätzte Totzeit $\hat{d}_{k-1}$ verwendet wird. Der Messvektor $\boldsymbol{\phi}_k$ im RLS-Algorithmus nach (D.3.17) muss also durch einen geschätzten Messvektor $\hat{\boldsymbol{\phi}}_k$ ersetzt werden, bei dem zurückliegende Eingangswerte entsprechend $\hat{d}_{k-1}$ verwendet werden. Dieser und der Schätzvektor $\hat{\boldsymbol{\theta}}_k$ haben also folgende Struktur:

$$\hat{\boldsymbol{\Phi}}_k = \hat{\boldsymbol{\Phi}}_k\Big(\,\hat{d}_{k-1}\,\Big) = \begin{pmatrix} -y_{k-1} \\ \vdots \\ -y_{k-n} \\ u_{k+m-n-\hat{d}_{k-1}} \\ \vdots \\ u_{k-n-\hat{d}_{k-1}} \\ u_{k-n-1-\hat{d}_{k-1}} \end{pmatrix} \quad ; \quad \hat{\boldsymbol{\theta}}_k = \begin{pmatrix} \hat{a}_{n-1,k} \\ \vdots \\ \hat{a}_{0,k} \\ \hat{b}_{m,k} \\ \vdots \\ \hat{b}_{0,k} \\ \hat{b}_{-1,k} \end{pmatrix} \tag{D.3.25}$$

Die Idee besteht nun darin, die Struktur nach (D.3.24) mit zwei anderen, sogenannten Kandidaten-Strukturen zu vergleichen. Diese lauten:

$$G_+\Big(\,z\,\Big) = \frac{\hat{b}_{m-1}z^{m-1-d} + \ldots + \hat{b}_0 z^{-d} + \hat{b}_{-1}z^{-1-d}}{z^n + \hat{a}_{n-1}z^{n-1} + \ldots + \hat{a}_0} \tag{D.3.26}$$

$$G_-\Big(\,z\,\Big) = \frac{\hat{b}_m z^{m-d} + \hat{b}_{m-1}z^{m-1-d} + \ldots + \hat{b}_0 z^{-d}}{z^n + \hat{a}_{n-1}z^{n-1} + \ldots + \hat{a}_0} \tag{D.3.27}$$

Es wird nun zu jedem Zeitschritt überprüft, welche der beiden obigen Kandidaten-Strukturen (D.3.26) und (D.3.27) der geschätzten Struktur (D.3.24) besser entspricht. Kommt die Struktur $\hat{G}_+(z)$ der geschätzten Struktur $\hat{G}(z)$ näher, dann weist dies auf eine gestiegene Totzeit $\hat{d}_k = \hat{d}_{k-1} + 1$ hin. Entspricht aber eher die Kandidaten-Struktur $G_-(z)$ der geschätzten Struktur $\hat{G}(z)$, dann ist dies ein Hinweis auf eine gleichbleibende bzw. sinkende Totzeit ($\hat{d}_k = \hat{d}_{k-1}$ oder $\hat{d}_k = \hat{d}_{k-1} - 1$). In [20] wird gezeigt, dass sich diese Überlegung in der folgenden, einfachen Vorschrift ausdrücken lässt:

$$\hat{d}_k = \begin{cases} \hat{d}_{k-1} - 1 & \text{für} \quad \Big|\,\hat{b}_{m,k}\,\Big| > C_2 \Big|\,\hat{b}_{-1,k}\,\Big| \\ \hat{d}_{k-1} & \text{sonst} \\ \hat{d}_{k-1} + 1 & \text{für} \quad \Big|\,\hat{b}_{-1,k}\,\Big| > C_2 \Big|\,\hat{b}_{m,k}\,\Big| \end{cases} \tag{D.3.28}$$

Für die Konstante C_2 wird ein empirisch ermittelter Wert $C_2 = 5$ vorgeschlagen. Mit dieser Methode ist es also möglich, anhand des Vergleichs von nur zwei geschätzten Zählerkoeffizienten eine Aussage darüber zu treffen, ob die aktuell geschätzte Totzeit $\hat{d}_k$ im Vergleich zur zuletzt geschätzten Totzeit $\hat{d}_{k-1}$ erhöht oder verringert werden muss. Daraus wird deutlich, dass diese Methode insbesondere dann sinnvoll ist, wenn sich eine auftretende Totzeit nur einstufig ändert, also entweder um einen Zeitschritt steigt oder sinkt. Außerdem ist hier ein sinnvoller Initialisierungswert $\hat{d}_0$ immens wichtig.

3.3.3 Methode nach Elnaggar

Die Methode nach Elnaggar et al. wird in [21] und [22] beschrieben. Hier wird die Totzeitschätzung nicht wie in den beiden vorherigen Methoden aus einer Überparametrierung des Zählerpolynoms der geschätzten Übertragungsfunktion abgeleitet. Stattdessen erfolgt in einem erstem Schritt zu jedem Zeitpunkt k eine Schätzung der Modellparameter folgender Übertragungsfunktion mit Zählergrad m und Nennergrad n:

$$\hat{G}\left(z\right) = \frac{\hat{b}_m z^{m-d} + \hat{b}_{m-1} z^{m-1-d} + \ldots + \hat{b}_0 z^{-d}}{z^n + \hat{a}_{n-1} z^{n-1} + \ldots + \hat{a}_0} \tag{D.3.29}$$

Der geschätzte Messvektor $\hat{\boldsymbol{\phi}}_k$ (auch hier muss die wahre Totzeit d durch $\hat{d}_{k-1}$ ersetzt werden) und der zugehörige Schätzvektor $\hat{\boldsymbol{\theta}}_k$ sehen hier also folgendermaßen aus:

$$\hat{\boldsymbol{\phi}}_k = \hat{\boldsymbol{\phi}}_k\left(\hat{d}_{k-1}\right) = \begin{pmatrix} -y_{k-1} \\ \vdots \\ -y_{k-n} \\ u_{k+m-n-\hat{d}_{k-1}} \\ \vdots \\ u_{k-n-\hat{d}_{k-1}} \end{pmatrix} \quad ; \quad \hat{\boldsymbol{\theta}}_k = \begin{pmatrix} \hat{a}_{n-1,k} \\ \vdots \\ \hat{a}_{0,k} \\ \hat{b}_{m,k} \\ \vdots \\ \hat{b}_{0,k} \end{pmatrix} \tag{D.3.30}$$

Es ist zu erkennen, dass der Messvektor $\hat{\boldsymbol{\phi}}_k$ wie schon bei der Methode nach De Keyser (Kapitel 3.3.2) von der zuletzt geschätzten Totzeit $\hat{d}_{k-1}$ abhängt. Um in einem zweiten Schritt auf die aktuell geschätzte Totzeit $\hat{d}_k$ zu kommen, wird eine Gütefunktion $J_k(d)$ für alle möglichen Totzeiten $d \in [d_{\min}, d_{\max}]$ ausgewertet:

$$J_k\left(d\right) = \lambda J_{k-1}\left(d\right) + \left[\, y_k - \boldsymbol{\phi}_k^T\left(d\right)\hat{\boldsymbol{\theta}}_k \,\right]^2 \quad \forall \quad d \in \left[\, d_{\min}, d_{\max} \,\right] \tag{D.3.31}$$

Der Wert dieser Gütefunktion zum Zeitpunkt k berechnet sich also aus einer Summe. Den ersten Summanden bildet dabei das Produkt aus dem Wert der entsprechenden Gütefunktion zum Zeitpunkt $k-1$ und dem Vergessensfaktor λ - dieser entspricht dem Vergessensfaktor des RLS-Algorithmus nach (D.3.17). Es handelt sich also um eine Gütefunktion, die ähnlich wie der RLS-Algorithmus auf exponentiellem Vergessen basiert. Der zweite Summand entspricht dem quadrierten Modellfehler unter der Annahme, dass eine Totzeit d vorliegt. Zur Berechnung dieses Modellfehlers wird der im ersten Schritt mit Hilfe des geschätzten Messvektors $\hat{\boldsymbol{\phi}}_k(\hat{d}_{k-1})$ berechnete Schätzvektor $\hat{\boldsymbol{\theta}}_k$ und der folgende angepasste Messvektor $\boldsymbol{\phi}_k(d)$ verwendet:

$$\Phi_k\left(\, d \,\right) = \begin{pmatrix} -y_{k-1} \\ \vdots \\ -y_{k-n} \\ u_{k+m-n-d} \\ \vdots \\ u_{k-n-d} \end{pmatrix} \tag{D.3.32}$$

Nach Auswertung der eben erläuterten Gütefunktion (D.3.31) für alle möglichen Totzeiten $d \in [d_{\min}, d_{\max}]$ ergibt sich anschließend die neue, geschätzte Totzeit $\hat{d}_k$ aus derjenigen Totzeit, für welche die zugehörige Gütefunktion den minimalen Wert annimmt:

$$\hat{d}_k = \arg\min\left\{ J_k\left(\, d \,\right)\right\} \quad \forall \quad d \in \left[\, d_{\min}, d_{\max} \,\right] \tag{D.3.33}$$

Aufgrund ihres Ablaufes in zwei Schritten (zunächst RLS-Schätzung, dann Auswertung einer Gütefunktion für alle möglichen Totzeiten) wird diese Methode zur Totzeit-Schätzung auch Zweischritt-Methode [23] genannt. Es wird also in jedem Zeitschritt k eine RLS-Identifikation des Systems mit der angenommenen Modellordnung durchgeführt und anschließend wird mit Hilfe der identifizierten Parameter und einer Gütefunktion auf die Totzeit geschlossen. Die Besonderheit besteht darin, dass der Messvektor für den RLS-Algorithmus immer entsprechend der zuletzt geschätzten Totzeit gebildet wird.

Bei ersten Versuchsbeispielen unter Verwendung dieser Methode wurde relativ früh festgestellt, dass zur Verbesserung der Anwendbarkeit noch eine kleine Veränderung des Algorithmus notwendig ist. Es stellte sich heraus, dass die Tatsache, dass der Vergessensfaktor für den rekursiven Least-Squares-Algorithmus (D.3.17) und für die Gütefunktionen der möglichen Totzeiten (D.3.31) identisch ist, nicht unbedingt vorteilhaft ist. Beispielsweise wäre bei gleich bleibenden Modellparametern und zeitlich variierender Totzeit ein Vergessensfaktor $\lambda \approx 1$ für den RLS-Algorithmus und ein Vergessensfaktor $\lambda < 1$ für die Totzeit-Gütefunktionen wünschenswert. Andererseits sollte bei zeitvariablen Modellparametern der RLS-Vergessensfaktor $\lambda < 1$ sein, während für die Gütefunktionen der Totzeit $\lambda \approx 1$ sinnvoll wäre. Diese Konfigurationen sind bei der Elnaggar-Methode in ihrer Grundform jedoch nicht möglich. Eine naheliegende Anpassung des Algorithmus liegt darin, dass man für den RLS-Algorithmus und die Gütefunktionen der Totzeit unterschiedliche Vergessensfaktoren verwendet. Das heißt, der Vergessensfaktor λ wird entkoppelt und man verwendet einen separaten Vergessensfaktor λ_1 für den RLS-Algorithmus und einen zweiten separaten Vergessensfaktor λ_2 für die Gütefunktionen der möglichen Totzeiten. Im weiteren Verlauf dieser Arbeit wird unter dem Algorithmus nach Elnaggar also der folgende, leicht angepasste Ablauf verstanden:

$$\mathbf{k}_k = \frac{\mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}{\lambda_1 + \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k \Big(y_k - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big)$$

$$\mathbf{P}_k = \frac{1}{\lambda_1} \Big(\mathbf{P}_{k-1} - \mathbf{k}_k \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \Big) \tag{D.3.34}$$

$$J_k\Big(\, d \,\Big) = \lambda_2 J_{k-1}\Big(\, d \,\Big) + \Big[\, y_k - \boldsymbol{\phi}_k^T\Big(\, d \,\Big)\hat{\boldsymbol{\theta}}_k \,\Big]^2 \qquad \forall \qquad d \in \Big[\, d_{\min}, d_{\max} \,\Big]$$

$$\hat{d}_k = \arg\min \Big\{ J_k\Big(\, d \,\Big) \Big\} \qquad \forall \qquad d \in \Big[\, d_{\min}, d_{\max} \,\Big]$$

3.3.4 Methode nach Bedoui

Die letzte rekursive Least-Squares-Methode zur Totzeitschätzung, welche hier vorgestellt wird, ist die Methode nach Bedoui et al., welche in [24], [25] und [26] zu finden ist. Analog zur von den gleichen Autoren vorgeschlagenen gradientenbasierten Methode (Kapitel 2) ist hier die Totzeit Teil eines generalisierten Schätzvektors $\hat{\boldsymbol{\theta}}_{G,k}$:

$$\hat{\boldsymbol{\theta}}_{G,k} = \begin{pmatrix} \hat{a}_{n-1,k} \\ \vdots \\ \hat{a}_{0,k} \\ \hat{b}_{m,k} \\ \vdots \\ \hat{b}_{0,k} \\ \hat{d}_k \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\theta}}_k \\ \hat{d}_k \end{pmatrix} \tag{D.3.35}$$

Um einen für diesen Schätzvektor passenden generalisierten Messvektor $\hat{\boldsymbol{\phi}}_{G,k}$ zu erhalten, ermittelt man die negative Ableitung des Modellfehlers e_k nach dem generalisierten Schätzvektor $\hat{\boldsymbol{\theta}}_{G,k-1}$. Der Modellfehler sieht dabei wie folgt aus:

$$e_k = y_k - \begin{pmatrix} \boldsymbol{\phi}_k^T & 0 \end{pmatrix} \hat{\boldsymbol{\theta}}_{G,k-1} = y_k - \boldsymbol{\phi}_k^T \hat{\boldsymbol{\theta}}_{k-1} \tag{D.3.36}$$

Die weitere Berechnung des generalisierten Messvektors $\hat{\boldsymbol{\phi}}_{G,k}$ samt Approximation der Ableitung des Modellfehlers e_k nach der zuletzt geschätzten Totzeit $\hat{d}_{k-1}$ verläuft anschließend ähnlich zu (D.2.8) und (D.2.9):

$$
\begin{aligned}
\hat{\boldsymbol{\phi}}_{G,k} &= -\frac{\partial e_k}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \\
&= -\frac{\partial}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \Big(y_k - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big) \\
&= \frac{\partial}{\partial \hat{\boldsymbol{\theta}}_{G,k-1}} \left(\sum_{i=0}^{m} \hat{b}_{i,k-1} u_{k+i-n-\hat{d}_{k-1}} - \sum_{i=0}^{n-1} \hat{a}_{i,k-1} y_{k+i-n} \right) \\
&= \begin{pmatrix} \hat{\boldsymbol{\phi}}_k \\ \dfrac{\partial}{\partial \hat{d}_{k-1}} \left(\displaystyle\sum_{i=0}^{m} \hat{b}_{i,k-1} u_{k+i-n-\hat{d}_{k-1}} \right) \end{pmatrix} \\
&\approx \begin{pmatrix} \hat{\boldsymbol{\phi}}_k \\ \displaystyle\sum_{i=0}^{m} \hat{b}_{i,k-1} \left(\dfrac{u_{k+i-n-\hat{d}_{k-1}-1} - u_{k+i-n-\hat{d}_{k-1}}}{\Big(\hat{d}_{k-1} + 1 \Big) - \hat{d}_{k-1}} \right) \end{pmatrix} \\
&= \begin{pmatrix} \hat{\boldsymbol{\phi}}_k \\ \displaystyle\sum_{i=0}^{m} \hat{b}_{i,k-1} \Big(u_{k+i-n-\hat{d}_{k-1}-1} - u_{k+i-n-\hat{d}_{k-1}} \Big) \end{pmatrix}
\end{aligned}
\tag{D.3.37}
$$

Unter Verwendung des obigen generalisierten Messvektors $\hat{\boldsymbol{\phi}}_{G,k}$ ergibt sich der folgende modifizierte RLS-Algorithmus zur sukzessiven Ermittlung des generalisierten Schätzvektors $\hat{\boldsymbol{\theta}}_{G,k}$ gemäß (D.3.36):

$$
\begin{aligned}
\mathbf{k}_k &= \frac{\mathbf{P}_{k-1} \hat{\boldsymbol{\phi}}_{G,k}}{\lambda + \hat{\boldsymbol{\phi}}_{G,k}^T \mathbf{P}_{k-1} \hat{\boldsymbol{\phi}}_{G,k}} \\
\hat{\boldsymbol{\theta}}_{G,k} &= \hat{\boldsymbol{\theta}}_{G,k-1} + \mathbf{k}_k \Big(y_k - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big) \\
\mathbf{P}_k &= \frac{1}{\lambda} \Big(\mathbf{P}_{k-1} - \mathbf{k}_k \hat{\boldsymbol{\phi}}_{G,k}^T \mathbf{P}_{k-1} \Big)
\end{aligned}
\tag{D.3.38}
$$

Mit der Methode nach Bedoui ist es also möglich, die Totzeit als Teil des Schätzvektors anhand eines modifizierten RLS-Algorithmus unmittelbar mitzuschätzen. Dies bringt gegenüber den anderen auf dem RLS-Algorithmus basierenden Schätzmethoden eine Verbesserung in der Recheneffizienz mit sich.

3.4 Auswahl der geeignetsten Methode

Zur Auswahl der am besten geeigneten Methode wurde ein lineares zeitinvariantes System mit zeitvariabler Totzeit d_k verwendet:

$$G\Big(\, z \,\Big) = \frac{0.5}{z - 0.8}\, z^{-d_k} \tag{D.3.39}$$

Offenkundig besitzt dieses System einen Zählergrad von $m = 0$ und einen Nennergrad von $n = 1$. Für das Testbeispiel wird nun ein Zeitrahmen $0 \leq k \leq 999$ betrachtet. Die zeitvariable Totzeit d_k während dieser 1000 Samples verhält sich wie folgt:

$$d_k = \begin{cases} 0 \;\text{für}\; 0 \leq \mathrm{k} \leq 99 \\ 1 \;\text{für}\; 100 \leq \mathrm{k} \leq 199 \\ 2 \;\text{für}\; 200 \leq \mathrm{k} \leq 299 \\ 3 \;\text{für}\; 300 \leq \mathrm{k} \leq 399 \\ 4 \;\text{für}\; 400 \leq \mathrm{k} \leq 499 \\ 5 \;\text{für}\; 500 \leq \mathrm{k} \leq 599 \\ 4 \;\text{für}\; 600 \leq \mathrm{k} \leq 699 \\ 3 \;\text{für}\; 700 \leq \mathrm{k} \leq 799 \\ 2 \;\text{für}\; 800 \leq \mathrm{k} \leq 899 \\ 1 \;\text{für}\; 900 \leq \mathrm{k} \leq 999 \end{cases} \tag{D.3.40}$$

Die Totzeit d_k steigt also stufenförmig vom Wert 0 bis zum Wert 5 an und sinkt anschließend wieder bis auf den Wert 1 ab. Abbildung 14 veranschaulicht diesen Verlauf:

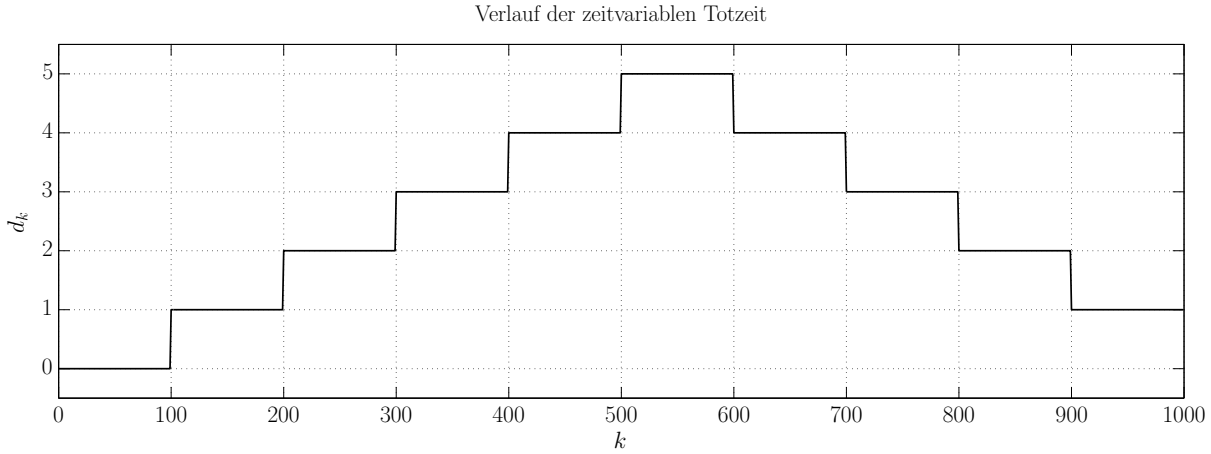

Abbildung 14: Auswahl geeignetste Methode - wahre Totzeit

Das System (D.3.39) wird jetzt mit verschiedenartigen Eingangssignalen u_k angeregt. Angemerkt sei, dass für u_k und y_k wieder Kausalität vorausgesetzt wurde (es gilt also $u_k = 0 \; \forall \; k < 0$ und $y_k = 0 \; \forall \; k < 0$). Folgende Eingangssignale mit jeweils 1000 Samples Länge wurden verwendet:

- Random-Binärfolge (Werte -1 und 1)
- Normalverteiltes Rauschen mit Standardabweichung $\sigma = 1$
- Sinusschwingung mit Periodendauer 100
- Sägezahn mit Periodendauer 100
- Rechteck mit Periodendauer 100

Die vier vorgestellten rekursiven Least-Squares-Methoden zur Totzeitschätzung wurden mit diesen fünf Eingangssignalen und den dazugehörigen Ausgangssignalen getestet. Um einen fairen Vergleich der Methoden zu gewährleisten, wurde eine Art Raster-Optimierung der jeweils frei wählbaren Algorithmus-Parameter durchgeführt: Dies wurde durch eine einfache gleichverteilte Abtastung sinnvoller Parameterbereiche erreicht. Derjenige Parameter bzw. diejenige Parameterkonfiguration, für welche die Totzeitschätzung des jeweiligen Algorithmus das beste Ergebnis lieferte, wurde für den entsprechenden Testfall ausgewählt. Tabelle 1 zeigt die auf diese Art und Wiese optimierten Algorithmus-Parameter, die abgetasteten Bereiche und die Anzahl der abgetasteten Punkte:

Algorithmus	Optimierte Parameter	Abgetasteter Bereich	Anzahl Punkte
Normey-Rico	λ (Vergessensfaktor RLS)	[0.9 , 1]	21
	C_1 (Schwelle Koeffizientenvergleich)	[0 , 0.5]	21
De Keyser	λ (Vergessensfaktor RLS)	[0.9 , 1]	21
	C_2 (Schwelle Koeffizientenvergleich)	[2 , 12]	21
Elnaggar	λ_1 (Vergessensfaktor RLS)	[0.9 , 1]	21
	λ_2 (Vergessensfaktor Gütefunktionen)	[0.9 , 1]	21
Bedoui	λ (Vergessensfaktor RLS)	[0.9 , 1]	21

Tabelle 1: Optimierte Algorithmus-Parameter

Eine Gemeinsamkeit im Vergleich der Methoden stellen die Initialisierungswerte für den RLS-Algorithmus dar. Diese lauten:

$$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad ; \quad \hat{\boldsymbol{\theta}}_0 \;/\; \hat{\boldsymbol{\theta}}_{G,0} = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \quad ; \quad \left(\hat{d}_0 = 0 \right) \qquad \text{(D.3.41)}$$

Die Dimensionen von $\mathbf{P}_0$ und $\hat{\boldsymbol{\theta}}_0$ ergeben sich entsprechend den Algorithmen. Außerdem wurden die minimale und die maximale Totzeit mit $d_{\min} = 0$ und $d_{\max} = 10$ gewählt. Abbildung 15 zeigt Eingangs- und Ausgangssignal bei Anregung des Systems mit einer Random-Binärfolge:

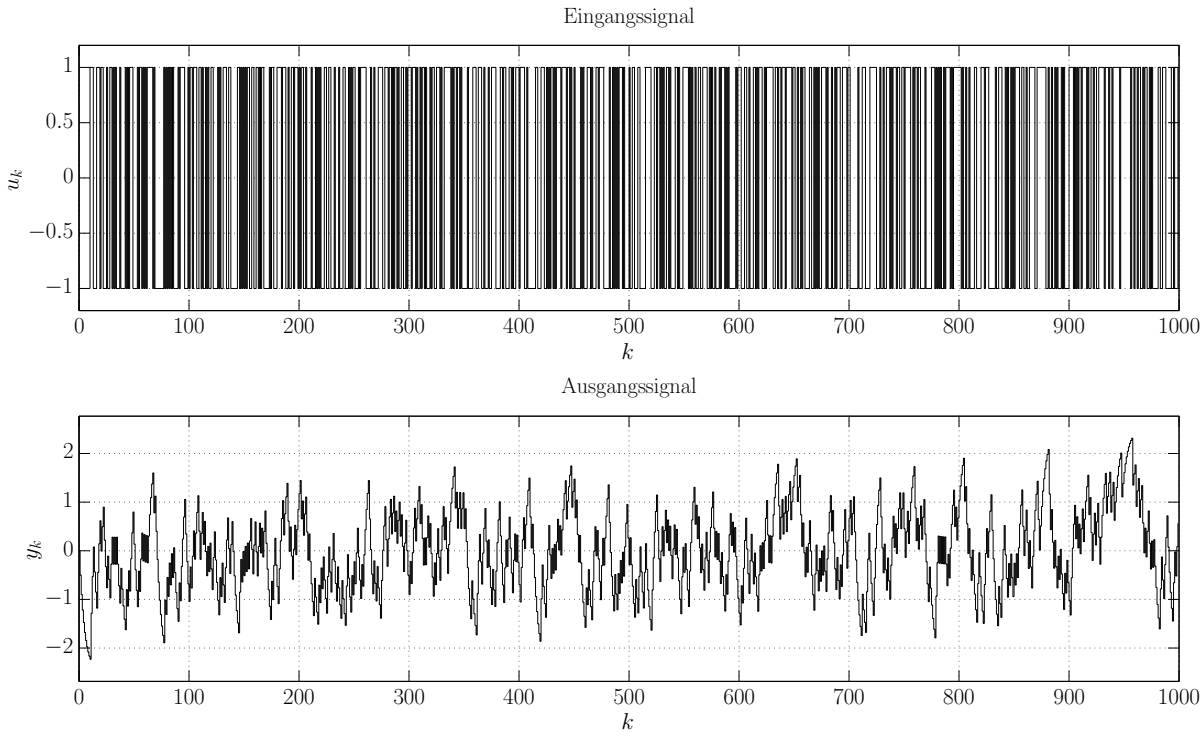

Abbildung 15: Auswahl geeignetste Methode - Ein-/Ausgang Random-Binärfolge

Die beiden obigen Signale u_k und y_k dienen nun als Eingangsgrößen für die beschriebenen Methoden zur Totzeitschätzung basierend auf der rekursiven Least-Squares-Identifikation. Abbildung 16 zeigt die Ergebnisse, wobei die sich durch die beschriebene Optimierung ergebenden optimalen Algorithmus-Parameter jeweils in der Legende angeführt sind. Dies ist auch in den weiteren Testbeispielen der Fall.

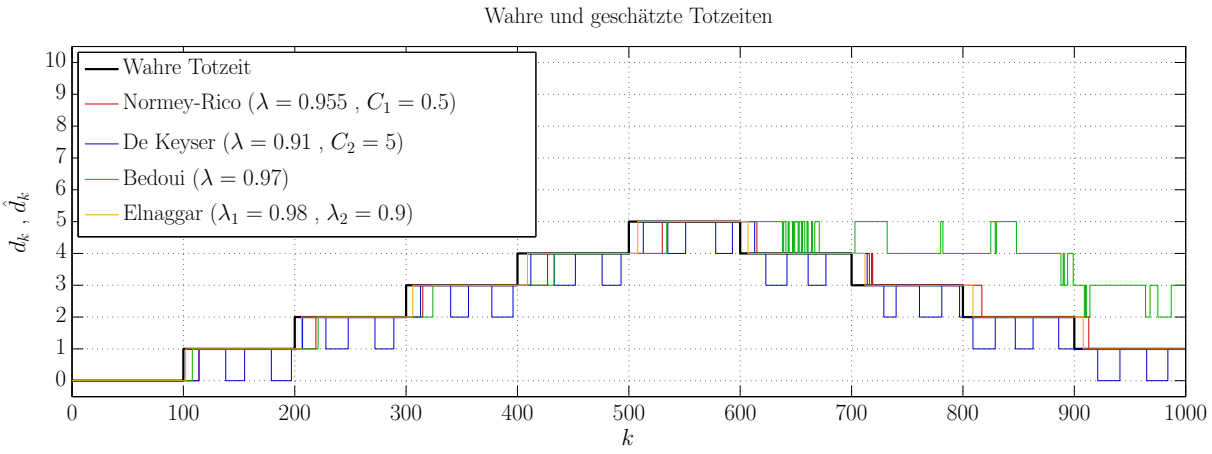

Abbildung 16: Auswahl geeignetste Methode - Totzeit Random-Binärfolge

Man erkennt, dass der Verlauf der Totzeit im Grunde von allen vier Methoden zufriedenstellend geschätzt wird. Ein leicht schlechteres Verhalten zeigt die Methode nach De Keyser, hier schwankt die geschätzte Totzeit zwischen dem wahren Wert und dem darunterliegenden Wert. Beim Bedoui-Algorithmus fällt auf, dass dieser für eine steigende Totzeit sehr gut funktioniert, jedoch ergeben sich für eine sinkende Totzeit Schwierigkeiten. Dies ist das selbe Phänomen, dass schon bei der gradientenbasierten Methode (Kapitel 2) festgestellt wurde. Auch hier konnte jedoch keine entscheidende Verbesserung durch eine andere Approximierung erzielt werden.

Abbildung 17 zeigt das Ein- und Ausgangssignal für normalverteiltes Rauschen:

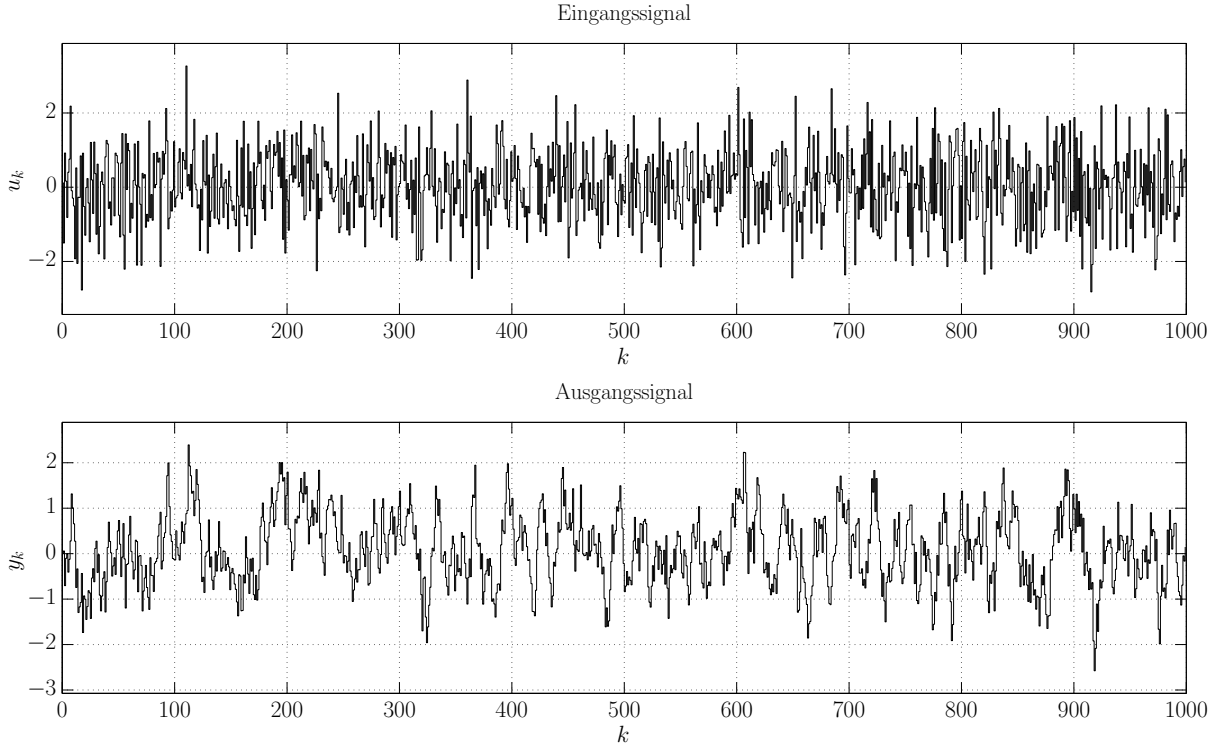

Abbildung 17: Auswahl geeignetste Methode - Ein-/Ausgang Rauschen

Nachfolgend sieht man die in diesem Test geschätzten Totzeitverläufe:

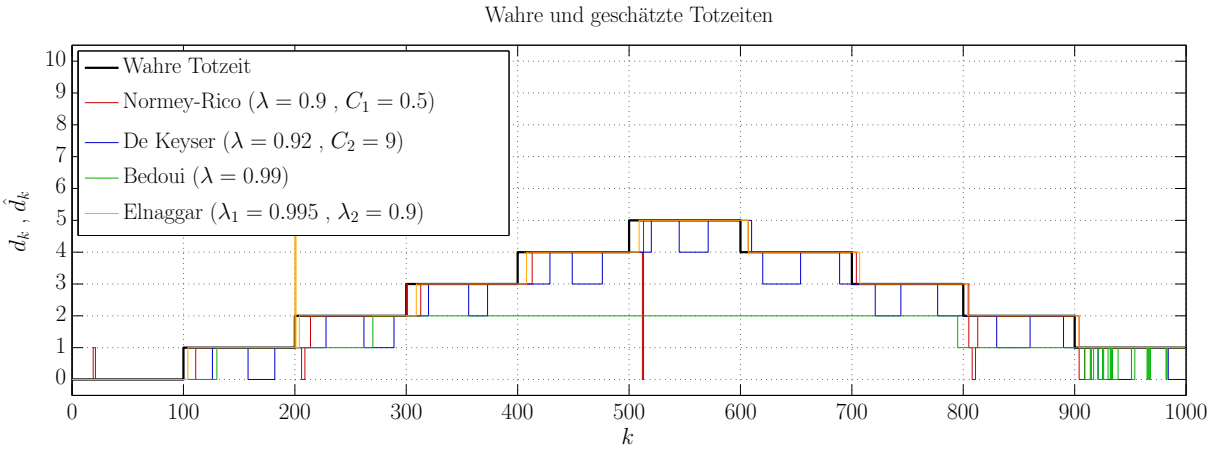

Abbildung 18: Auswahl geeignetste Methode - Totzeit Rauschen

Die Beobachtungen sind hier ähnlich wie bei der Random-Binärfolge als Eingangssignal, der Bedoui-Algorithmus ist jedoch nicht mehr wirklich brauchbar. Der Ausreißer der Methode nach Elnaggar zu Beginn ist vernachlässigbar, dieser kommt daher, dass diese Methode eine gewisse Einschwingzeit benötigt.

In Abbildung 19 sieht man Ein- und Ausgangssignal für den dritten Testfall - die Anregung mit einer vergleichsweise niederfrequenten Sinusschwingung:

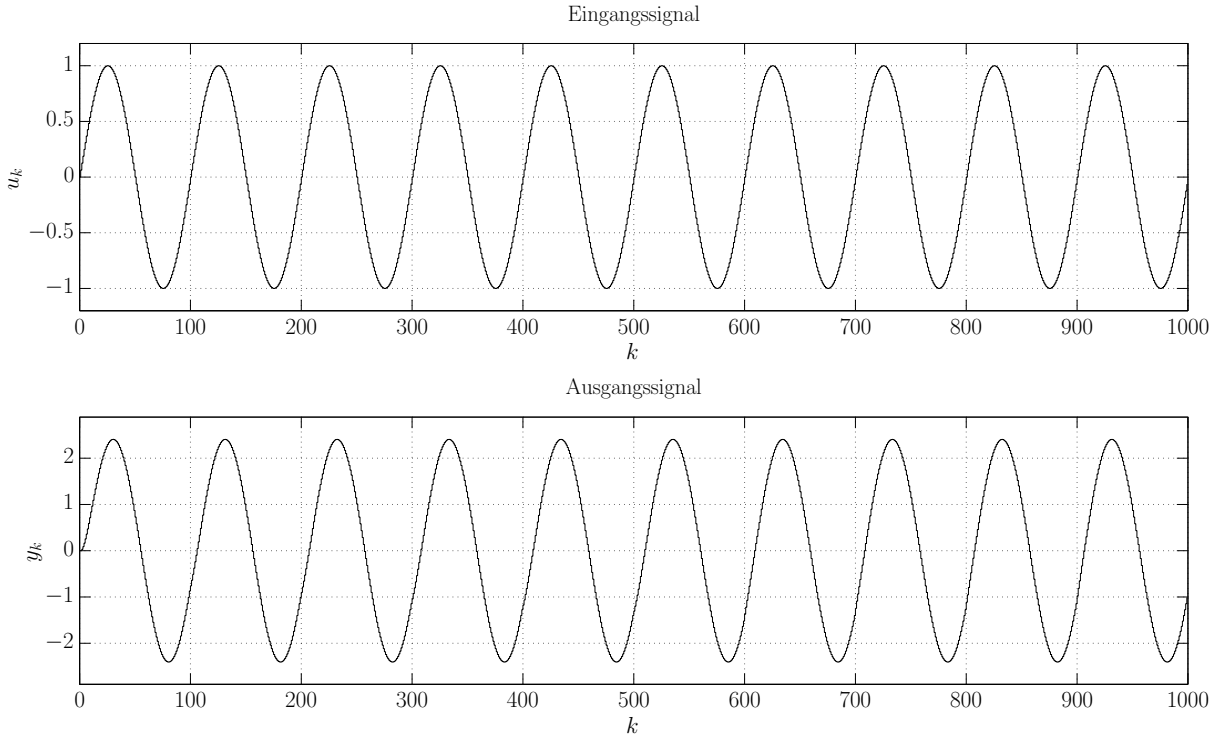

Abbildung 19: Auswahl geeignetste Methode - Ein-/Ausgang Sinus

Die zugehörigen geschätzten Totzeiten zeigt Abbildung 20:

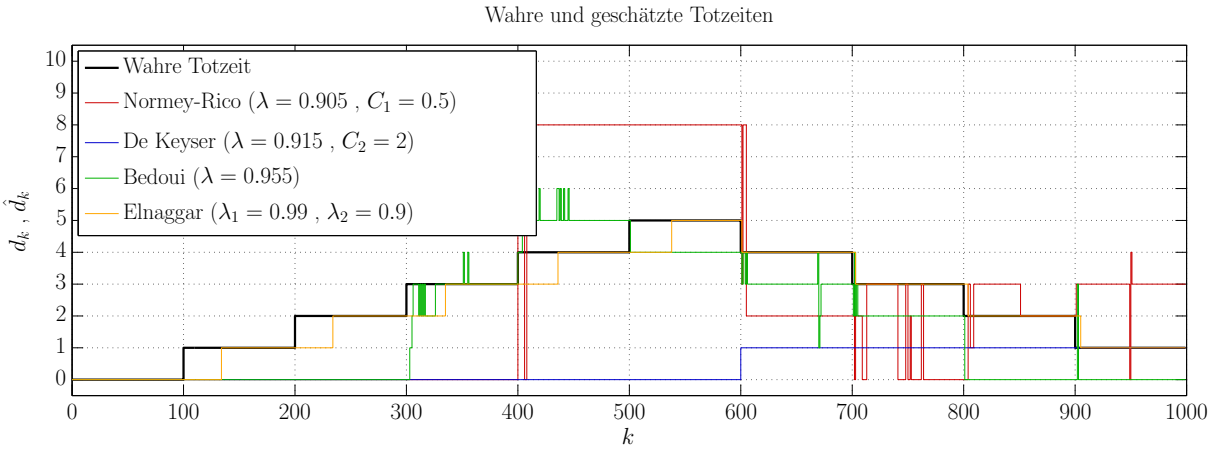

Abbildung 20: Auswahl geeignetste Methode - Totzeit Sinus

Für die Sinusanregung sind die Methode nach Normey-Rico und De Keyser offensichtlich unbrauchbar. Der Elnaggar-Algorithmus liefert dagegen ein gutes Ergebnis, auch der Bedoui-Algorithmus funktioniert zufriedenstellend. Dass sich hier insgesamt schlechtere Ergebnisse ergeben, war jedoch zu erwarten, da das System nur durch eine einzige Frequenz angeregt wird. Bekanntermaßen wäre für eine qualitativ hochwertige Systemidentifikation ein breitbandiges Anregungssignal nötig.

Abbildung 21 zeigt Ein-/Ausgang für die Anregung mit einem Sägezahnsignal:

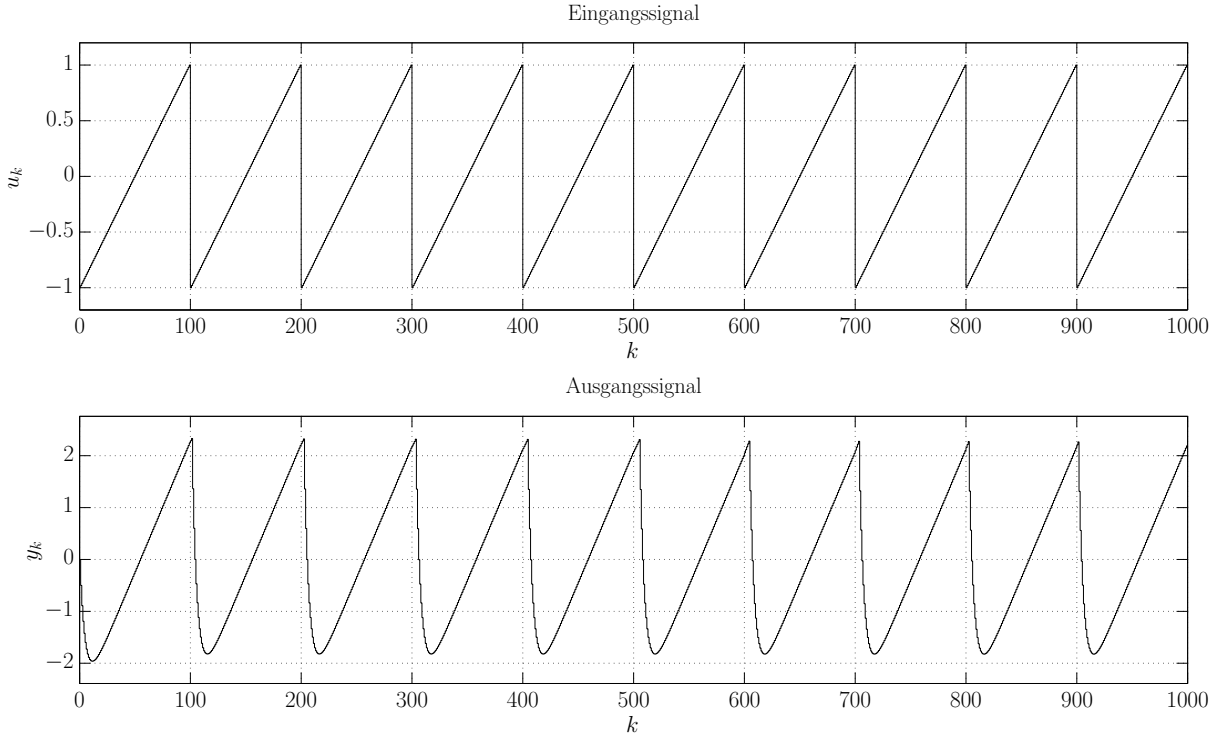

Abbildung 21: Auswahl geeignetste Methode - Ein-/Ausgang Sägezahn

In Abbildung 22 sieht man die Schätzergebnisse:

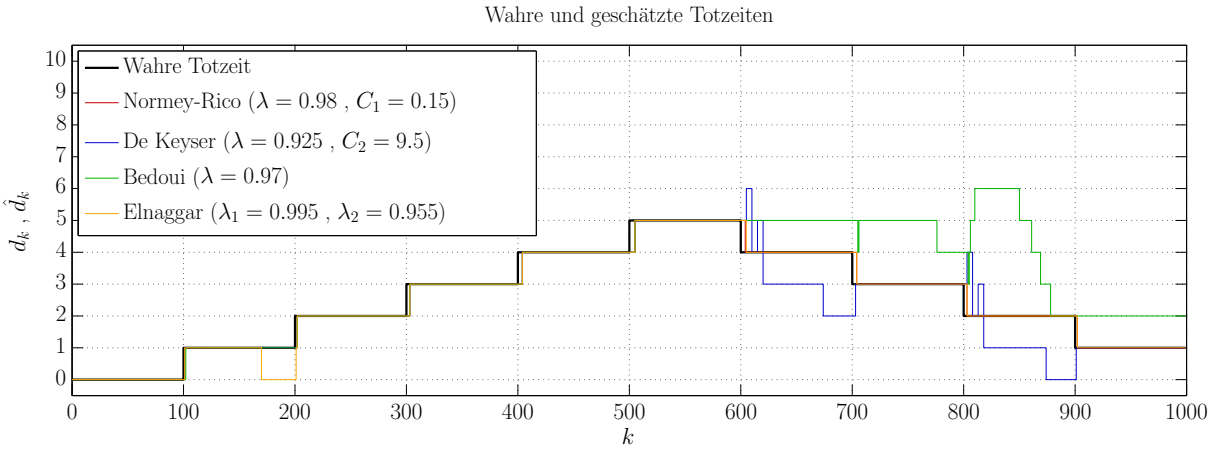

Abbildung 22: Auswahl geeignetste Methode - Totzeit Sägezahn

Man erhält für die Sägezahnanregung wieder hauptsächlich gute Ergebnisse, der Algorithmus nach Normey-Rico funktioniert dabei am besten. Auch der Elnaggar-Algorithmus zeigt aber bis auf den Ausreißer zu Beginn wieder ein gutes Ergebnis.

Der letzte Testfall ist eine Anregung mit einem Rechtecksignal – Abbildung 23 zeigt das Eingangs- und das Ausgangssignal:

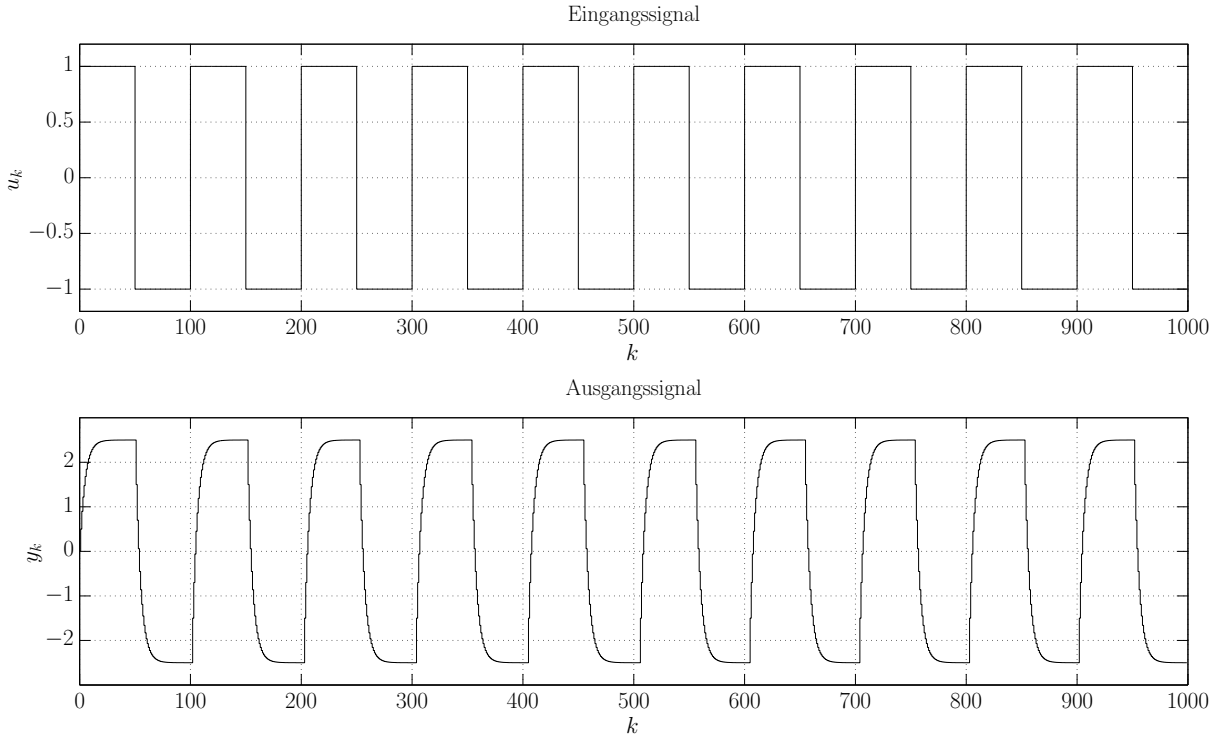

Abbildung 23: Auswahl geeignetste Methode - Ein-/Ausgang Rechteck

Die zugehörigen, geschätzten Totzeiten sehen wie folgt aus:

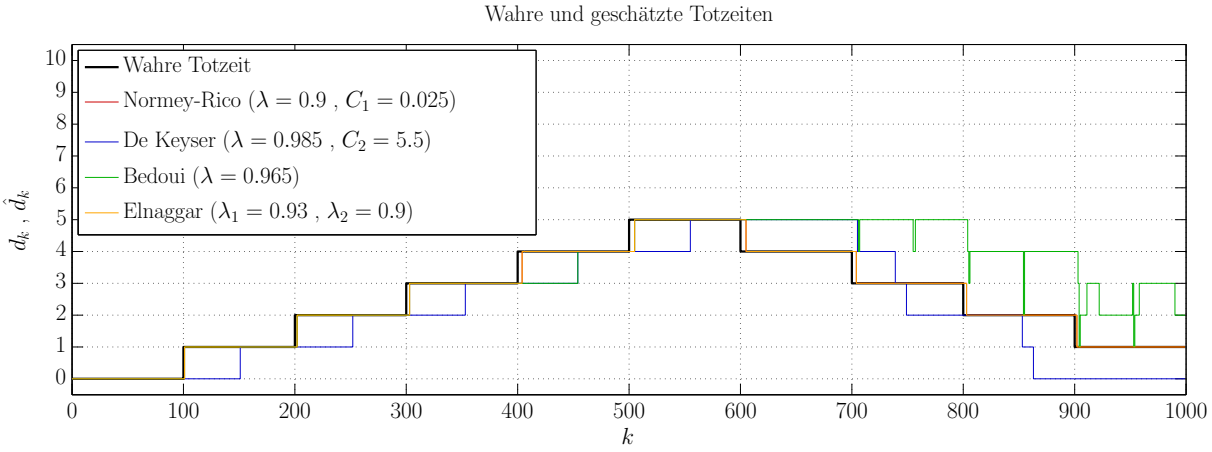

Abbildung 24: Auswahl geeignetste Methode - Totzeit Rechteck

Auch hier erhält man brauchbare Ergebnisse. Lediglich die Methode nach De Keyser hinkt ein wenig nach und der Algorithmus nach Bedoui zeigt die gewohnten Schwächen bei sinkender Totzeit.

Die nachfolgende Tabelle 2 veranschaulicht alle obigen Ergebnisse hinsichtlich der Qualität der Totzeitschätzung aufgeschlüsselt nach Methode und Anregungssignal:

gut	
mittel	
schlecht	

	Binär	Rauschen	Sinus	Sägezahn	Rechteck
Normey-Rico					
De Keyser					
Elnaggar					
Bedoui					

Tabelle 2: Vergleich der RLS-Methoden zur Totzeitschätzung

Man stellt fest, dass einzig der Elnaggar-Algorithmus für alle Testfälle ein gutes Schätzergebnis für die Totzeit liefert. Alle anderen Methoden schlagen in zumindest einem Testfall fehl bzw. weisen insgesamt schlechtere Ergebnisse auch.

Im Zuge dieser Masterarbeit wurden weitergehende Testreihen für Systeme mit weniger Phasenreserve, Systeme höherer Ordnung und für den Einfluss von Ausgangsrauschen durchgeführt. Anhand dieser weiteren Testreihen konnte die obige Aussage bekräftigt werden - der Algorithmus nach Elnaggar lieferte bei sämtlichen Testreihen die besten Resultate bezüglich der Totzeitschätzung.

4 Zusammenfassung

In diesem Kapitel wurden drei verschiedene Herangehensweisen zur modellbasierten On-Line-Schätzung von Totzeiten in zeitdiskreten Systemen vorgestellt. Die erste Methode basierend auf Kalman-Filtern lieferte einen interessanten Ansatz, welcher laut [12] für verschiedenste Modellcharakteristika durchaus brauchbare Ergebnisse liefert. Jedoch ist dieser Ansatz zwingend auf sprungförmige Eingangsgrößen beschränkt, weshalb er für die Zwecke dieser Arbeit ungeeignet ist. Eine zweite vorgestellte Herangehensweise ist die gradientenbasierte Methode mit unmittelbarer Schätzung der Totzeit in einem Schätzvektor. Diese Methode zeigte für Systeme mit konstanten Totzeiten gute Schätzergebnisse, jedoch ergaben sich vor allem bei sinkenden Totzeiten Schwierigkeiten, welche nicht behoben werden konnten. Zuletzt wurden noch mehrere Methoden analysiert, welche auf der rekursiven Least-Squares-Identifikation beruhen. Insgesamt vier solcher Methoden wurden anhand einer Testreihe für unterschiedliche Anregungsarten verglichen. Dabei stellte sich der Elnaggar-Algorithmus (Kapitel 3.3.3) als diejenige Methode mit dem meisten Potential heraus. Sie lieferte für alle durchgeführten Testbeispiele ein gutes Ergebnis und ist weder in der Art des Anregungssignals, noch in der Modellordnung eingeschränkt. Daher wird dieser Ansatz in den nachstehenden Kapiteln weiterverfolgt und wenn nötig für weiterführende Anwendungen angepasst.

E Adaptierung für Regelkreise

In Kapitel D3.3.3 wurde eine Methode zur On-Line-Schätzung von Totzeiten in zeitdiskreten Systemen vorgestellt, welche das Potential hat, die Zielsetzung der vorliegenden Arbeit zu erfüllen: Der Elnaggar-Algorithmus. Dieser ist weder bezüglich des Anregungssignals, noch bezüglich der Modellcharakteristik eingeschränkt. In diesem Kapitel soll nun eine Adaptierung des genannten Ansatzes für Regelkreise vollzogen werden. Unter dem Begriff *Regelkreis* wird in dieser Arbeit ein zeitdiskreter Standard-Regelkreis verstanden. Dies entspricht einer Regelstruktur mit einem Regler $R(z)$ und einer Strecke $P(z)$ im Vorwärtszweig und einer negativen Rückkopplung. Als Eingangsgröße für den Regler dient die Differenz zwischen einer Referenz r_k und dem Streckenausgang y_k, die Ausgangsgröße des Reglers ist die Stellgröße u_k. Eben diese Stellgröße u_k dient der Strecke $P(z)$ als Eingangssignal, der zugehörigen Ausgang ist y_k:

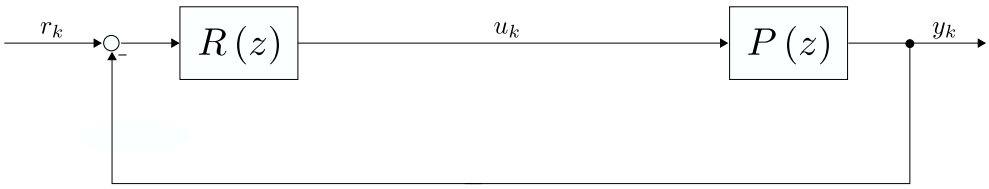

Abbildung 25: Standard-Regelkreis

Wie in der Einleitung dieser Arbeit erläutert, soll die Totzeitschätzung in einer Co-Simulations-Software implementiert werden. Diese Co-Simulations-Software dient unter anderem dazu, unterschiedlichste Subsysteme eines Gesamtsystems miteinander zu koppeln, um eine Simulation des Gesamtverhaltens zu ermöglichen. Diese Arbeit beschränkt sich darauf, den zeitdiskreten Standard-Regelkreis nach Abbildung 25 als Gesamtsystem zu betrachten, wobei der Regler inklusive der Differenzenbildung zwischen r_k und y_k das erste Subsystem bildet und die Strecke das zweite Subsystem darstellt. Die Kopplung geschieht durch ein Koppelelement zwischen ebendiesen beiden Subsystemen, wodurch sich folgende Struktur ergibt:

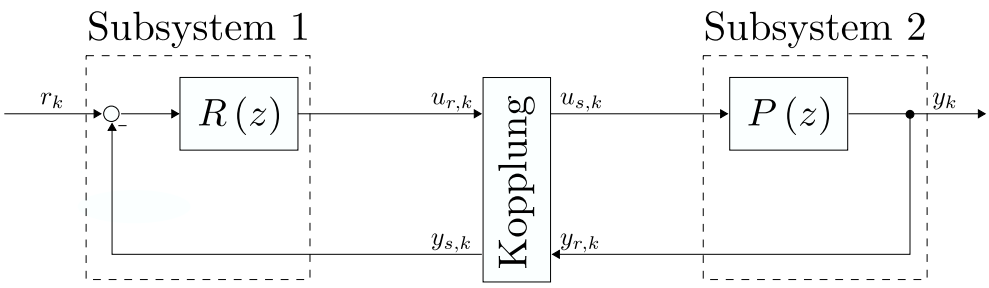

Abbildung 26: Standard-Regelkreis mit Koppelelement

Innerhalb des in der Co-Simulations-Software bereits vorhandenen Koppelelements werden die Signale $u_{r,k}$ und $y_{r,k}$ konditioniert, beispielsweise besteht hier die Möglichkeit einer Kompensation von Totzeiteffekten durch Extrapolation [27]. Auf die genauen Verarbeitungsmechanismen soll hier nicht weiter eingegangen werden, wichtig für den weiteren Verlauf ist nur, dass die Signale $u_{r,k}$ bzw. $y_{r,k}$ durch die Verarbeitung in der Kopplung nicht mehr mit den Signalen $u_{s,k}$ bzw. $y_{s,k}$ übereinstimmen. Die Indizes r und s stehen dabei für *send* (senden) und *receive* (empfangen). Man geht nun davon aus, dass jeweils am Eingang und am Ausgang der beiden Subsysteme (Regler und Strecke) mit Ausnahme des Referenz-Eingangs des Reglers zeitvariable Totzeiten auftreten. Damit verändert sich die Struktur in Abbildung 26 wie folgt:

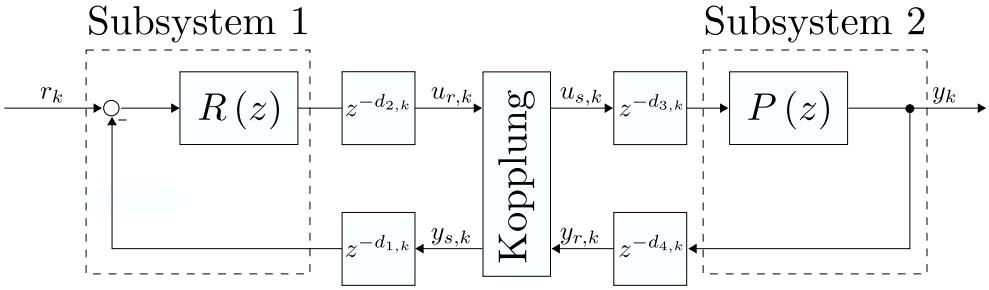

Abbildung 27: Standard-Regelkreis mit Koppelelement und Totzeiten

Die Aufgabe besteht nun darin die zeitvariablen Totzeiten $d_{1,k}$, $d_{2,k}$, $d_{3,k}$ und $d_{4,k}$ mit Hilfe des Elnaggar-Algorithmus (D.3.34) innerhalb der Kopplung zu schätzen. In weiterer Folge wird hierfür zwischen der *Streckenseite* und der *Reglerseite* unterschieden. Unter der Streckenseite wird hierbei das zweite Subsystem mit den Totzeiten $d_{3,k}$ und $d_{4,k}$ verstanden, während die Reglerseite aus dem ersten Subsystem und den Totzeiten $d_{1,k}$ und $d_{2,k}$ besteht.

1 Streckenseite

Zur Schätzung der Totzeiten $d_{3,k}$ und $d_{4,k}$, also der Totzeiten auf der Streckenseite stehen die Signale $u_{s,k}$ und $y_{r,k}$ zur Verfügung. Wie in Kapitel B1.1 gezeigt wurde, kann anhand dieser beiden Signale nicht zwischen den beiden Totzeiten unterschieden werden. Stattdessen kann nur ihre Summe geschätzt werden, weshalb man die zusammengefasste Totzeit $d_k := d_{3,k} + d_{4,k}$ einführt. Das Strukturbild aus Abbildung 27 verändert sich somit folgendermaßen:

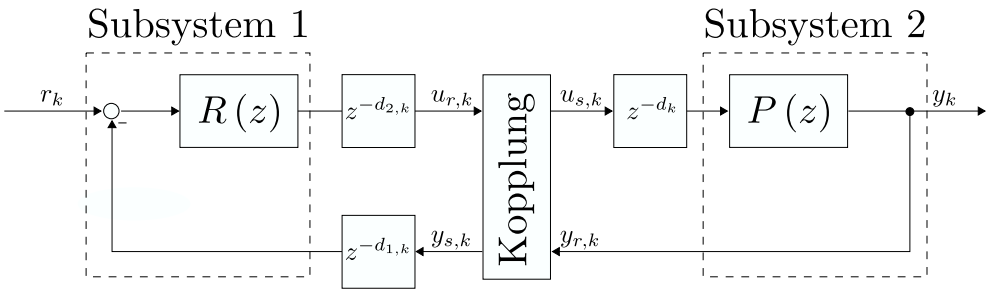

Abbildung 28: Standard-Regelkreis mit zusammengefasster Totzeit

Anzumerken ist an dieser Stelle, dass diese Zusammenfassung keinerlei Nachteile mit sich bringt, da es für das Verhalten eines geregelten Systems ohnehin nicht von Belang ist, ob die Totzeit am Eingang oder am Ausgang der Strecke auftritt. Auch dies wurde in Kapitel B1.1 für den zeitkontinuierlichen Fall erläutert. Durch die eingeführte Abkürzung d_k sieht der geschätzte Messvektor $\hat{\phi}_k$ für den Algorithmus wie folgt aus:

$$\hat{\phi}_k = \hat{\phi}_k\left(\hat{d}_{k-1}\right) = \begin{pmatrix} -y_{r,k-1} \\ \vdots \\ -y_{r,k-n} \\ u_{s,k+m-n-\hat{d}_{k-1}} \\ \vdots \\ u_{s,k-n-\hat{d}_{k-1}} \end{pmatrix} \tag{E.1.1}$$

Entsprechend ist auch der angepasste Messvektor für die Berechnung des Modellfehlers der jeweiligen Gütefunktionen $J_k(d)$ aufgebaut:

$$\phi_k\left(d\right) = \begin{pmatrix} -y_{r,k-1} \\ \vdots \\ -y_{r,k-n} \\ u_{s,k+m-n-d} \\ \vdots \\ u_{s,k-n-d} \end{pmatrix} \tag{E.1.2}$$

Mit dem geschätzten Messvektor gemäß (E.1.1) kann der Algorithmus (D.3.34) für die Streckenseite, also das zweite Subsystem gemeinsam mit der Totzeit, ohne weitere Veränderungen eingesetzt werden:

$$\mathbf{k}_k = \frac{\mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}{\lambda_1 + \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k \Big(y_{r,k} - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big)$$

$$\mathbf{P}_k = \frac{1}{\lambda_1} \Big(\mathbf{P}_{k-1} - \mathbf{k}_k \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \Big) \tag{E.1.3}$$

$$J_k\Big(d\Big) = \lambda_2 J_{k-1}\Big(d\Big) + \Big[y_{r,k} - \boldsymbol{\phi}_k^T\Big(d\Big)\hat{\boldsymbol{\theta}}_k \Big]^2 \quad \forall \quad d \in \Big[d_{\min}, d_{\max} \Big]$$

$$\hat{d}_k = \arg\min \Big\{ J_k\Big(d\Big) \Big\} \quad \forall \quad d \in \Big[d_{\min}, d_{\max} \Big]$$

2 Reglerseite

Im Gegensatz zur Streckenseite ist eine derart einfache Adaptierung des Elnaggar-Algorithmus für die Reglerseite nicht möglich. Das Problem besteht darin, dass man es beim Subsystem auf der linken Seite nicht mit einem SISO- (*Single Input Single Output*) sondern mit einem MISO-System (*Multiple Input Single Output*) zu tun hat. Für die Reglerseite inklusive der Totzeiten $d_{1,k}$ und $d_{2,k}$ dienen r_k und $y_{s,k}$ als Eingangssignale und $u_{r,k}$ stellt das Ausgangssignal dar. Damit wird klar, dass die Signale $u_{r,k}$ und $y_{s,k}$ für die Kopplung nicht ausreichen, um die Koeffizienten des Reglers schätzen zu können. Genau die Schätzung dieser Koeffizienten wird jedoch vom Elnaggar-Algorithmus benötigt, um eine Aussage über die Totzeiten treffen zu können. Zur Schätzung der Totzeiten $d_{1,k}$ und $d_{2,k}$ ist es somit absolut notwendig, dass auch die Referenz r_k zum Koppelelement geführt wird:

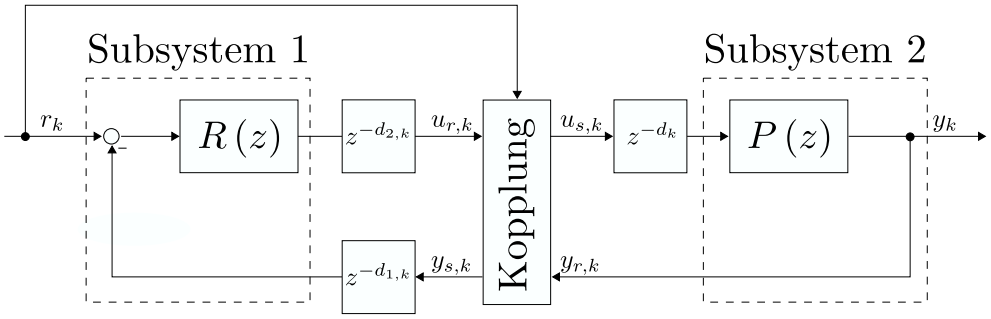

Abbildung 29: Standard-Regelkreis mit Referenz als Kopplungs-Eingang

Im Koppelelement stehen nun also die drei Signale r_k, $y_{s,k}$ und $u_{r,k}$ zur Verfügung, um die Totzeiten $d_{1,k}$ und $d_{2,k}$ zu schätzen. Um auch für diese Aufgabe den Elnaggar-Algorithmus (D.3.34) verwenden zu können, darf man die Signale r_k und $y_{s,k}$ jedoch nicht als separate Eingänge betrachten, sondern man betrachtet die Differenz der beiden Signale als alleinigen Eingang. Man verwendet hier also Vorwissen über das Innenleben des Subsystems, nämlich dass es sich um Regler mit Regelfehler als Eingangsgröße handelt. Zur Adaptierung des Elnaggar-Algorithmus auf der Reglerseite muss zudem beachtet werden, dass die Signale r_k und $y_{s,k}$ zur Bildung des geschätzten Messvektors $\hat{\phi}_k$ entsprechend den zuletzt geschätzten Totzeiten $\hat{d}_{1,k-1}$ und $\hat{d}_{2,k-1}$ verzögert werden müssen:

$$\hat{\phi}_k = \hat{\phi}_k\left(\hat{d}_{1,k-1}, \hat{d}_{2,k-1}\right) = \begin{pmatrix} -u_{r,k-1} \\ \vdots \\ -u_{r,k-n} \\ r_{k+m-n-\hat{d}_{2,k-1}} - y_{s,k+m-n-\hat{d}_{1,k-1}-\hat{d}_{2,k-1}} \\ \vdots \\ r_{k-n-\hat{d}_{2,k-1}} - y_{s,k-n-\hat{d}_{1,k-1}-\hat{d}_{2,k-1}} \end{pmatrix} \tag{E.2.1}$$

Außerdem hängen die Gütefunktionen hier nicht wie im Algorithmus (D.3.34) von einer, sondern hier von jeweils zwei verschiedenen Totzeiten ab. Die Totzeitschätzungen $\hat{d}_{1,k}$ und $\hat{d}_{2,k}$ in jedem Zeitschritt k erhält man dann aus derjenigen Totzeitkombination, für welche die Gütefunktion $J_k(d_1, d_2)$ minimal wird. Der angepasste Messvektor zur Berechnung des jeweiligen Modellfehlers sieht wie folgt aus:

$$\phi_k = \phi_k\left(d_1, d_2\right) = \begin{pmatrix} -u_{r,k-1} \\ \vdots \\ -u_{r,k-n} \\ r_{k+m-n-d_2} - y_{s,k+m-n-d_1-d_2} \\ \vdots \\ r_{k-n-d_2} - y_{s,k-n-d_1-d_2} \end{pmatrix} \tag{E.2.2}$$

Folgender Algorithmus für die Schätzung der beiden Totzeiten $d_{1,k}$ und $d_{2,k}$ auf Reglerseite ergibt sich also:

$$\mathbf{k}_k = \frac{\mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}{\lambda_1 + \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k \Big(u_{r,k} - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big)$$

$$\mathbf{P}_k = \frac{1}{\lambda_1} \Big(\mathbf{P}_{k-1} - \mathbf{k}_k \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \Big) \tag{E.2.3}$$

$$J_k\Big(d_1, d_2 \Big) = \lambda_2 J_{k-1}\Big(d_1, d_2 \Big) + \Big[u_{r,k} - \boldsymbol{\phi}_k^T \Big(d_1, d_2 \Big) \hat{\boldsymbol{\theta}}_k \Big]^2 \quad \forall \quad d_1, d_2 \in \Big[d_{\min}, d_{\max} \Big]$$

$$\Big(\hat{d}_{1,k}, \hat{d}_{2,k} \Big) = \arg\min \Big\{ J_k \Big(d_1, d_2 \Big) \Big\} \quad \forall \quad d_1, d_2 \in \Big[d_{\min}, d_{\max} \Big]$$

Es fällt auf, dass im obigen Algorithmus in jedem Zeitschritt $(d_{\max} - d_{\min} + 1)^2$ Gütefunktionen $J_k(d_1, d_2)$ ausgewertet werden müssen, um alle möglichen Totzeiten $d_1, d_2 \in [d_{\min}, d_{\max}]$ zu berücksichtigen. Dies bringt bei großen Werten für $d_{\max}$ einen erheblichen Rechenaufwand mit sich. Außerdem wurde festgestellt, dass der Algorithmus durch die vielen möglichen Totzeitkombinationen dazu neigt, auf falsch geschätzten Werten für $d_{1,k}$ und $d_{2,k}$ zu verharren. Daher wurde zur Verringerung des Rechenaufwands und zur Verbesserung der Konvergenzeigenschaften eine weitere Adaptierung für die Reglerseite eingeführt. Diese besteht darin, dass in jedem Zeitschritt zuerst die nur Totzeit $d_{2,k}$ und anschließend nur die Totzeit $d_{1,k}$ geschätzt wird. Die beiden geschätzten Totzeiten $\hat{d}_{1,k}$ und $\hat{d}_{2,k}$ werden also nicht in einem Schritt gemeinsam, sondern sequentiell ermittelt. Man berechnet zunächst die Gütefunktionen $J_{2,k}(d_2)$ für die Schätzung $\hat{d}_{2,k}$, wobei die angepassten Messvektoren für die Modellfehler die zuletzt geschätzte Totzeit $\hat{d}_{1,k-1}$ miteinbeziehen:

$$\boldsymbol{\phi}_{2,k} = \boldsymbol{\phi}_{2,k}\Big(\hat{d}_{1,k-1}, d_2 \Big) = \begin{pmatrix} -u_{r,k-1} \\ \vdots \\ -u_{r,k-n} \\ r_{k+m-n-d_2} - y_{s,k+m-n-\hat{d}_{1,k-1}-d_2} \\ \vdots \\ r_{k-n-d_2} - y_{s,k-n-\hat{d}_{1,k-1}-d_2} \end{pmatrix} \tag{E.2.4}$$

Liegt dann eine Schätzung $\hat{d}_{2,k}$ vor, so kann man diese anschließend gleich zur Bildung der angepassten Messvektoren für die Gütefunktionen $J_{1,k}(d_1)$ verwenden:

$$\boldsymbol{\phi}_{1,k} = \boldsymbol{\phi}_{1,k}\Big(d_1, \hat{d}_{2,k} \Big) = \begin{pmatrix} -u_{r,k-1} \\ \vdots \\ -u_{r,k-n} \\ r_{k+m-n-\hat{d}_{2,k}} - y_{s,k+m-n-d_1-\hat{d}_{2,k}} \\ \vdots \\ r_{k-n-\hat{d}_{2,k}} - y_{s,k-n-d_1-\hat{d}_{2,k}} \end{pmatrix} \tag{E.2.5}$$

Der Algorithmus (E.2.3) wird damit durch den folgenden Algorithmus ersetzt:

$$\mathbf{k}_k = \frac{\mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}{\lambda_1 + \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k\Big(u_{r,k} - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \Big)$$

$$\mathbf{P}_k = \frac{1}{\lambda_1}\Big(\mathbf{P}_{k-1} - \mathbf{k}_k \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \Big)$$

$$J_{2,k}\Big(d_2 \Big) = \lambda_2 J_{2,k-1}\Big(d_2 \Big) + \Big[u_{r,k} - \boldsymbol{\phi}_k^T\Big(\hat{d}_{1,k-1}, d_2 \Big)\hat{\boldsymbol{\theta}}_k \Big]^2 \quad \forall \quad d_2 \in \Big[d_{\min}, d_{\max} \Big] \tag{E.2.6}$$

$$\hat{d}_{2,k} = \arg\min\Big\{ J_{2,k}\Big(d_2 \Big) \Big\} \quad \forall \quad d_2 \in \Big[d_{\min}, d_{\max} \Big]$$

$$J_{1,k}\Big(d_1 \Big) = \lambda_2 J_{1,k-1}\Big(d_1 \Big) + \Big[u_{r,k} - \boldsymbol{\phi}_{1,k}^T\Big(d_1, \hat{d}_{2,k} \Big)\hat{\boldsymbol{\theta}}_k \Big]^2 \quad \forall \quad d_1 \in \Big[d_{\min}, d_{\max} \Big]$$

$$\hat{d}_{1,k} = \arg\min\Big\{ J_{1,k}\Big(d_1 \Big) \Big\} \quad \forall \quad d_1 \in \Big[d_{\min}, d_{\max} \Big]$$

Im Algorithmus (E.2.6) müssen anstatt $(d_{\max} - d_{\min} + 1)^2$ nur mehr $2(d_{\max} - d_{\min} + 1)$ Gütefunktionen pro Zeitschritt berechnet werden, was eine erhebliche Steigerung der Recheneffizienz nach sich zieht.

3 Beispiel

Mit den Adaptionen (E.1.3) und (E.2.6) des Elnaggar-Algorithmus ist es nun möglich innerhalb eines Koppelelements die Totzeiten d_k bzw. $d_{1,k}$ und $d_{2,k}$ anhand der Signale $u_{s,k}$ und $y_{r,k}$ bzw. r_k, $y_{s,k}$ und $u_{r,k}$ on-line zu schätzen (siehe Abbildung 29).

Ein Beispiel soll nun die Funktionstüchtigkeit der Adaptionen veranschaulichen. Hierzu sei eine einfache kontinuierliche Strecke erster Ordnung angenommen:

$$P\Big(\, s \,\Big) = \frac{0.2}{s + 0.1} \tag{E.3.1}$$

Für diese Strecke ergibt sich eine Phasenreserve von $\phi_r = 120°$ bei der Durchtrittsfrequenz $\omega_c = 0.173\,\mathrm{rad/s}$. Als Regler wird ein klassischer PI-Regler mir Proportionalfaktor $K_P = 100$ und Nachstellzeit $T_N = 0.1$ eingesetzt:

$$R\Big(\, s \,\Big) = K_P\left(\, 1 + \frac{1}{T_N s} \,\right) = 100\left(\, 1 + \frac{1}{0.1 s} \,\right) \tag{E.3.2}$$

Durch diesen Regler ergibt sich für den offenen Kreis $L(s) = P(s)R(s)$ eine Phasenreserve von $\phi_r = 65.8°$ bei der Durchtrittsfrequenz $\omega_c = 22\,\mathrm{rad/s}$. $P(s)$ und $R(s)$ werden jetzt mit einer Abtastzeit von $T_S = 0.01\,\mathrm{s}$ diskretisiert, womit sich die folgenden beiden, zeitdiskreten Übertragungsfunktionen ergeben:

$$P\Big(\, z \,\Big) = \frac{0.001999}{z - 0.999} \quad ; \quad R\Big(\, z \,\Big) = \frac{100z - 90}{z - 1} \tag{E.3.3}$$

Gemäß der Struktur in Abbildung 29 treten nun zeitvariable Totzeiten d_k, $d_{1,k}$ und $d_{2,k}$ auf:

$$d_k = \begin{cases} 0 & \text{für} \quad 0 \le \mathrm{k} \le 299 \\ 2 & \text{für} \quad 300 \le \mathrm{k} \le 1099 \\ 1 & \text{für} \quad 1100 \le \mathrm{k} \le 1499 \\ 0 & \text{für} \quad 1500 \le \mathrm{k} \le 1999 \end{cases}$$

$$d_{1,k} = \begin{cases} 0 & \text{für} \quad 0 \le \mathrm{k} \le 849 \\ 1 & \text{für} \quad 850 \le \mathrm{k} \le 1249 \\ 2 & \text{für} \quad 1250 \le \mathrm{k} \le 1649 \\ 1 & \text{für} \quad 1650 \le \mathrm{k} \le 1999 \end{cases} \tag{E.3.4}$$

$$d_{2,k} = \begin{cases} 1 & \text{für} \quad 0 \le \mathrm{k} \le 449 \\ 0 & \text{für} \quad 450 \le \mathrm{k} \le 849 \\ 1 & \text{für} \quad 850 \le \mathrm{k} \le 1649 \\ 2 & \text{für} \quad 1650 \le \mathrm{k} \le 1999 \end{cases}$$

Als Referenz r_k wird ein sprungförmiges Signal vorgegeben, welches zwischen den Werten 0 , 1 und 2 springt:

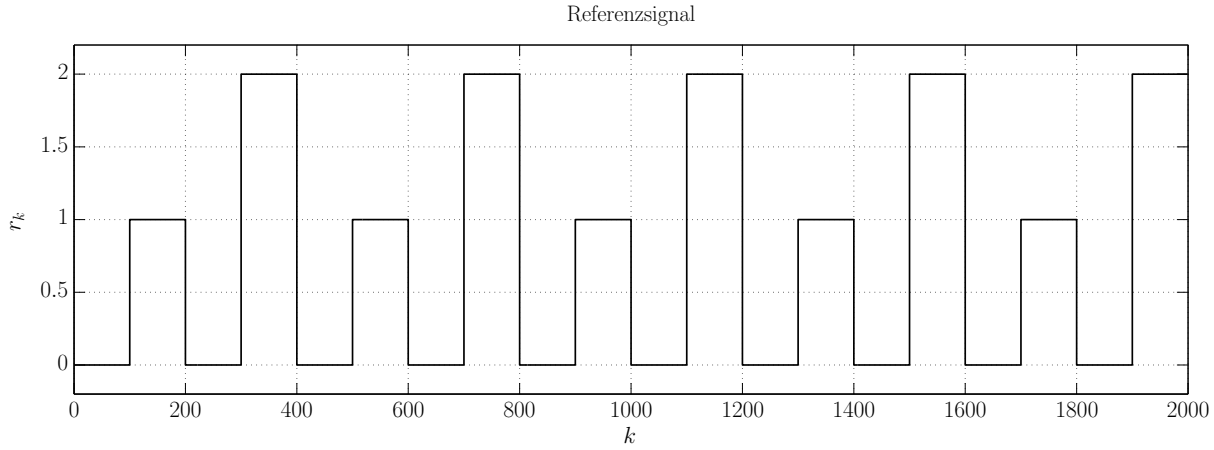

Abbildung 30: Beispiel zeitvariable Totzeiten - Referenzsignal

Gemeinsam mit dem Regler $R(z)$ und der Strecke $P(z)$ ergeben sich damit die in Abbildung 31 zu sehenden Signale u_k und y_k:

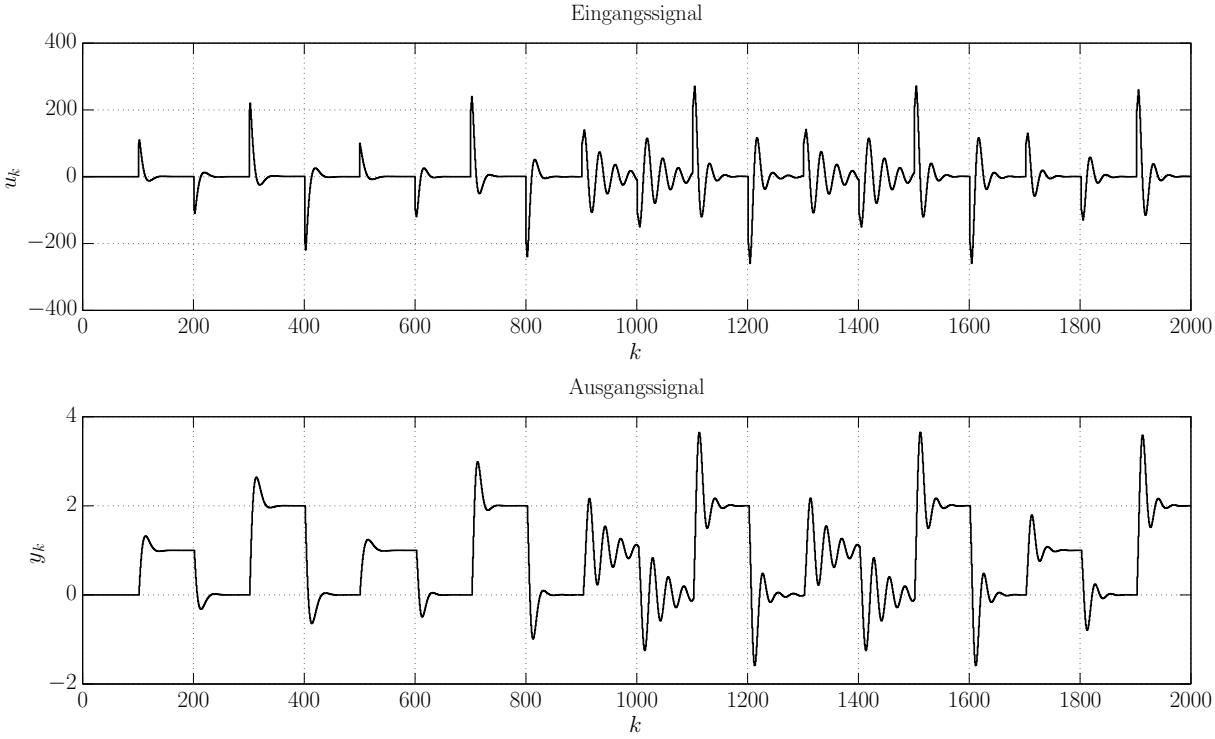

Abbildung 31: Beispiel zeitvariable Totzeiten - Ein- und Ausgangssignal

Man kann erkennen, dass sich das Überschwingen im geregelten System mit zunehmenden Totzeiten teils erheblich erhöht. Die drei Signale r_k (Abbildung 30), u_k und y_k (Abbildung 31) dienen nun als Eingangsgrößen für das Koppelelement, wobei hier

$u_{r,k} = u_{s,k} = u_k$ und $y_{r,k} = y_{s,k} = y_k$ gilt. Anhand der Algorithmen (E.1.3) und (E.2.6) können damit die Totzeiten d_k, $d_{1,k}$ und $d_{2,k}$ on-line geschätzt werden. Hierfür wurden die Vergessensfaktoren aufgrund der zeitinvarianten Modellparameter und der zeitvariablen Totzeit für beide Algorithmen mit $\lambda_1 = 1$ und $\lambda_2 = 0.95$ gewählt. Außerdem wurden die Matrix $\mathbf{P}_0$, der Schätzvektor $\hat{\boldsymbol{\theta}}_0$ und die geschätzte Totzeit $\hat{d}_0$ gemäß (D.3.41) initialisiert. Abbildung 32 zeigt die Schätzergebnisse:

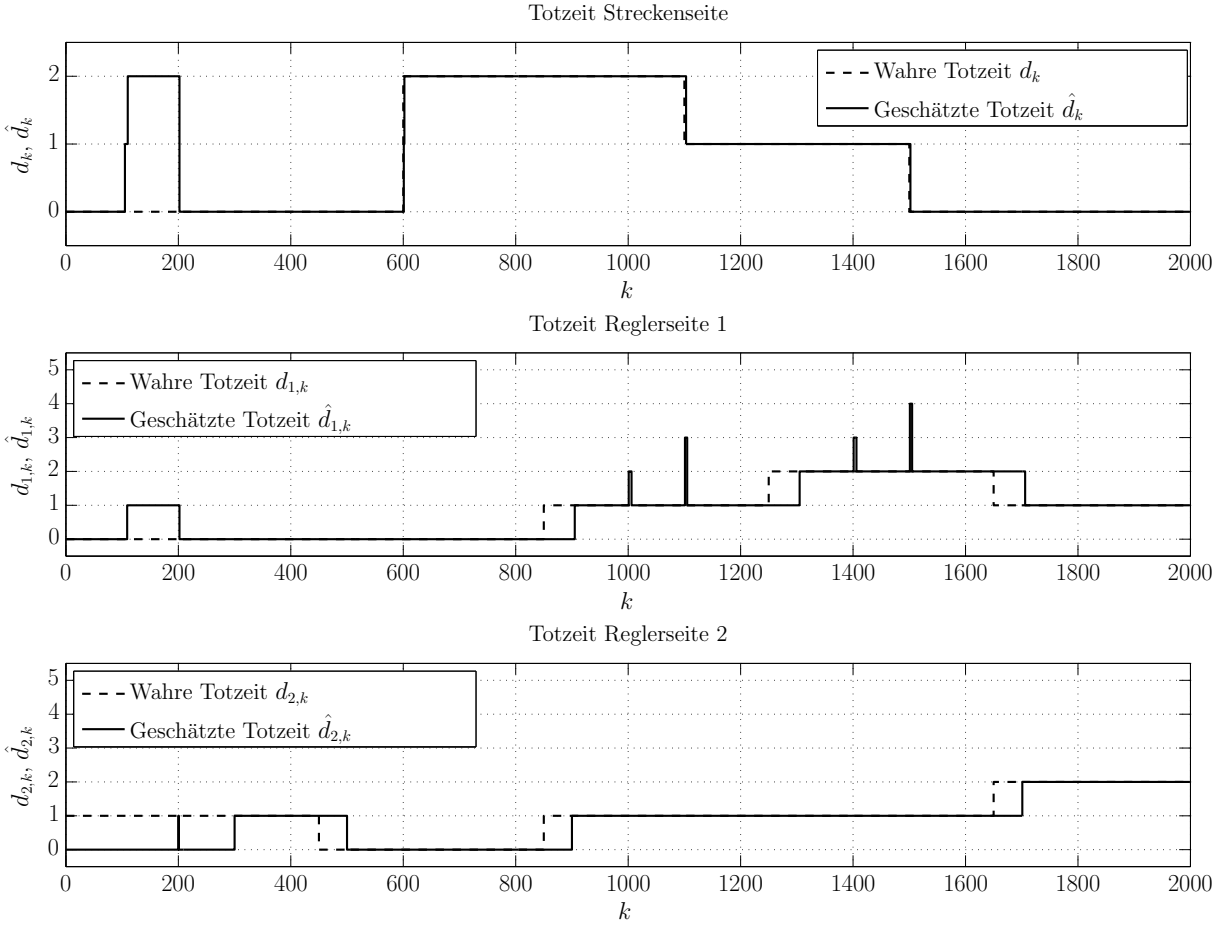

Abbildung 32: Beispiel zeitvariable Totzeiten - Wahre und geschätzte Totzeiten

Man kann beobachten, dass alle drei Totzeiten relativ zuverlässig geschätzt werden. Lediglich ein kurzer Einschwingvorgang und ein kleiner Zeitverzug in der Schätzung treten auf, was jedoch ohne Probleme hingenommen werden kann.

Zusammenfassend lässt sich festhalten, dass es in diesem Kapitel gelungen ist, den Elnaggar-Algorithmus so zu adaptieren, dass er zur On-Line-Totzeitschätzung in einem Standard-Regelkreis nach Abbildung 29 verwendet werden kann.

F Verbesserung der Anwendbarkeit

In Kapitel E wurde die in Kapitel D3.3.3 vorgestellte Elnaggar-Methode für die On-Line-Totzeitschätzung innerhalb eines Koppelelements in Regelkreisen adaptiert. Im vorliegenden Kapitel soll es nun darum gehen, die Anwendbarkeit dieser Methode durch geeignete Maßnahmen zu verbessern. Hierbei kommen zunächst die Berücksichtigung von Einschwingvorgängen (Kapitel 1) und die Optimierung der Vergessensfaktoren (Kapitel 2) zur Sprache. Anschließend wird eine alternative Vergessensmethode zur Verringerung der Rauschsensibilität (Kapitel 3) und eine Sprungerkennung für Arbeitspunktwechsel (Kapitel 4) eingeführt.

1 Berücksichtigung von Einschwingvorgängen

Um die Zuverlässigkeit der Methode nach Elnaggar zu erhöhen, ist es zur Verbesserung der Parameterkonvergenz empfehlenswert, Einschwingvorgänge zu berücksichtigen. Betrachtet man die für Regelkreise adaptierten Algorithmen (E.1.3) und (E.2.6), so wird deutlich, dass diese vor allem dann gut funktionieren werden, wenn die Einträge des Schätzvektors $\hat{\boldsymbol{\theta}}_k$ den wahren Modellparametern zum Zeitschritt k sehr nahe kommen. Dies gewährleistet, dass die Gütefunktionen $J_k(d)$ bzw. $J_k(d_1, d_2)$ ein möglichst genaues Bild über die wahren Totzeiten d_k bzw. $d_{1,k}$ und $d_{2,k}$ zu diesem Zeitschritt abgeben. Bei gut geschätzten Modellparametern kann man also annehmen, dass diejenigen Gütefunktionen die kleinsten Werte annehmen, die zu den wahren Totzeiten d_k bzw. $d_{1,k}$ und $d_{2,k}$ gehören. Wenn sich der Schätzvektor $\hat{\boldsymbol{\theta}}_k$ jedoch sehr stark vom wahren Parametervektor $\boldsymbol{\theta}_k$ unterscheidet, nehmen die Gütefunktionen Werte an, aus denen nur wenig brauchbare Information über die wahren Totzeiten ableitbar ist. Dies liegt daran, dass der Modellfehler in diesem Fall für alle angenommenen Totzeiten verhältnismäßig groß ist, wodurch die zu den wahren Totzeiten d_k bzw. $d_{1,k}$ und $d_{2,k}$ gehörigen Gütefunktionen nicht mehr zwingend den kleinsten Wert annehmen. In weiterer Folge kann es passieren, dass sich der Algorithmus auf eine falsch geschätzte Totzeit bzw. auf falsch geschätzte Totzeiten versteift, wodurch sich auch die geschätzten Modellparameter nicht mehr den wahren Modellparametern annähern können. Dieser Effekt kann insbesondere dann beobachtet werden, wenn Mess- bzw. Ausgangsrauschen auftritt. Zur Veranschaulichung wird die folgende lineare, zeitinvariante Strecke mit Zählergrad $m = 0$ und Nennergrad $n = 1$ betrachtet:

$$P\Big(\, z \,\Big) = \frac{0.5}{z + 0.8}\, z^{-4} \tag{F.1.1}$$

Diese Strecke weist neben den konstanten Modellparametern $b_{0,k} = b_0 = 0.5$ bzw. $a_{0,k} = a_0 = 0.8$ offensichtlich auch eine konstante Totzeit von $d_k = d = 4$ auf. Es wird nun eine Open-Loop-Totzeitschätzung anhand des Algorithmus (E.1.3) durchgeführt, d.h. es wird alleinig die Streckenseite in Abbildung 29 analysiert. Als Anregungssignal $u_{s,k}$ wurde eine Random-Binärfolge mit einer Länge von 1000 Samples gewählt und es tritt ein Ausgangsrauschen mit Standardabweichung $\sigma_{y_r} = 0.3$ auf. Außerdem wurden die Vergessensfaktoren aufgrund des zeitinvarianten Systems und der konstanten Totzeit mit $\lambda_1 = 1$ und $\lambda_2 = 1$ gewählt. Die Initialisierung erfolgte wie üblich gemäß Vorschrift (D.3.41). Die folgende Abbildung zeigt die geschätzten Parameter $\hat{b}_{0,k}$ bzw. $\hat{a}_{0,k}$ und die geschätzte Totzeit $\hat{d}_k$ für den Fall, dass die Totzeitschätzung vollständig ab dem ersten Sample gestartet wird:

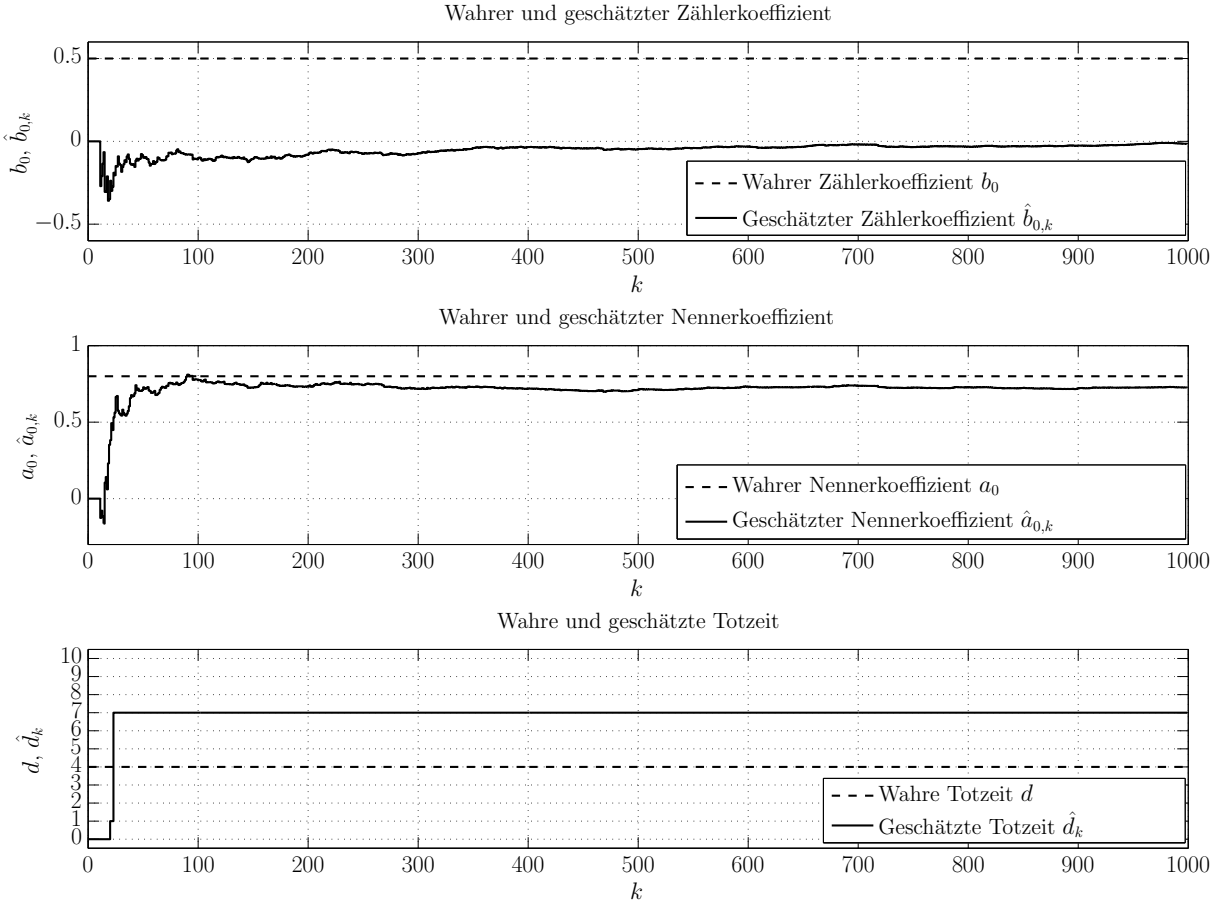

Abbildung 33: Totzeitschätzung ohne Berücksichtigung des Einschwingens

Man erkennt, dass der geschätzte Zählerkoeffizient $\hat{b}_{0,k}$ während der ersten Samples einen Wert aufweist, der weit vom wahren Wert bei $b_0 = 0.5$ entfernt liegt. Aufgrund dieser Tatsache wird der Algorithmus sozusagen in die Irre geleitet und es wird eine falsche Totzeit von $\hat{d} = 7$ geschätzt. Da sich der Algorithmus in weiterer Folge wie beschrieben auf diese falsch geschätzte Totzeit versteift, kann der besagte Zählerkoeffizient auch später nicht mehr richtig geschätzt werden. Der Algorithmus verliert

dadurch die Fähigkeit, die richtige Totzeit von $d = 4$ zu identifizieren. Wie schon angedeutet wäre es sinnvoller, zunächst das Einschwingen der Modellparameter abzuwarten und erst dann die Gütefunktionen für die möglichen Totzeiten auszuwerten. Die letzten beiden Zeilen des Algorithmus (E.1.3) verändern sich dadurch wie folgt:

$$J_k\Big(\,d\,\Big) = \begin{cases} 0 & \text{für } k \le K_1 \\ \lambda_2 J_{k-1}\Big(\,d\,\Big) + \Big[\,y_{r,k} - \boldsymbol{\phi}_k^T\Big(\,d\,\Big)\hat{\boldsymbol{\theta}}_k\,\Big]^2 & \text{für } k > K_1 \end{cases} \quad \forall \quad d \in \Big[\,d_{\min}, d_{\max}\,\Big]$$

$$\hat{d}_k = \begin{cases} d_{\min} & \text{für } k \le K_1 \\ \arg\min\Big\{\,J_k\Big(\,d\,\Big)\Big\} & \text{für } k > K_1 \end{cases} \quad \forall \quad d \in \Big[\,d_{\min}, d_{\max}\,\Big] \tag{F.1.2}$$

Der Parameter K_1 beschreibt hier die Anzahl der Samples, bis zu welcher mit der Auswertung der Totzeit-Gütefunktionen $J_k(d)$ gewartet wird. Die Beziehung (F.1.2) kann in analoger Art und Weise für den Algorithmus (E.2.6), also für die Reglerseite adaptiert werden. Für das Beispiel (F.1.1) ergeben sich mit der Wahl $K_1 = 100$ die folgenden Schätzungen für die Modellparameter b_0 bzw. a_0 und die Totzeit d:

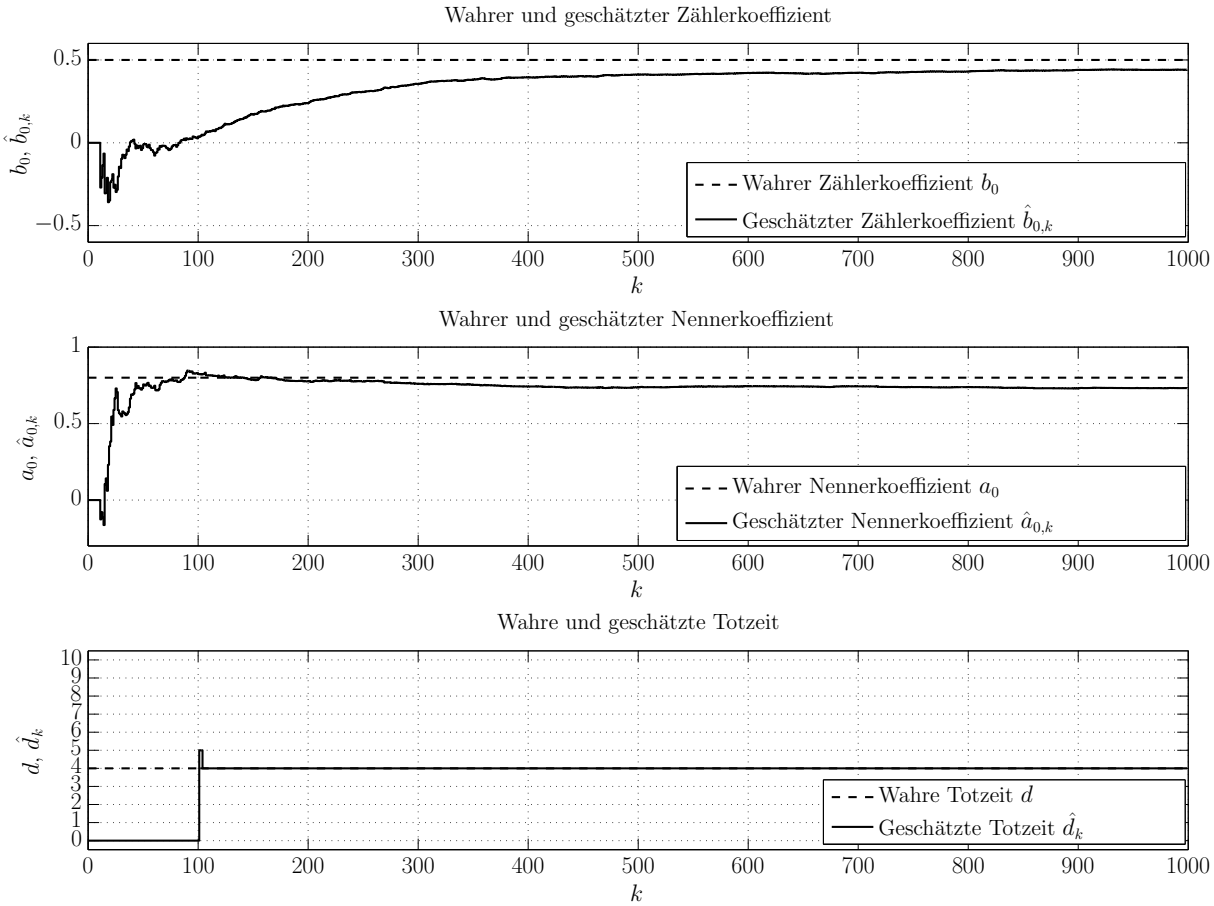

Abbildung 34: Totzeitschätzung mit Berücksichtigung des Einschwingens

Man erkennt, dass der geschätzte Zählerkoeffizient $\hat{b}_{0,k}$ zum Start der Auswertung der Gütefunktionen bei $k = K_1 + 1 = 101$ zwar auch hier nicht korrekt geschätzt wird, jedoch bewegt sich die Schätzung zu diesem Zeitpunkt zumindest schon in die richtige Richtung und trägt das korrekte Vorzeichen. Allein dieser Umstand reich dafür aus, dass die in weiterer Folge ausgewerteten Gütefunktionen $J_k(d)$ Werte annehmen, die eine korrekte Totzeitschätzung ermöglichen. Das heißt, dass die zur wahren Totzeit gehörige Gütefunktion $J_k(d = 4)$ schon sehr früh den vergleichsweise kleinsten Wert annimmt, wodurch die Totzeitschätzung in diesem Fall sehr gut funktioniert.

Durch die Berücksichtigung von Einschwingvorgängen bezüglich der Parameter mit Hilfe der Beziehung (F.1.2) kann eine Irreleitung des für die Strecken- und Reglerseite adaptierten Algorithmus also erfolgreich vermieden werden.

2 Optimierung der Vergessensfaktoren

Eine wichtige Frage, die sich in vielen Anwendungsfällen des Elnaggar-Algorithmus oft ergibt, ist diejenige nach der Wahl der beiden Vergessensfaktoren λ_1 und λ_2. In den bisherigen im Zuge dieser Arbeit aufgeführten Beispielen konnten diese beiden Algorithmus-Parameter aufgrund von Vorwissen sinnvoll festgelegt werden. So ist es bei zeitlich stark variierenden Modellparametern sinnvoll, den Vergessensfaktor λ_1 eher klein, also z.B. im Bereich $\lambda_1 \approx 0.95$ zu wählen. Bei nur langsam variierenden Parametern wählt man für diesen Vergessensfaktor einen etwas größeren Wert und für den zeitinvarianten Fall liegt die Wahl $\lambda_1 = 1$ nahe. Beachtet werden muss jedoch auch der Fall, dass man es mit einem nichtlinearen System zu tun hat. Ein solches System kann bei Verwendung eines rekursiven RLS-Algorithmus nur durch eine lineare Approximation geschätzt werden. Dies führt zu Modellparametern, welche zeitlich variieren, um der nichtlinearen Dynamik des Systems folgen zu können. Auch bei nichtlinearen Systemen wäre somit ein Vergessensfaktor $\lambda_1 < 1$ sinnvoll, obwohl die Parameter des realen, nichtlinearen Systems konstant bleiben. Für die Wahl des zweiten Vergessensfaktors λ_2 zur Auswertung der Gütefunktionen für die möglichen Totzeiten kann man analoge Überlegungen anstellen. Auch hier empfiehlt es sich, bei zeitlich variablen Totzeiten einen kleineren Vergessensfaktor zu wählen, während bei konstanter Totzeit ein Wert nahe Eins vorteilhaft ist.

Die vorangegangenen Erläuterungen zur Wahl passender Vergessensfaktoren λ_1 und λ_2 sollen anhand zweier Beispiele veranschaulicht werden. Das erste Beispiel behandelt eine Strecke mit Zählergrad $m = 0$ und Nennergrad $n = 1$ und den gleichen, konstanten Modellparametern wie die Strecke nach Beziehung (F.1.1). Es tritt jedoch keine konstante, sondern eine zeitlich variierende Totzeit d_k auf:

$$P\Big(\,z\,\Big) = \frac{0.5}{z+0.8}\,z^{-d_k} \qquad \text{mit} \qquad d_k \;= \begin{cases} 0 & \text{für} \quad 0 \le \mathrm{k} \le 99 \\ 1 & \text{für} \quad 100 \le \mathrm{k} \le 199 \\ 3 & \text{für} \quad 200 \le \mathrm{k} \le 299 \\ 4 & \text{für} \quad 300 \le \mathrm{k} \le 399 \\ 3 & \text{für} \quad 400 \le \mathrm{k} \le 499 \\ 5 & \text{für} \quad 500 \le \mathrm{k} \le 599 \\ 4 & \text{für} \quad 600 \le \mathrm{k} \le 699 \\ 2 & \text{für} \quad 700 \le \mathrm{k} \le 799 \\ 3 & \text{für} \quad 800 \le \mathrm{k} \le 899 \\ 1 & \text{für} \quad 900 \le \mathrm{k} \le 999 \end{cases} \tag{F.2.1}$$

Zur Totzeitschätzung wird der Elnaggar-Algorithmus unter Berücksichtigung des Einschwingvorgangs nach (F.1.2) mit $K_1 = 100$ verwendet. Für die minimal und maximal mögliche Totzeit wird $d_{\min} = 0$ und $d_{\max} = 10$ festgelegt und es werden die üblichen Initialisierungswerte nach (D.3.41) verwendet. Außerdem kommen verschiedene Kombinationen der Vergessensfaktoren λ_1 und λ_2 zum Einsatz. Diese werden so gewählt, dass sich die Zeitkonstante des rekursiven Least-Squares-Algorithmus logarithmisch verteilt. Als Zeitkonstante des rekursiven Least-Squares-Algorithmus versteht man diejenige zurückliegende Zeitdauer κ, deren zugehöriger Messvektor $\boldsymbol{\phi}_{k-\kappa}$ mit einer Gewichtung von ungefähr $\mathrm{e}^{-1} \approx 37.8\%$ in die Berechnung von $\mathbf{P}_k^{-1}$ nach Beziehung (D.3.16) eingeht [28]. Diese Zeitkonstante berechnet sich zu:

$$\begin{aligned} \lambda^{\kappa} &= \mathrm{e}^{-1} \\ \ln\Big(\,\lambda^{\kappa}\,\Big) &= \ln\Big(\,\mathrm{e}^{-1}\,\Big) \\ \kappa \ln\Big(\,\lambda\,\Big) &= -1 \\ \kappa &= -\frac{1}{\ln\Big(\,\lambda\,\Big)} \end{aligned} \tag{F.2.2}$$

Der Ausdruck $\ln(\lambda)$ lässt sich durch eine Potenzreihen-Approximation erster Ordnung mit $\ln(\lambda) \approx \lambda - 1$ annähern, wodurch man auf folgende übliche Formel für die Zeitkonstante κ kommt [8]:

$$\kappa = -\frac{1}{\lambda - 1} = \frac{1}{1-\lambda} \tag{F.2.3}$$

Diese Zeitkonstante wird nun wie angemerkt für beide Vergessensfaktoren logarithmisch verteilt. Hierfür verteilt man 5 Werte für κ logarithmisch im Intervall $[20,1000]$, und rechnet mit der Umkehrformel von (F.2.3) auf die entsprechenden Vergessensfaktoren zurück:

$$\lambda = 1 - \frac{1}{\kappa} \tag{F.2.4}$$

Außerdem wurde noch der Vergessensfaktor $\lambda = 1$ (kein Vergessen) hinzugenommen, der einer RLS-Zeitkonstante von $\kappa \to \infty$ entspricht. Tabelle 3 zeigt die gewählten Werte für die Zeitkonstanten und die sich durch (F.2.4) ergebenden Vergessensfaktoren:

κ	20	53	141	376	1000	∞
λ	0.95	0.98120	0.99293	0.99734	0.999	1

Tabelle 3: Verteilung der Zeitkonstanten und Vergessensfaktoren

Die möglichen Kombinationen der Vergessensfaktoren λ_1 und λ_2 ergeben sich aus allen möglichen Paarungen der Einträge in der zweiten Zeile der obigen Tabelle. Die Totzeitschätzung wird nun für jede dieser Kombinationen mit einem Random-Binär-Signal der Länge $N = 1000$ als Eingangssignal $u_{s,k}$ getestet. Das Ausgangssignal $y_{r,k}$ ergibt sich anhand der Strecke (F.2.1) – auch hier wird nur die Streckenseite (open-loop) betrachtet. Um zu überprüfen, welche Vergessensfaktoren λ_1 und λ_2 gut für die Totzeitschätzung geeignet sind, wertet man für jede der möglichen Kombinationen eine Gütefunktion $I_d(\lambda_1, \lambda_2)$ aus, welche aus der Summe der Quadrate der Totzeit-Fehler $e_{d,k}(\lambda_1, \lambda_2)$ besteht. Diese Summe wird ab dem Zeitschritt $k = K_1 + 1$, also ab dem Zeitpunkt, zu welchem die Totzeitschätzung startet, berechnet:

$$\begin{aligned} I_d\left(\lambda_1, \lambda_2\right) &= \sum_{k=K_1+1}^{N} e_{d,k}^2\left(\lambda_1, \lambda_2\right) \\ &= \sum_{k=K_1+1}^{N} \left(d_k - \hat{d}_k\left(\lambda_1, \lambda_2\right)\right)^2 \end{aligned} \tag{F.2.5}$$

Trägt man diese Gütefunktionen über den Vergessensfaktoren auf, dann ergibt sich folgendes Bild:

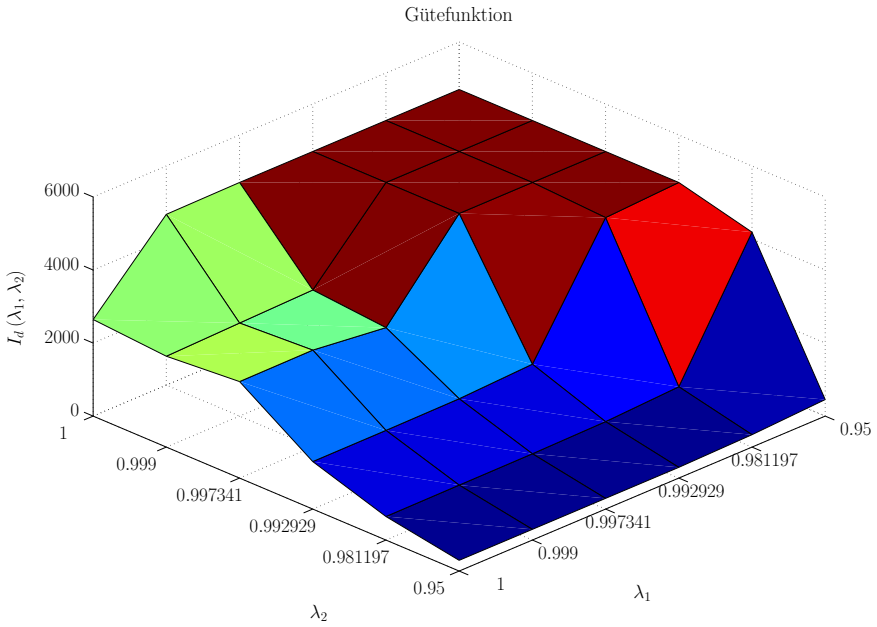

Abbildung 35: Vergleich der Kombinationen der Vergessensfaktoren (1. Beispiel)

Wie zu erwarten war, funktioniert die Totzeitschätzung für einen kleinen Vergessensfaktor $\lambda_2 = 0.95$ am besten, da die Totzeit zeitlich variabel ist (F.2.1). Die Wahl von λ_1 ist relativ unkritisch, jedoch kann man erkennen, dass höhere λ_1 wegen der konstanten Modellparameter ein leicht besseres Ergebnis zeigen. Diese Beobachtungen bestätigt auch Abbildung 36. Man erkennt, dass die Totzeitschätzung für $\lambda_1 = 1$ und $\lambda_2 = 0.95$ problemlos abläuft, während sie für $\lambda_1 = 0.95$ und $\lambda_2 = 1$ fehlschlägt:

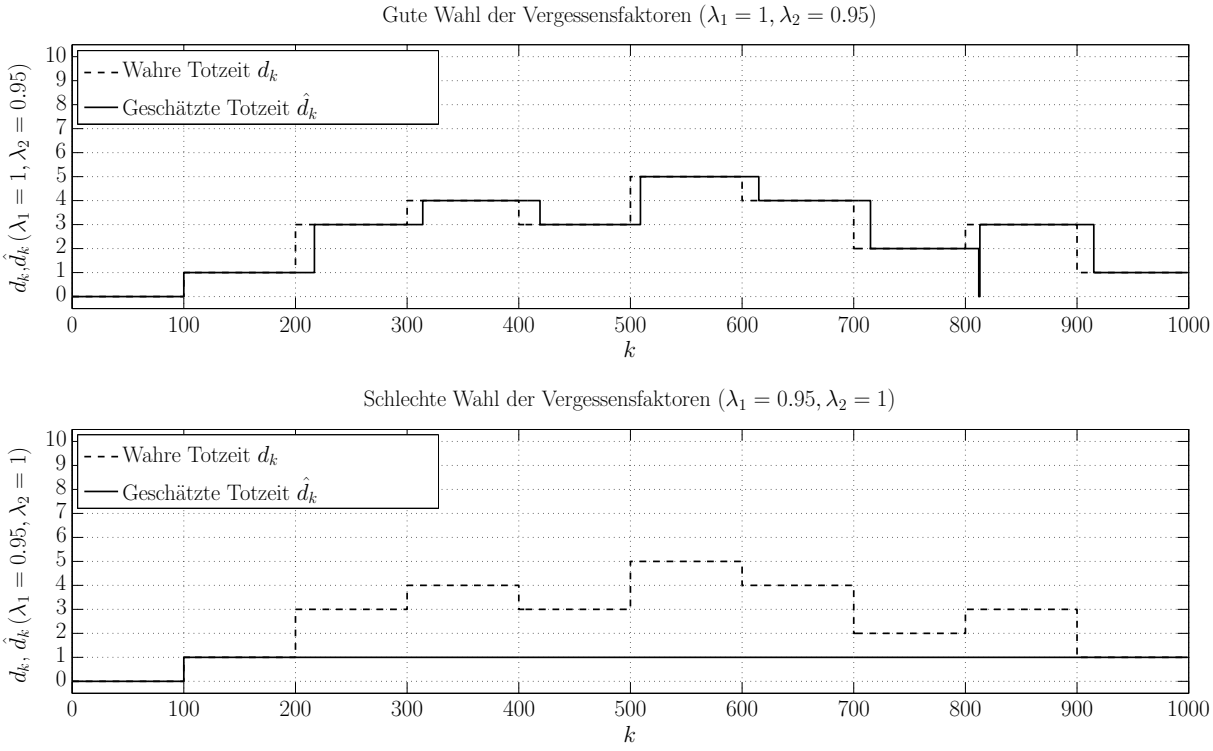

Abbildung 36: Gute und schlechte Wahl der Vergessensfaktoren (1. Beispiel)

Im zweiten Beispiel liegt zwar eine konstante Totzeit $d_k = d = 3$, jedoch ein zeitvariabler Zählerparameter $b_{0,k}$ vor:

$$P\Big(\,z\,\Big) = \frac{b_{0,k}}{z + 0.8}\, z^{-3} \qquad \text{mit} \qquad b_{0,k} \quad = \begin{cases} \;\;0.5 & \text{für} \quad\;\; 0 \leq \mathrm{k} \leq 249 \\ -0.7 & \text{für} \;\; 250 \leq \mathrm{k} \leq 499 \\ \;\;0.5 & \text{für} \;\; 500 \leq \mathrm{k} \leq 749 \\ -0.5 & \text{für} \;\; 750 \leq \mathrm{k} \leq 999 \end{cases} \tag{F.2.6}$$

Das Eingangssignal $u_{s,k}$ ist nun ein Rechtecksignal. Auch hier wurde die Gütefunktion $I_d(\lambda_1, \lambda_2)$ wieder für alle möglichen Kombinationen der Vergessensfaktoren ausgewertet:

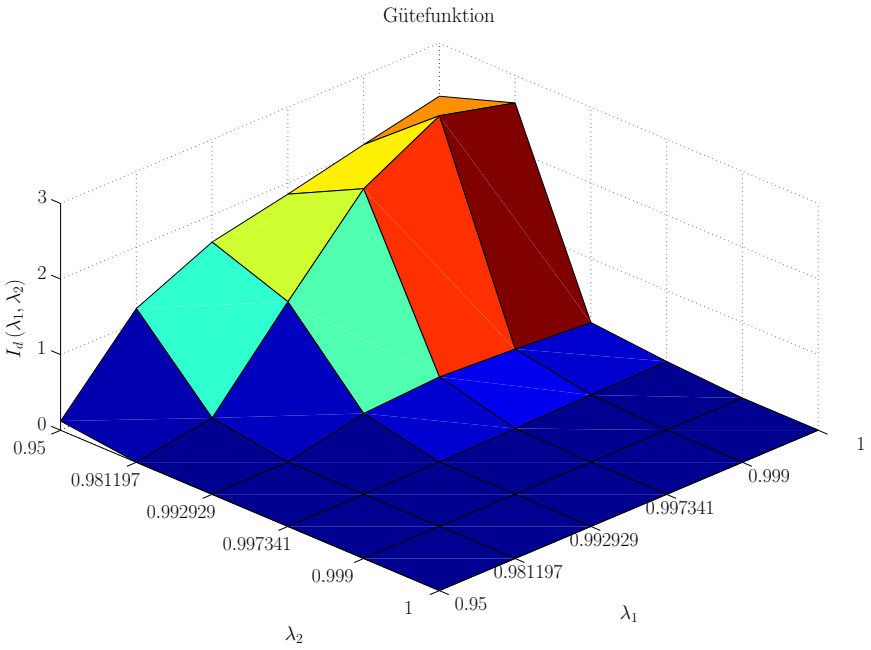

Abbildung 37: Vergleich der Kombinationen der Vergessensfaktoren (2. Beispiel)

Man beachte die im Vergleich zur Abbildung 35 veränderte Achseneinteilung. Es zeigt sich, dass hier die Wahl $\lambda_1 = 0.95$ und $\lambda_2 = 1$ die beste Totzeitschätzung mit sich bringt, da die Modellparameter zeitlich variieren und die Totzeit konstant ist. Die umgekehrte Wahl $\lambda_1 = 1$ und $\lambda_2 = 0.95$ liefert dagegen ein bedeutend schlechteres Ergebnis der Totzeitschätzung. Dies zeigt sich auch, wenn man genau diese beiden Kombinationen der Vergessensfaktoren nochmals explizit betrachtet. In Abbildung 38 ist zu erkennen, dass die wahre Totzeit im ersten Fall ($\lambda_1 = 0.95$ und $\lambda_2 = 1$) relativ zügig korrekt identifiziert wird, gegenteiliges zeigt sich für den zweiten Fall ($\lambda_1 = 1$ und $\lambda_2 = 0.95$). Hier geht die korrekte Totzeitinformation immer wieder verloren:

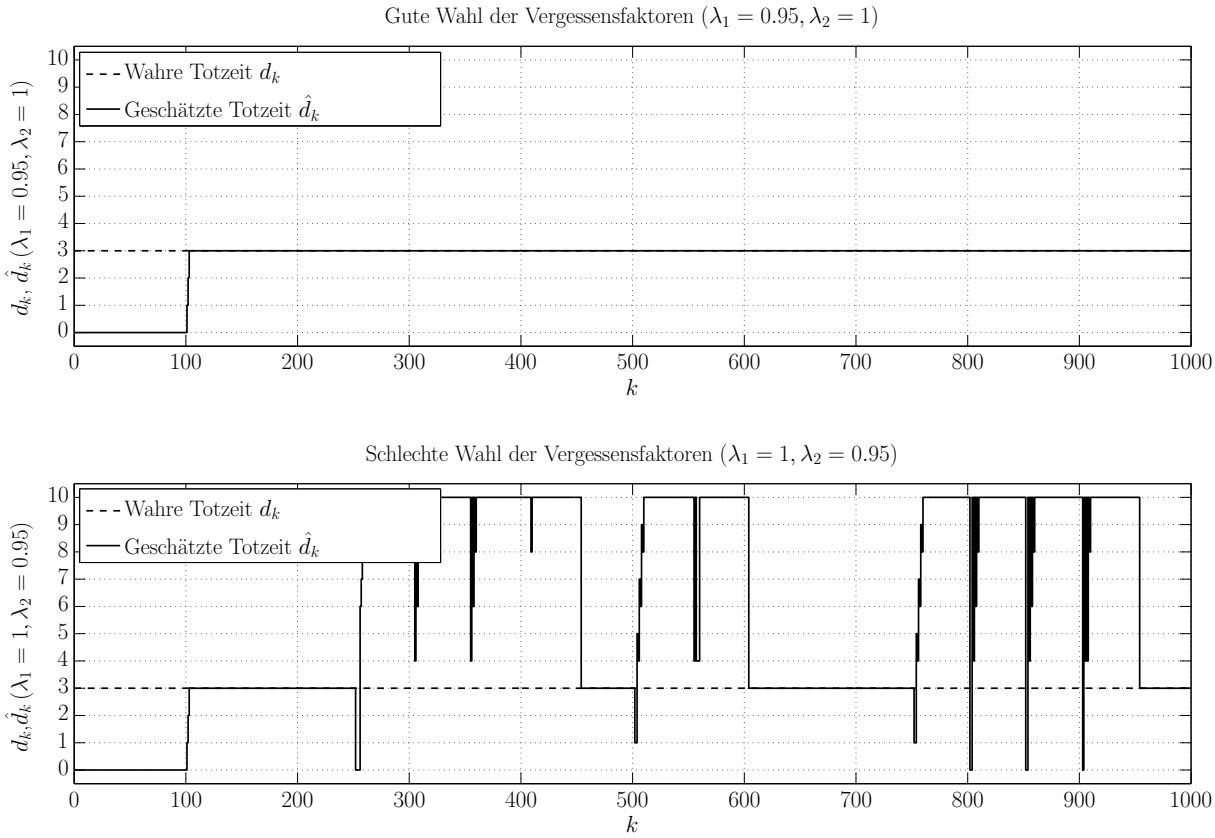

Abbildung 38: Gute und schlechte Wahl der Vergessensfaktoren (2. Beispiel)

Anhand der oben angeführten Beispiele lässt sich schlussfolgern, dass eine geeignete Wahl der Vergessensfaktoren für die Qualität der Totzeitschätzung von immenser Bedeutung ist. Außerdem lässt sich ableiten, dass es mit Sicherheit keine Kombination der beiden Vergessensfaktoren gibt, die für alle möglichen Anwendungsfälle einsetzbar ist. Wie schon angedeutet, ist Vorwissen für eine plausible Wahl von λ_1 und λ_2 zwingend notwendig. Genau dieses Vorwissen ist jedoch nur selten gegeben – oft trifft man auf Systeme, bei welchen nicht bekannt ist, ob die Modellparameter und die Totzeit zeitlich variieren oder konstant sind. In diesen Fällen stellt sich zwangsläufig die Frage, wie die Vergessensfaktoren korrekt eingestellt werden können, um eine gut funktionierende Totzeitschätzung zu gewährleisten.

Abhilfe schafft hier jedoch ein Blick auf den Ablauf des Elnaggar-Algorithmus (D.3.34). Hier wird in jedem Zeitschritt k für alle möglichen Totzeiten eine Gütefunktion $J_k(d)$ ausgewertet. Diese berechnet sich aus der Summe des mit dem Vergessensfaktor λ_2 gewichteten Gütefunktionswertes des letzten Zeitschrittes $k-1$ und des quadrierten Schätzfehlers des Modells unter Annahme der entsprechenden Totzeit. Diejenige angenommene Totzeit, zu der die minimale Gütefunktion $J_k(d)$ gehört, bestimmt dann die aktuell geschätzte Totzeit $\hat{d}_k$. Dieser Ansatz lässt vermuten, dass eine ganz ähnliche Herangehensweise auch bei der Auswahl der geeignetsten Kombination der Vergessensfaktoren λ_1 und λ_2 sinnvoll sein könnte. Hierfür definiert man

eine neue Gütefunktion $I_y(\lambda_1, \lambda_2)$. Im Vergleich zur Gütefunktion (F.2.5) wird hier nicht der Totzeitfehler $e_{d,k}(\lambda_1, \lambda_2)$ in jedem Zeitschritt quadriert, sondern der Schätzfehler $e_{y,k}(\lambda_1, \lambda_2)$. Dieser Schätzfehler ist derjenige Fehler, den das geschätzte Modell des mit den Vergessensfaktoren λ_1 und λ_2 parametrierten Elnaggar-Algorithmus zum Zeitschritt k macht. Zudem erwies es sich als sinnvoll, mit der Auswertung der genannten Gütefunktion bis zum Zeitschritt $K_1 + K_2$ zu warten. Dies hat den Grund, dass damit ein möglicher Einschwingvorgang der geschätzten Totzeit nicht in die Gütefunktionen mit einfließt.

$$\begin{aligned} I_y\left(\lambda_1, \lambda_2\right) &= \sum_{k=K_1+K_2+1}^{K_1+K_2+K_3} e_{y,k}^2\left(\lambda_1, \lambda_2\right) \\ &= \sum_{k=K_1+K_2+1}^{K_1+K_2+K_3} \left(y_k - \hat{y}_k\left(\lambda_1, \lambda_2\right)\right)^2 \end{aligned} \tag{F.2.7}$$

Die Idee besteht nun darin, dass man die zu allen $6^2 = 36$ möglichen Vergessensfaktor-Kombinationen gehörigen Elnaggar-Algorithmen parallel startet und die Gütefunktionen nach (F.2.7) in jedem Zeitschritt auswertet. Zum Zeitpunkt $k = K_1 + K_2 + K_3$ wählt man dann diejenige Kombination aus λ_1 und λ_2 aus, für welche die Gütefunktion $I_y(\lambda_1, \lambda_2)$ den kleinsten Wert aufweist und führt die Totzeitschätzung mit ebendieser Kombination durch. Die Leistungsfähigkeit dieses Ansatzes zeigt sich bei Auswertung der Gütefunktionen (F.2.7) für die beiden in diesem Kapitel angeführten Beispiele. Für das erste Beispiel (konstante Modellparameter, zeitlich variierende Totzeit) ergibt sich mit $K_1 = 100$, $K_2 = 0$ und $K_3 = 899$ folgendes Bild:

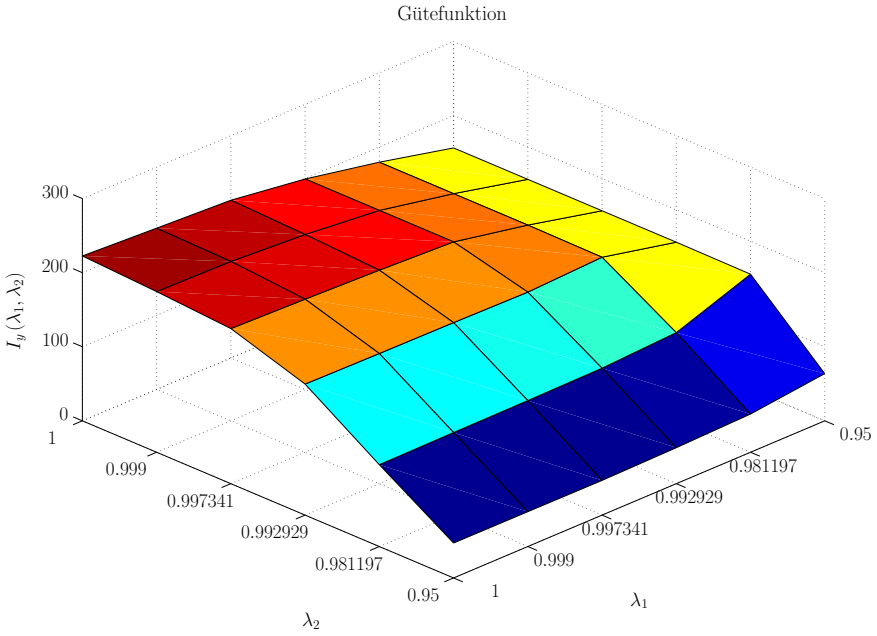

Abbildung 39: Gütefunktion zur Auswahl der Vergessensfaktoren (1. Beispiel)

Vergleicht man dieses Bild mit Abbildung 35, so fällt auf, dass sich diese stark ähneln. Auch die Gütefunktion $I_y(\lambda_1, \lambda_2)$ nimmt für $\lambda_2 = 0.95$ und eher größere λ_1 ihre kleinsten Werte an. Die Vermutung, dass sich anhand der mit Hilfe der Schätzfehler gebildeten Gütefunktionen $I_y(\lambda_1, \lambda_2)$ eine Aussage über eine geeignete Kombination von Vergessensfaktoren machen lässt, trifft also für dieses Beispiel absolut zu.

Abbildung 40 zeigt das entsprechende Bild für das zweite Beispiel (zeitlich variierende Modellparameter, konstante Totzeit):

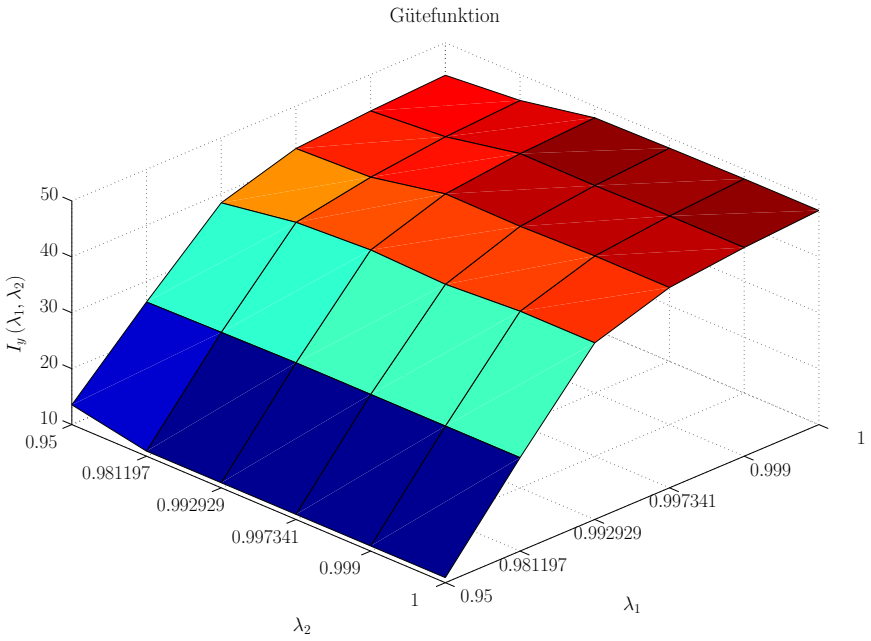

Abbildung 40: Gütefunktion zur Auswahl der Vergessensfaktoren (2. Beispiel)

Hier ist die Ähnlichkeit zum entsprechenden Bild (Abbildung 37) mit den ausgewerteten Gütefunktionen $I_d(\lambda_1, \lambda_2)$ zwar nicht ganz so stark, jedoch führt die Herangehensweise auch hier zum Erfolg. Die Gütefunktionen $I_y(\lambda_1, \lambda_2)$ sind genau für solche Kombinationen der Vergessensfaktoren klein, für die auch die Totzeitschätzung entsprechend $I_d(\lambda_1, \lambda_2)$ gut funktioniert.

Es konnte also eine Methodik entwickelt werden, mit welcher es möglich ist, eine geeignete Wahl der beiden Vergessensfaktoren λ_1 und λ_2 auch ohne Vorwissen über das zeitliche Verhalten der Modellparameter bzw. der Totzeit (konstant oder zeitlich variabel) zu treffen. Um diese Erweiterung des Elnaggar-Algorithmus zur Optimierung der Vergessensfaktoren auch on-line, also während des laufenden Betriebes, einsetzen zu können, benötigt man jedoch noch eine rekursive Ausführung der Beziehung (F.2.7). Diese sieht wie folgt aus:

$$
\begin{aligned}
I_{y,k}\left(\lambda_1,\lambda_2\right) &= I_{y,k-1}\left(\lambda_1,\lambda_2\right)+\left(y_k-\hat{y}_k\left(\lambda_1,\lambda_2\right)\right)^2 \\
&= I_{y,k-1}\left(\lambda_1,\lambda_2\right)+\left(y_k-\hat{\phi}_k\left(\lambda_1,\lambda_2\right)\hat{\boldsymbol{\theta}}_k\left(\lambda_1,\lambda_2\right)\right)^2
\end{aligned}
\tag{F.2.8}
$$

Der Ablauf des adaptierten Elnaggar-Algorithmus bei Einsatz der Optimierung der Vergessensfaktoren ist dann der folgende:

- Paralleler Start der RLS-Schätzung (Modellparameter) für alle möglichen Kombinationen der Vergessensfaktoren λ_1 und λ_2 zum Zeitpunkt $k=0$
- Paralleler Start der Totzeitschätzung zum Zeitpunkt $k=K_1+1$ (λ_2 hat erst ab hier einen Einfluss)
- Auswertung der Gütefunktionen $I_{y,k}(\lambda_1,\lambda_2)$ ab dem Zeitschritt $k=K_1+K_2+1$
- Zum Zeitschritt $k=K_1+K_2+K_3$ Auswahl der geeignetsten Vergessensfaktoren $\lambda_{1,\mathrm{opt}}$ und $\lambda_{2,\mathrm{opt}}$ anhand der kleinsten Gütefunktion $I_{y,k}(\lambda_{1,\mathrm{opt}},\lambda_{2,\mathrm{opt}})$
- Ab dem Zeitschritt $k=K_1+K_2+K_3+1$ standardmäßige Fortführung des Elnaggar-Algorithmus mit Vergessensfaktoren $\lambda_{1,\mathrm{opt}}$ und $\lambda_{2,\mathrm{opt}}$ und Abbruch der anderen, bis hierher parallel laufenden Algorithmen

Anzumerken bleibt noch, dass die erläuterte Optimierung der Vergessensfaktoren im Anwendungsfall eines durch ein Koppelelement verbundenen, geregelten Systems (Abbildung 29) sowohl auf der Streckenseite ($y_k \triangleq y_{r,k}$) als auch auf der Reglerseite ($y_k \triangleq u_{r,k}$) angewandt werden kann.

3 Direktionales Vergessen zur Verringerung der Rauschsensibilität

Ein Problem, dass bei Verwendung des RLS-Algorithmus mit exponentiellem Vergessen immer wieder auftritt, ist die hohe Rauschsensibilität. Bei Auftreten von Mess- bzw. Ausgangsrauschen und gleichzeitiger unzureichender Anregung kann es passieren, dass zuvor schon gut geschätzte Modellparameter vom Algorithmus „verlernt“ werden. Da der in dieser Arbeit adaptierte Elnaggar-Algorithmus auf dem RLS-Algorithmus basiert, liegt ein Problem auf der Hand: Weichen die geschätzten Modellparameter im Schätzvektor $\hat{\boldsymbol{\theta}}_k$ zu stark von den wahren Parametern $\boldsymbol{\theta}_k$ ab, dann können die Gütefunktionen $J_k(d)$ kein aussagekräftiges Bild zur Ermittlung der geschätzten Totzeit d_k abgeben. Ein ähnlicher Effekt wurde schon in Kapitel 1 bei der Berücksichtigung von Einschwingvorgängen besprochen. Auch in diesem Fall kann es

passieren, dass die Totzeitschätzung durch falsch geschätzte Modellparameter fehlschlägt.

Bevor ein Lösungsansatz für das beschriebene Problem erklärt wird, soll zunächst genauer auf die Ursache eingegangen werden. Wie beschrieben kann es passieren, dass die Modellparameter bei Ausgangsrauschen und unzureichender Anregung verlernt werden. Dieses Phänomen liegt in einem sogenannten Blow-Up, also einem Aufblähen der Kovarianzmatrix $\mathbf{P}_k$ begründet [29]. Um diesen Effekt genauer betrachten zu können, stellt man die Kovarianzmatrix grafisch durch die Beziehung $\mathbf{x}^T\mathbf{P}_k^{-1}\mathbf{x} = \mathrm{C}$ dar. Der Vektor $\mathbf{x}$ hat hierbei die zur Kovarianzmatrix (Dimension $(m+n+1)\times(m+n+1)$) passende Dimension $m+n+1$. Die genannte Beziehung entspricht bei einer Konstante C einem Ellipsoiden der Dimension $m+n+1$. Der Einfachheit halber wird in weiterer Folge eine Dimension von $m+n+1=2$ angenommen und es wird $C=1$ gesetzt. Für die Beziehung $\mathbf{x}^T\mathbf{P}_k^{-1}\mathbf{x} = 1$ ergibt sich damit eine Ellipse in der x_1-x_2-Ebene. Man nimmt nun an, dass die Kovarianzmatrix zum Zeitschritt $k-1$ folgende Einträge besitzt:

$$\mathbf{P}_{k-1} = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \tag{F.3.1}$$

Für die Informationsmatrix $\mathbf{P}_{k-1}^{-1}$ gilt damit:

$$\mathbf{P}_{k-1}^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1/3 \end{pmatrix} \tag{F.3.2}$$

Im RLS-Algorithmus berechnet sich die Informationsmatrix im nächsten Zeitschritt nach Beziehung (D.3.16) durch die folgende Iterationsvorschrift:

$$\mathbf{P}_k^{-1} = \lambda\mathbf{P}_{k-1}^{-1} + \boldsymbol{\phi}_k\boldsymbol{\phi}_k^T \tag{F.3.3}$$

Zur genaueren Analyse wird Beziehung (F.3.3) aufgespalten:

$$\tilde{\mathbf{P}}_k^{-1} = \lambda\mathbf{P}_{k-1}^{-1} \tag{F.3.4}$$

$$\mathbf{P}_k^{-1} = \tilde{\mathbf{P}}_k^{-1} + \boldsymbol{\phi}_k\boldsymbol{\phi}_k^T \tag{F.3.5}$$

Um den Effekt des Blow-Ups der Kovarianzmatrix besser zeigen zu können, wird ein sehr kleiner Vergessensfaktor von $\lambda = 0.5$ gewählt. Abbildung 41 zeigt die Entwicklung der Kovarianzmatrix (F.3.2) durch das Vergessens-Update, also durch (F.3.4):

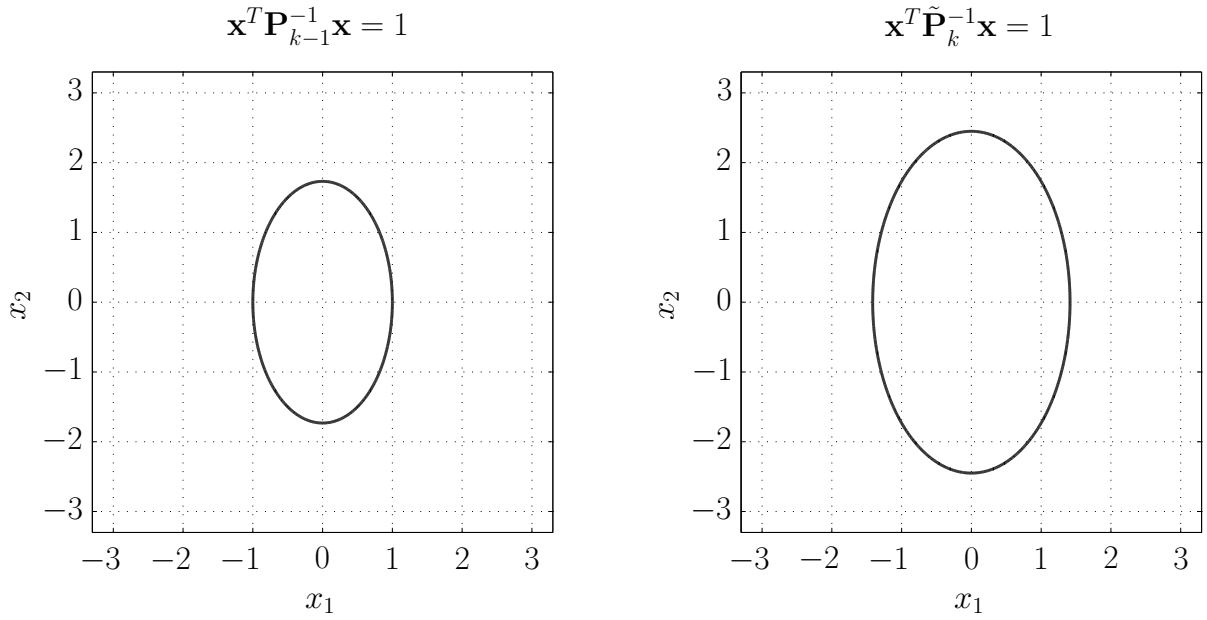

Abbildung 41: Vergessens-Update exponentielles Vergessen

Man erkennt, dass sich die Kovarianzmatrix durch das exponentielle Vergessen in allen Richtungen gleichmäßig vergrößert. Als nächstes werden vier verschiedene Messvektoren $\boldsymbol{\phi}_k$ angenommen:

$$\boldsymbol{\phi}_{1,k} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \boldsymbol{\phi}_{2,k} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \boldsymbol{\phi}_{3,k} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}; \boldsymbol{\phi}_{4,k} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{F.3.6}$$

Anhand Beziehung (F.3.5) lässt die Entwicklung der Kovarianzmatrix für diese vier Messvektoren darstellen (Messvektor-Update):

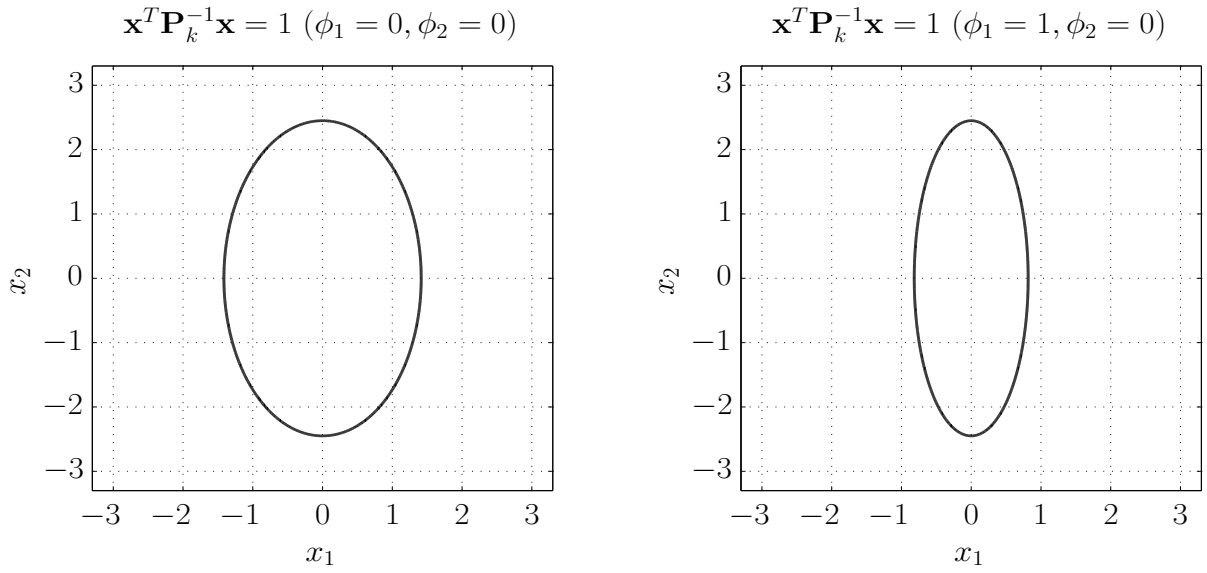

Abbildung 42: Messvektor-Update exponentielles Vergessen (1)

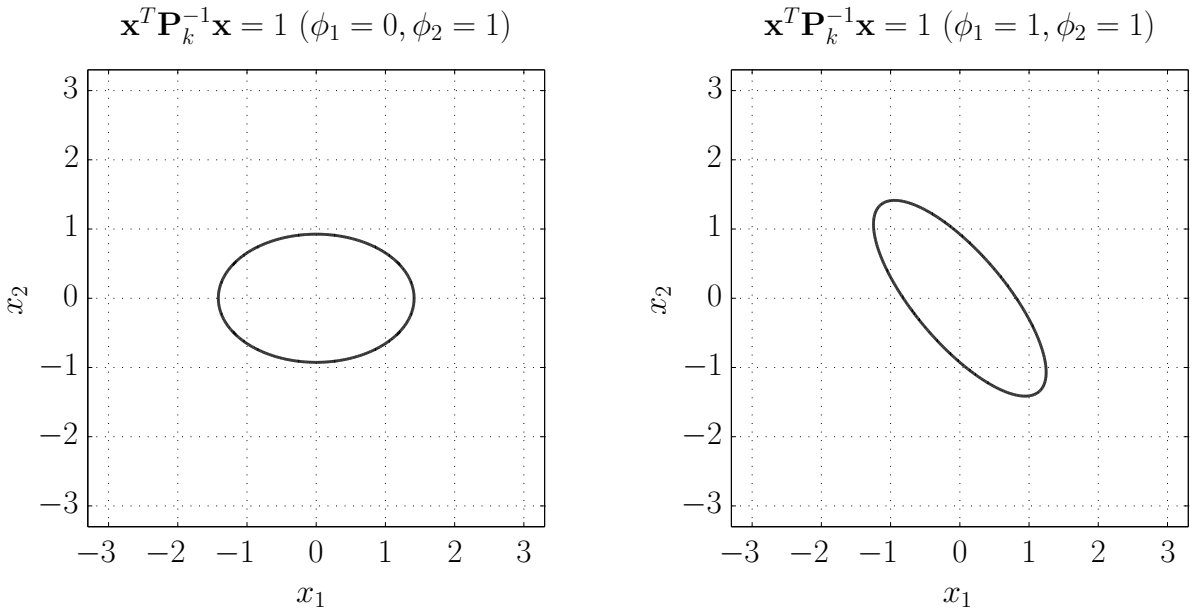

Abbildung 43: Messvektor-Update exponentielles Vergessen (2)

Man erkennt, dass sich die Kovarianzmatrix $\mathbf{P}_k$ im Vergleich zu $\tilde{\mathbf{P}}_k$ durch den Messvektor $\boldsymbol{\phi}_{1,k}$ wie zu erwarten nicht ändert. Bei den Messvektoren $\boldsymbol{\phi}_{2,k}$ und $\boldsymbol{\phi}_{3,k}$ kann man eine Verengung der Ellipse in die jeweilige Anregungsrichtung beobachten. Bei $\boldsymbol{\phi}_{2,k}$ verengt sich die Kovarianzmatrix also beispielsweise in horizontaler Richtung, da in dieser Richtung eine Anregung stattfindet. Eine Verengung der Kovarianzmatrix in dieser Richtung führt dazu, dass der erste Eintrag im Schätzvektor $\hat{\boldsymbol{\theta}}_k$ gefestigt wird und dadurch weniger sensibel auf Rauschen ist. Analog gelten diese Betrachtungen für den Messvektor $\boldsymbol{\phi}_{3,k}$. Beim Messvektor $\boldsymbol{\phi}_{4,k}$ verengt sich die Ellipse hingegen in beiden Richtungen, da auch beide Richtungen angeregt werden. Gemäß der identischen Anregung in beide Richtungen ergibt sich eine diagonale Verengung.

Zusammenfassend kann man sagen, dass sich die Kovarianzmatrix beim Messvektor-Update (F.3.5) immer in Richtung der Anregung verengt. In der orthogonalen Richtung findet jedoch keine Verengung der Ellipse statt. Geht man nun davon aus, dass die Anregung konstant in einer Richtung stattfindet, führt dies zum angesprochenen Kovarianzmatrix-Blowup. Dieser rührt daher, dass sich die Kovarianzmatrix durch das Vergessens-Update (F.3.4) wie erläutert in allen Richtungen gleichmäßig vergrößert. In der zur Anregungsrichtung orthogonalen Richtung wächst die Kovarianzmatrix dann fortwährend weiter, solange keine entsprechende Anregung stattfindet. Kommt es zum häufigen Fall von Mess- bzw. Ausgangsrauschen, kann jedoch genau in dieser Richtung dennoch eine geringfügige Anregung auftreten, auch wenn diese unerwünscht ist. Durch die aufgeblähte Kovarianzmatrix werden dadurch die entsprechenden Einträge im Schätzvektor $\hat{\boldsymbol{\theta}}$ beeinflusst, obwohl durch den Messvektor eigentlich kein Informationsgewinn bezüglich der entsprechenden Parameter möglich

ist. Dies kann wie angedeutet zu einem Verlernen der Modellparameter führen. Hierzu folgt ein kurzes Beispiel, dass die folgende, bekannte Strecke behandelt:

$$P\Big(\, z \,\Big) = \frac{0.5}{z - 0.8} \tag{F.3.7}$$

Als Anregung u_k dient ein Signal der Länge $N = 1000$, dass zunächst 400 Samples lang aus einem vergleichsweise hochfrequenten Rechteck besteht, anschließend verharrt das Signal auf dem Wert 1. Außerdem tritt am Ausgang y_k normalverteiltes Rauschen mit $\sigma_{_y} = 0.1$ auf. Abbildung 44 zeigt die Signale u_k und y_k:

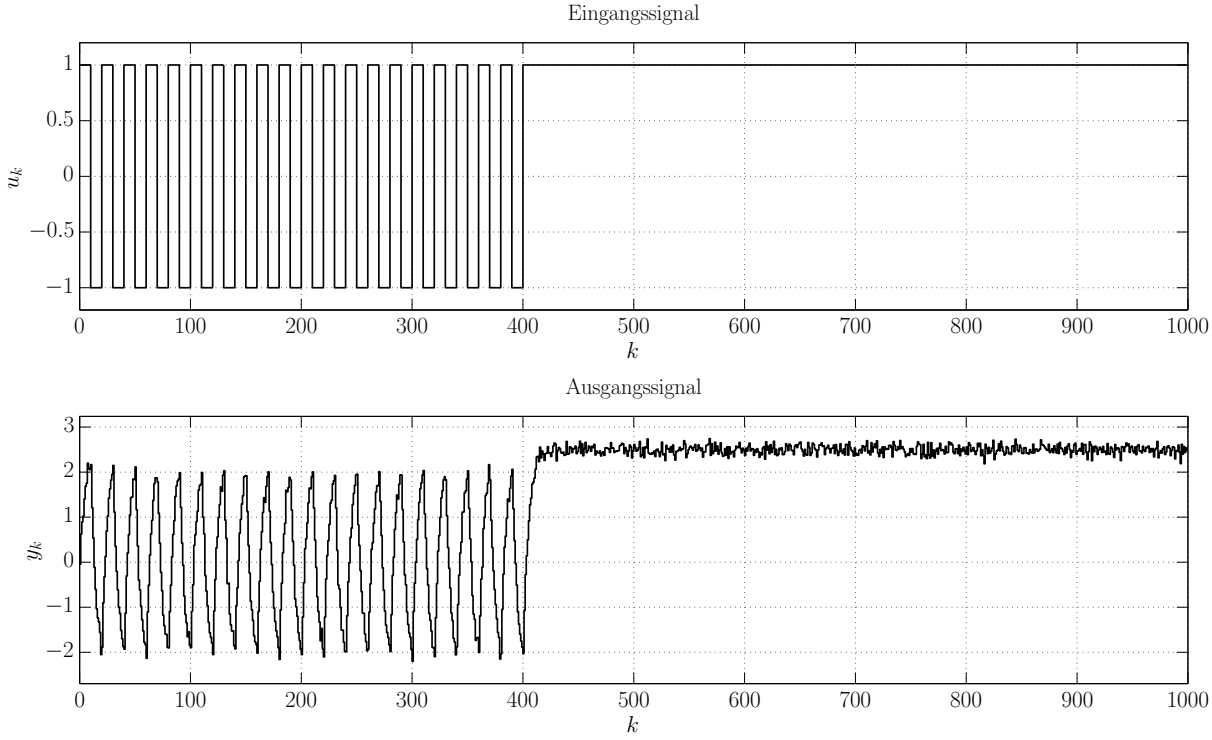

Abbildung 44: Beispiel Kovarianzmatrix-Blow-Up - Ein- und Ausgangssignal

Die Modellparameter sollen nun anhand des RLS-Algorithmus (D.3.17) mit exponentiellem Vergessen geschätzt werden, wobei für die Initialisierung $b_{0,0} = a_{0,0} = 0$ und für $\mathbf{P}_0$ die 2×2-Einheitsmatrix gewählt wurden. Abbildung 45 zeigt die geschätzten Parameter bei Verwendung des Vergessensfaktors $\lambda = 0.98$ für den RLS-Algorithmus:

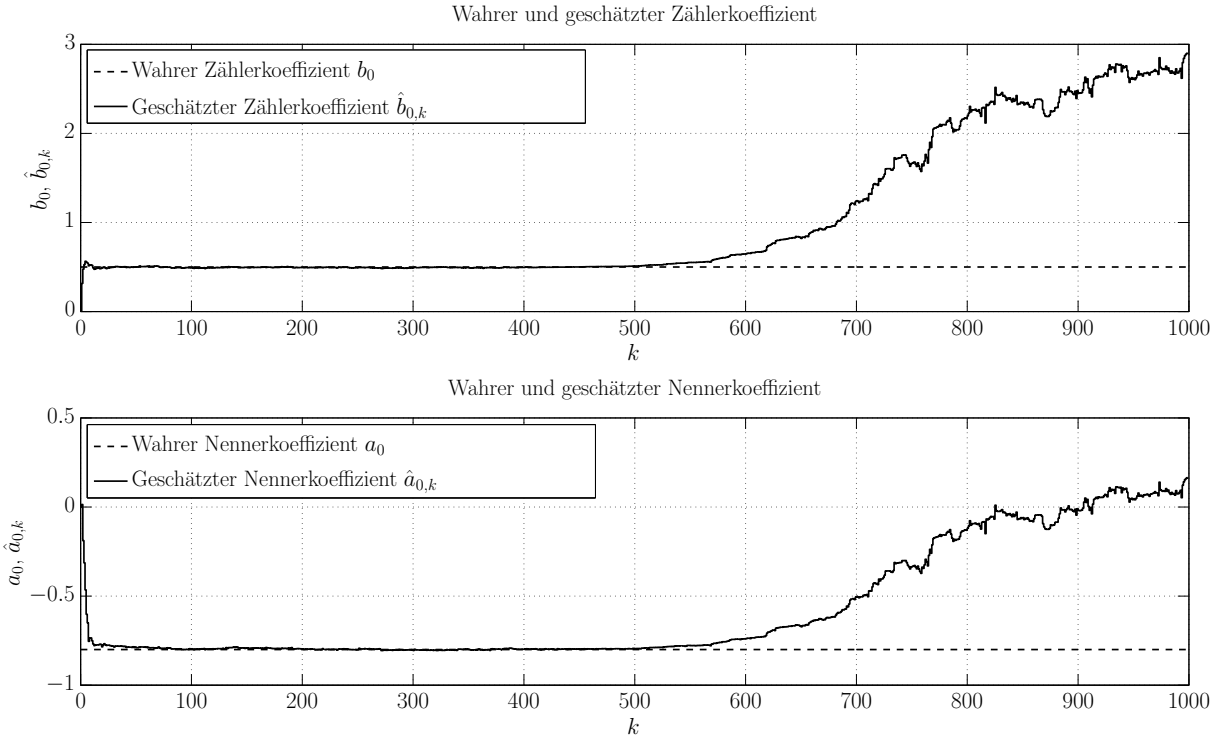

Abbildung 45: Beispiel Kovarianzmatrix-Blow-Up - Geschätzte Modellparameter

Man kann beobachten, dass die Modellparameter durch die Rechteck-Anregung zu Beginn sehr zügig richtig identifiziert werden. Sobald jedoch keine Anregung mehr auftritt kommt es zum erwähnten Kovarianzmatrix-Blow-Up und die zuvor schon richtig geschätzten Modellparameter werden unter Einfluss des Ausgangsrauschens wieder verlernt.

Um genau dieses Phänomen zu verhindern, wurde eine Vielzahl alternativer Vergessensstrategien zur Eindämmung des Blow-Up-Effekts entwickelt - Kraus [30] liefert hier einen ersten Überblick. Milek [29] schlägt einen Algorithmus mit stabilisiertem exponentiellen bzw. stabilisiertem linearen Vergessen vor. Bei diesen beiden Algorithmen werden die Eigenwerte der Kovarianzmatrix $\mathbf{P}_k$ von oben bzw. die Eigenwerte der Informationsmatrix $\mathbf{P}_k^{-1}$ von unten beschränkt. Dadurch kann ein Blow-Up der Kovarianzmatrix zwar verhindert werden, jedoch wird die Kovarianzmatrix bei unzureichender Anregung in den entsprechenden Richtungen durch das exponentielle Vergessen trotzdem unnötig vergrößert. Von Kulhávy & Kárny [31] und in ähnlicher Form von Hägglund [32] wurde erstmals die Methode des *direktionalen Vergessens* eingeführt. Beim direktionalen Vergessen wird das exponentielle Vergessen nur auf den kleinsten Unterraum derjenigen zu schätzenden Parameter angewendet, über den der Messvektor neue Informationen enthält. Die Grundlage hierfür liegt in der Bayesschen Statistik. Das direktionale Vergessen wurde zudem später von Kulhávy [33] durch die Einführung eines adaptiven Vergessensfaktors weiterentwickelt. Einzelheiten hierzu sind auch in [34] zu finden. Auch Bittanti, Bolzern & Campi [35] greifen das direktionale Vergessen auf, stellen aber fest, dass der RLS-Algorithmus mit direk-

tionalem Vergessen zwar einfache Konvergenz, jedoch nicht exponentielle Konvergenz aufweist. Daher wird versucht, die Kovarianzmatrix $\mathbf{P}_k$ in jedem Zeitschritt durch hinzufügen einer mit einem kleinen positiven Faktor $0 < \nu \ll 1$ multiplizierten Einheitsmatrix künstlich zu vergrößern. Dadurch wird zwar wieder exponentielle Konvergenz des Algorithmus erreicht, jedoch steuert man den errungenen Vorteilen des direktionalen Vergessens entgegen. Durch die Addition mit der „kleinen“ Einheitsmatrix wird die Kovarianzmatrix wiederum in alle Richtungen vergrößert und nicht nur in die Richtung der Anregung. Cao & Schwartz [36] beschäftigen sich ebenfalls mit dem direktionalen Vergessen in seiner Grundform, weisen jedoch darauf hin, dass die Gefahr von unbeschränkten Eigenwerten der Informationsmatrix $\mathbf{P}_k^{-1}$ besteht. Dies führt zum Verlust der Fähigkeit, zeitvarianten Modellparametern zu folgen. Aus diesem Grund wird in [36] eine verbesserte Variante des direktionalen Vergessens eingeführt, welche die oben angeführten Nachteile nicht aufweist. Hierfür wird die Informationsmatrix $\mathbf{P}_{k-1}^{-1}$ in zwei Teile $\mathbf{P}_{1,k-1}^{-1}$ und $\mathbf{P}_{2,k-1}^{-1}$ zerlegt, wobei $\mathbf{P}_{1,k-1}^{-1}$ den vom Messvektor $\boldsymbol{\phi}_k$ nicht angeregten Unterraum von $\mathbf{P}_{k-1}^{-1}$ repräsentiert, während $\mathbf{P}_{2,k-1}^{-1}$ für den angeregten Teil steht. Es gilt:

$$\begin{aligned} \mathbf{P}_{1,k-1}^{-1}\boldsymbol{\phi}_k &= 0 \\ \mathbf{P}_{2,k-1}^{-1}\boldsymbol{\phi}_k &= \mathbf{P}_{k-1}^{-1}\boldsymbol{\phi}_k \end{aligned} \tag{F.3.8}$$

Im weiteren Verlauf ergibt sich durch das Matrix-Inversion-Lemma [7]:

$$\mathbf{P}_{2,k-1}^{-1} = \begin{cases} \dfrac{\left(\mathbf{P}_{k-1}^{-1}\boldsymbol{\phi}_k\right)\left(\mathbf{P}_{k-1}^{-1}\boldsymbol{\phi}_k\right)^T}{\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}^{-1}\boldsymbol{\phi}_k} & \text{für } \left|\,\boldsymbol{\phi}_k\,\right| > \varepsilon \\ \mathbf{0} & \text{für } \left|\,\boldsymbol{\phi}_k\,\right| \leq \varepsilon \end{cases} \tag{F.3.9}$$

Die Überprüfung $|\,\boldsymbol{\phi}_k\,| > \varepsilon$ ist dabei elementweise gemeint. Sie dient dazu, im ersten Schritt des Algorithmus eine Division durch Null zu vermeiden ($0 < \varepsilon \ll 1$). Aus $\mathbf{P}_{k-1}^{-1} = \mathbf{P}_{1,k-1}^{-1} + \mathbf{P}_{2,k-1}^{-1}$ erhält man damit:

$$\mathbf{P}_{1,k-1}^{-1} = \begin{cases} \mathbf{P}_{k-1}^{-1} - \dfrac{\left(\mathbf{P}_{k-1}^{-1}\boldsymbol{\phi}_k\right)\left(\mathbf{P}_{k-1}^{-1}\boldsymbol{\phi}_k\right)^T}{\boldsymbol{\phi}_k^T\mathbf{P}_{k-1}^{-1}\boldsymbol{\phi}_k} & \text{für } \left|\,\boldsymbol{\phi}_k\,\right| > \varepsilon \\ \mathbf{P}_{k-1}^{-1} & \text{für } \left|\,\boldsymbol{\phi}_k\,\right| \leq \varepsilon \end{cases} \tag{F.3.10}$$

Die Idee besteht nun darin, das exponentielle Vergessen nur auf den angeregten Unterraum der Informationsmatrix $\mathbf{P}_{k-1}^{-1}$, also auf die Matrix $\mathbf{P}_{2,k-1}^{-1}$ anzuwenden. Die Beziehung (D.3.16) verändert sich damit zu:

$$\mathbf{P}_k^{-1} = \mathbf{P}_{1,k-1}^{-1} + \lambda \mathbf{P}_{2,k-1}^{-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \tag{F.3.11}$$

In (F.3.11) wird das Konzept hinter dieser Adaption des direktionalen Vergessens nochmals deutlich. $\mathbf{P}_{1,k-1}^{-1}$ repräsentiert die Information, die orthogonal zum Messvektor $\boldsymbol{\phi}_k$ steht und somit beibehalten werden soll. $\mathbf{P}_{2,k-1}^{-1}$ dagegen repräsentiert die Information, die durch den Messvektor $\boldsymbol{\phi}_k$ erneuert wird und damit (teilweise) vergessen werden kann. Nach weiteren Umformungen, die größtenteils auf Verbesserungen in Numerik und Recheneffizienz abzielen, ergibt sich schlussendlich der folgende Algorithmus zur RLS-Schätzung mittels direktionalem Vergessen:

$$\overline{\mathbf{P}}_{k-1} = \begin{cases} \mathbf{P}_{k-1} + \dfrac{1-\lambda}{\lambda} \dfrac{\boldsymbol{\phi}_k \boldsymbol{\phi}_k^T}{\boldsymbol{\phi}_k^T \mathbf{P}_{k-1}^{-1} \boldsymbol{\phi}_k} & \text{für } \left| \boldsymbol{\phi}_k \right| > \varepsilon \\ \mathbf{P}_{k-1} & \text{für } \left| \boldsymbol{\phi}_k \right| \leq \varepsilon \end{cases}$$

$$\mathbf{P}_k = \overline{\mathbf{P}}_{k-1} - \frac{\overline{\mathbf{P}}_{k-1} \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \overline{\mathbf{P}}_{k-1}}{1 + \boldsymbol{\phi}_k^T \overline{\mathbf{P}}_{k-1} \boldsymbol{\phi}_k} \tag{F.3.12}$$

$$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{P}_k \boldsymbol{\phi}_k \left(y_k - \boldsymbol{\phi}_k^T \hat{\boldsymbol{\theta}}_{k-1} \right)$$

Zur Veranschaulichung der Funktionsweise des direktionalen Vergessens wird Beziehung (F.3.11) wie schon beim exponentiellen Vergessen in zwei Teilgleichungen zerlegt:

$$\tilde{\mathbf{P}}_k^{-1} = \mathbf{P}_{1,k-1}^{-1} + \lambda \mathbf{P}_{2,k-1}^{-1} \tag{F.3.13}$$

$$\mathbf{P}_k^{-1} = \tilde{\mathbf{P}}_k^{-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \tag{F.3.14}$$

Das Vergessens-Update (F.3.13) ist nun nicht nur vom Vergessensfaktor λ, sondern wegen (F.3.9) und (F.3.10) auch vom Messvektor $\boldsymbol{\phi}_k$ abhängig. Geht man nun wieder von der Informationsmatrix $\mathbf{P}_{k-1}^{-1}$ gemäß (F.3.2) aus, dann entwickelt sich die Kovarianzmatrix mit den Messvektoren aus (F.3.6) und $\lambda = 0.5$ durch dass Vergessens-Update wie folgt:

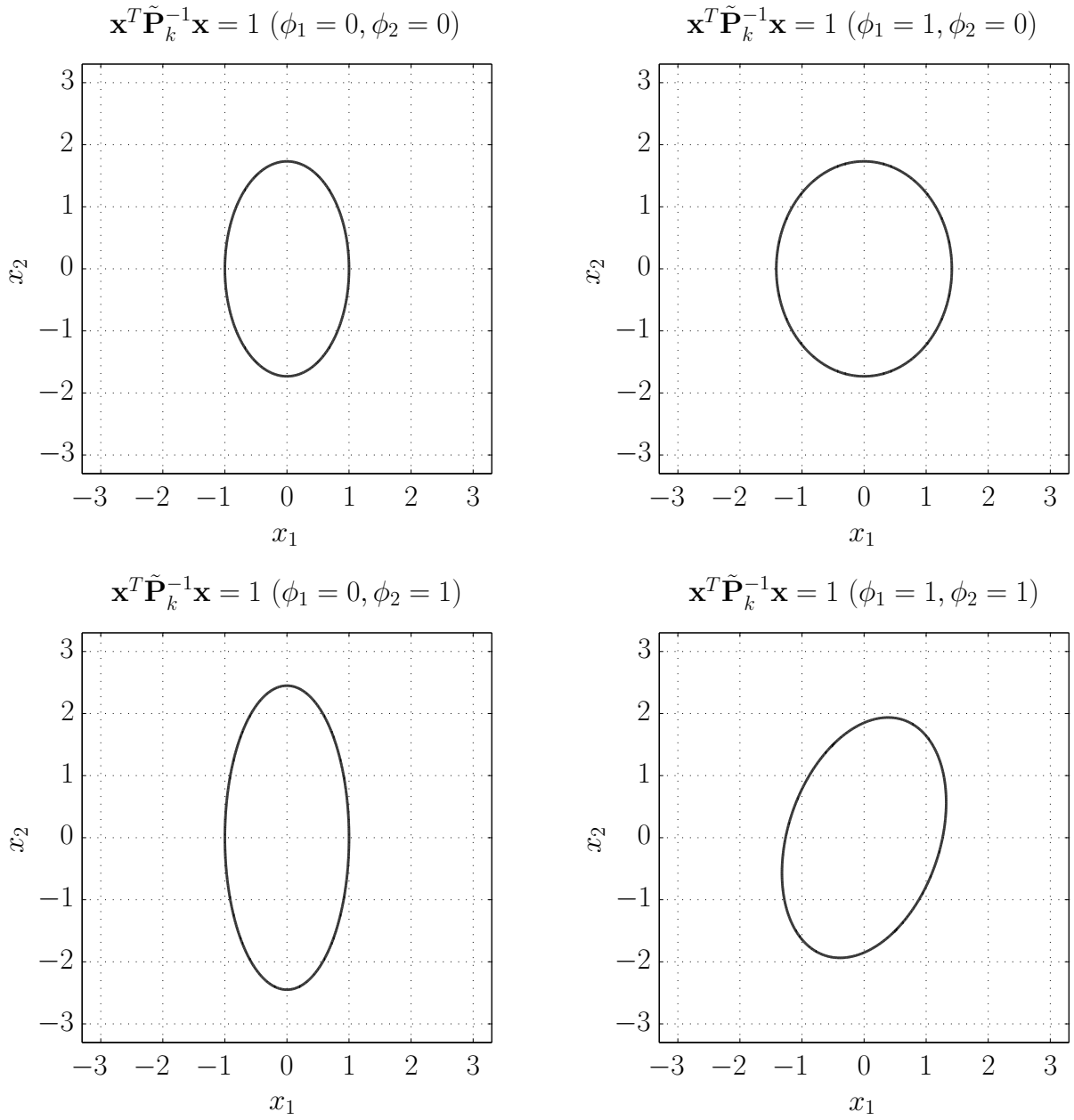

Abbildung 46: Vergessens-Update direktionales Vergessen

Man kann beobachten, dass sich die Ellipse wie zu erwarten tatsächlich nur in Richtung der Anregung vergrößert. Daher gibt es für den Messvektor $\boldsymbol{\phi}_{1,k}$ auch keinen Unterschied zwischen $\tilde{\mathbf{P}}_k^{-1}$ und $\mathbf{P}_{k-1}^{-1}$. Durch das Messvektor-Update (F.3.14) ergeben sich anschießend die folgenden Entwicklungen der Kovarianzmatrizen:

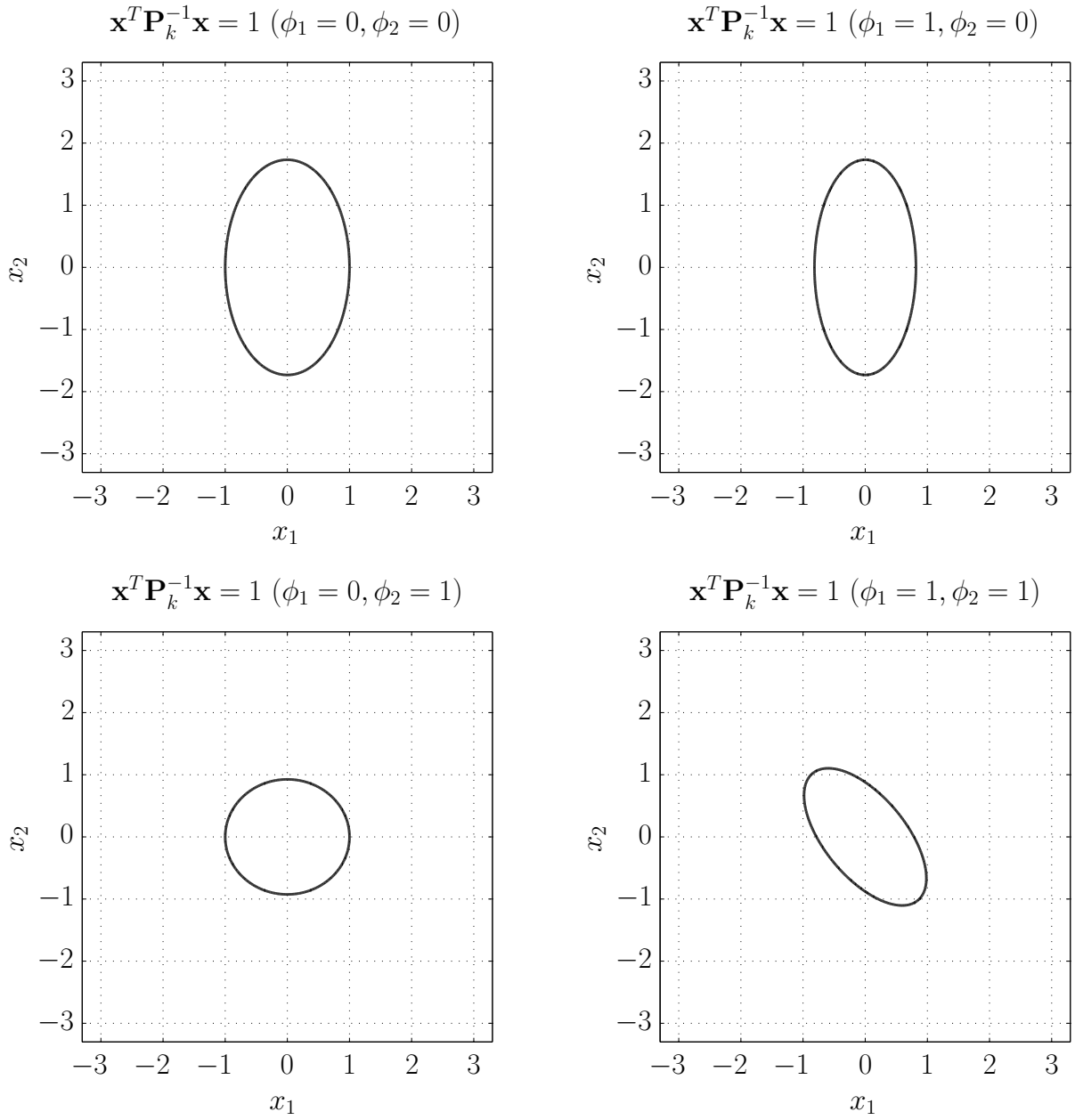

Abbildung 47: Messvektor-Update direktionales Vergessen

Durch das direktionale Vergessen in obiger Form konnte also erreicht werden, dass ein Blow-Up der Kovarianzmatrix vermieden wird und dennoch die Fähigkeit zum Tracking zeitvariabler Modellparameter in allen Richtungen erhalten bleibt. Wendet man das direktionale Vergessen ($\lambda = 0.98$) auf die Strecke (F.3.7) mit gleichem Eingangssignal und identischem Messrauschen an, dann sehen die Schätzungen der Parameter im zeitlichen Verlauf wie folgt aus:

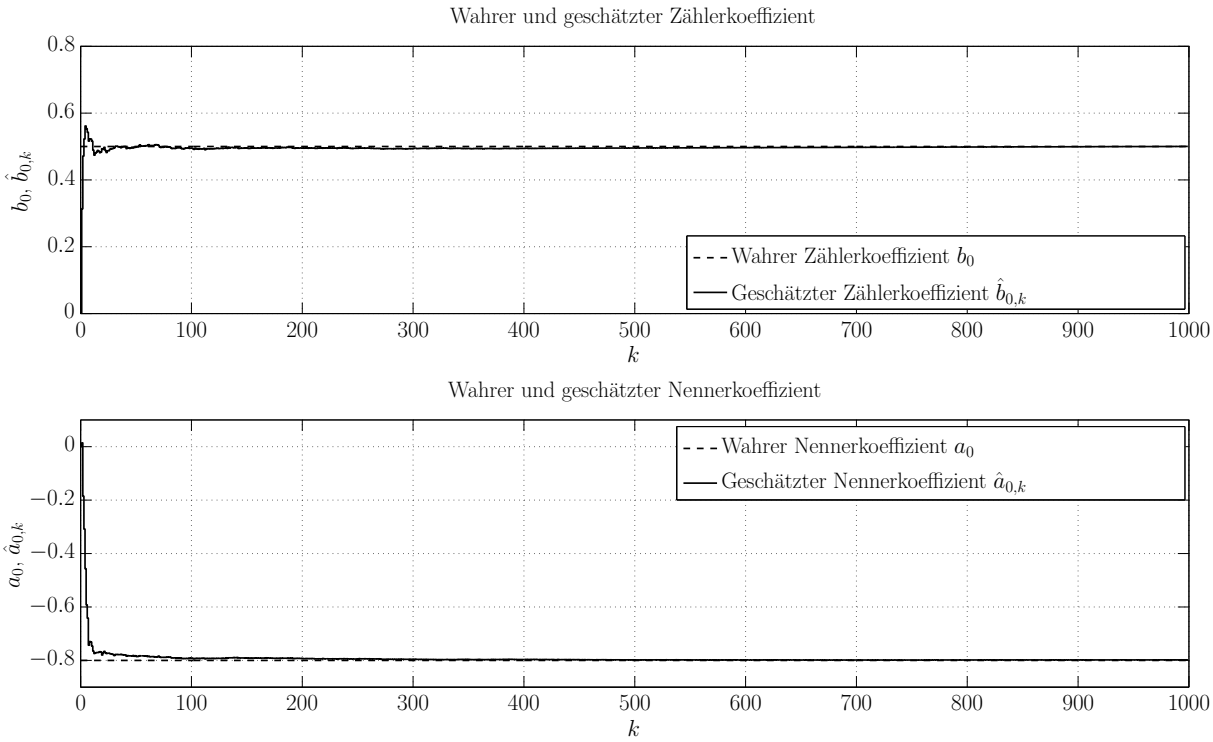

Abbildung 48: Beispiel direktionales Vergessen - Geschätzte Modellparameter

Man erkennt eine deutliche Verbesserung im Vergleich zu Abbildung 45. Hier entfernen sich die geschätzten Modellparameter auch bei unzureichender Anregung nahezu überhaupt nicht von de wahren Parametern, obwohl das selbe Ausgangsrauschen auftritt.

Um deutlich zu machen, dass das direktionale Vergessen auch für den Elnaggar-Algorithmus zur On-Line-Totzeitschätzung Vorteile mit sich bringt, wird ein weiteres Beispiel herangezogen. Die Totzeit wird dabei nur auf der Streckenseite (open-loop) geschätzt. Man betrachtet folgende Strecke:

$$P\Big(\,z\,\Big) = \frac{0.5}{z-0.8}\,z^{-d_k} \quad ; \quad d_k = \begin{cases} 3 \text{ für } 0 \leq \mathrm{k} \leq 899 \\ 6 \text{ für } 900 \leq \mathrm{k} \leq 1200 \end{cases} \tag{F.3.15}$$

Als Eingang $u_{s,k}$ dient ähnlich wie im vorangegangenen Beispiel ein Signal, dass zunächst 400 Samples lang ein Rechteck (Periodendauer 20) aufweist, anschließend 600 Samples auf dem Wert 1 verharrt. Nun folgen abschließend jedoch noch 200 Samples, während denen $u_{s,k}$ den Wert -1 annimmt. Außerdem tritt auch hier normalverteiltes Ausgangsrauschen mit Standardabweichung $\sigma_{y_r} = 0.1$ auf. Abbildung 49 zeigt die Signale $u_{s,k}$ und $y_{r,k}$:

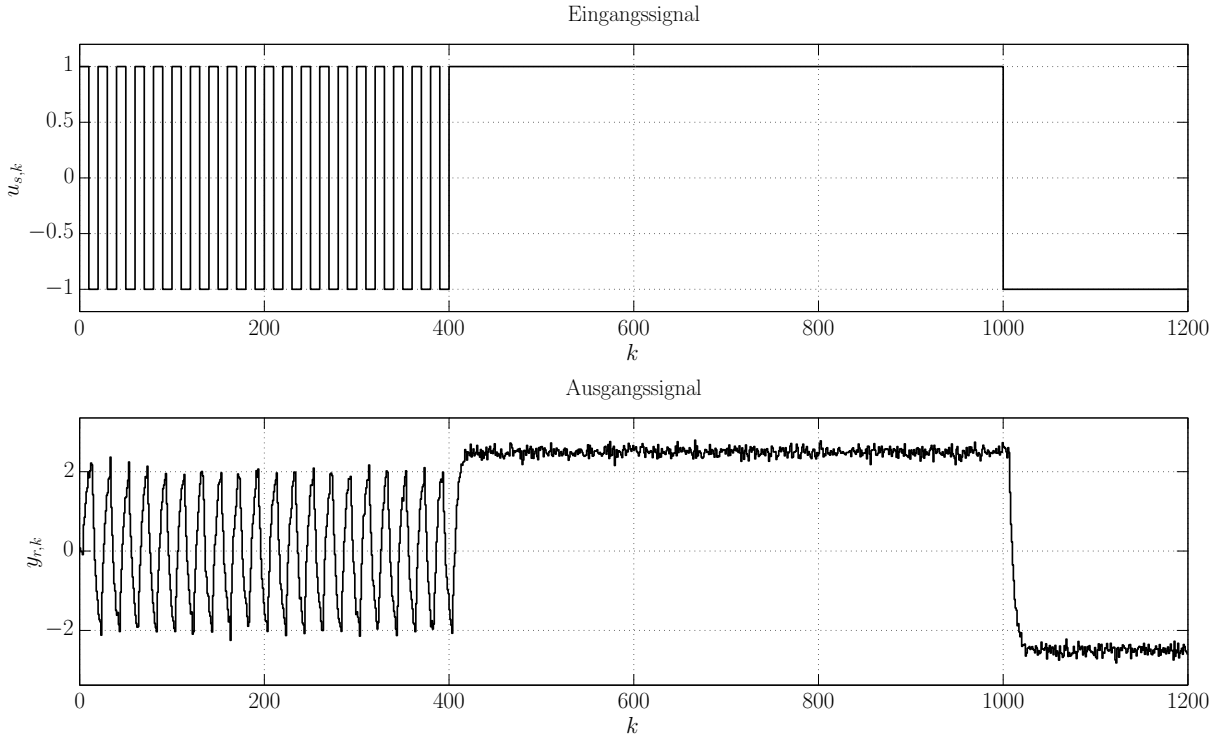

Abbildung 49: Vergleich der Vergessensstrategien - Ein- und Ausgangssignal

Abbildung 50 zeigt die geschätzten Parameter, wenn für die RLS-Schätzung im Elnaggar-Algorithmus (E.1.3) exponentielles Vergessen (durchgezogene Linie) bzw. direktionales Vergessen (Strichpunkt-Linie) verwendet wird. Dabei wurden in beiden Fällen die Vergessensfaktoren $\lambda_1 = \lambda_2 = 0.98$ gewählt:

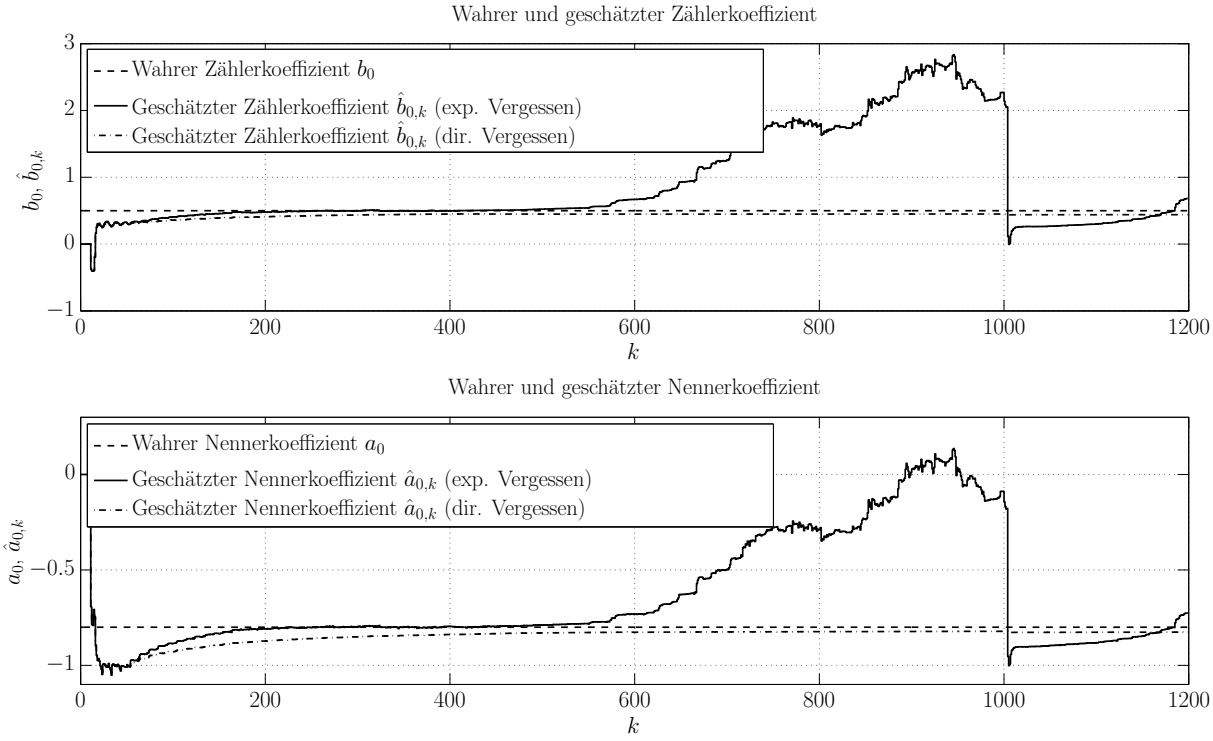

Abbildung 50: Vergleich der Vergessensstrategien - Geschätzte Modellparameter

Man erkennt, dass die Modellparameter beim exponentiellen Vergessen während der unzureichenden Anregung ($u_{s,k} = 1$) verlernt werden, wodurch es nicht möglich ist, die Änderung in der Totzeit von d_k auf $d_k = 6$ zu erkennen. Beim direktionalen Vergessen ist dies nicht der Fall. Hier weisen die geschätzten Parameter auch noch beim Sprung des Eingangssignals auf $u_{s,k} = -1$ zum Zeitpunkt $k = 1000$ sehr gute Werte auf, weshalb die Erkennung der richtigen Totzeit erfolgreich von Statten geht. Dies zeigt Abbildung 51:

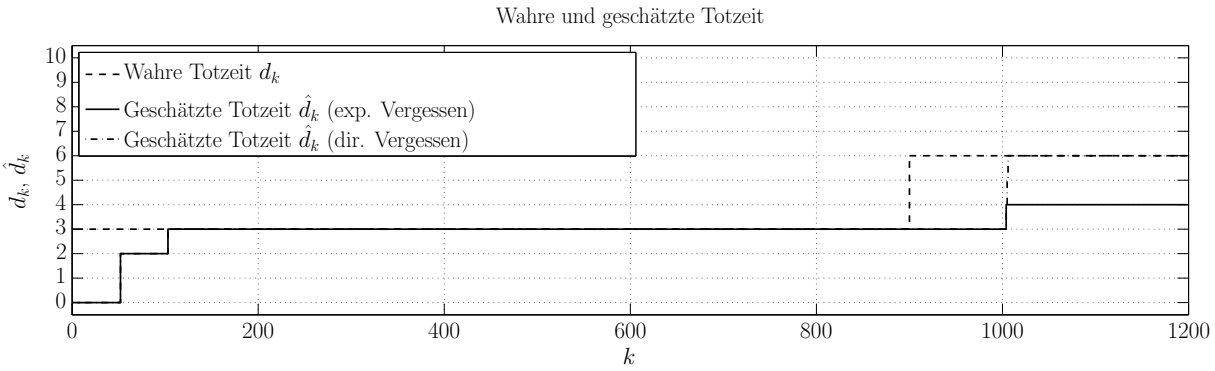

Abbildung 51: Vergleich Vergessensstrategien - Geschätzte Totzeiten

Zusammenfassend lässt sich sagen, dass durch die Einführung der Methode des direktionalen Vergessens im RLS-Algorithmus eine erhebliche Verbesserung bezüglich der Rauschsensibilität des Elnaggar-Algorithmus erzielt werden konnte. Insbesondere bei konstanten Modellparametern ist der Einsatz des direktionalen Vergessens zu empfehlen. Es muss jedoch beachtet werden, dass diese Vergessensstrategie bei schnell variierenden Modellparametern nicht immer von Vorteil ist, da die Fähigkeit des *Trackings* der Parameter beim direktionalen Vergessen grundsätzlich geringer als beim exponentiellen Vergessen ist.

4 Sprungerkennung für Arbeitspunktwechsel

Zur Verbesserung der Anwendbarkeit des Elnaggar-Algorithmus (D.3.34) wurden bereits zwei Maßnahmen eingeführt, um die Auswertung der Gütefunktionen für die möglichen Totzeiten robuster zu gestalten. Diese Gütefunktionen berechnen sich bekanntermaßen wie folgt:

$$J_k\left(d\right) = \lambda_2 J_{k-1}\left(d\right) + \left[\, y_k - \boldsymbol{\phi}_k^T\left(d\right)\hat{\boldsymbol{\theta}}_k \,\right]^2 \qquad \forall \qquad d \in \left[\, d_{\min}, d_{\max} \,\right] \tag{F.4.1}$$

Einerseits wurde darauf geachtet, dass Einschwingen der Modellparameter abzuwarten (Kapitel 1) und andererseits wurde das direktionale Vergessen für die RLS-

Schätzung (Kapitel 3) eingeführt, um die Rauschsensibilität zu verringern. Offensichtlich zielen beide diese Maßnahmen darauf ab, den Schätzvektor $\hat{\boldsymbol{\theta}}_k$ in Beziehung (F.4.1) zu verbessern. Es liegt jedoch auf der Hand, dass die Auswertung der Gütefunktionen auch dann zu Problemen führen kann, wenn nicht der Schätzvektor $\hat{\boldsymbol{\theta}}_k$, sondern der Ausgangswert y_k zu Verfälschungen führt. Beim Auftreten von Ausgangsrauschen ist eine derartige Verfälschung gegeben. Tritt am Eingang u_k über längere Zeit keine nennenswerte Anregung auf und hat der Vergessensfaktor λ_2 einen Wert $\lambda_2 < 1$, dann kann es passieren, dass die richtige Totzeitinformation in den Gütefunktionen verloren geht. Als Beispiel hierfür sei die folgende, aus (F.3.7) bekannte Strecke mit zeitvariabler Totzeit d_k herangezogen:

$$P\Big(z\Big) = \frac{0.5}{z-0.8}\, z^{-d_k} \quad ; \quad d_k = \begin{cases} 3 \text{ für } 0 \leq \mathrm{k} \leq 22999 \\ 6 \text{ für } 23000 \leq \mathrm{k} \leq 33999 \end{cases} \tag{F.4.2}$$

Für diese Strecke wird eine open-loop Totzeitschätzung eingesetzt - in der Struktur nach Abbildung 29 wird also wiederum nur die Streckenseite betrachtet. Das Eingangssignal $u_{s,k}$ der Länge $N = 40000$ besteht aus sehr spärlich auftretenden Sprüngen zwischen den Werten -1 und 1. Außerdem tritt normalverteiltes Ausgangsrauschen mit Standardabweichung $\sigma_{y_r} = 0.1$ auf. Abbildung 52 zeigt das beschriebene Eingangssignal und das zugehörige Ausgangssignal $y_{r,k}$:

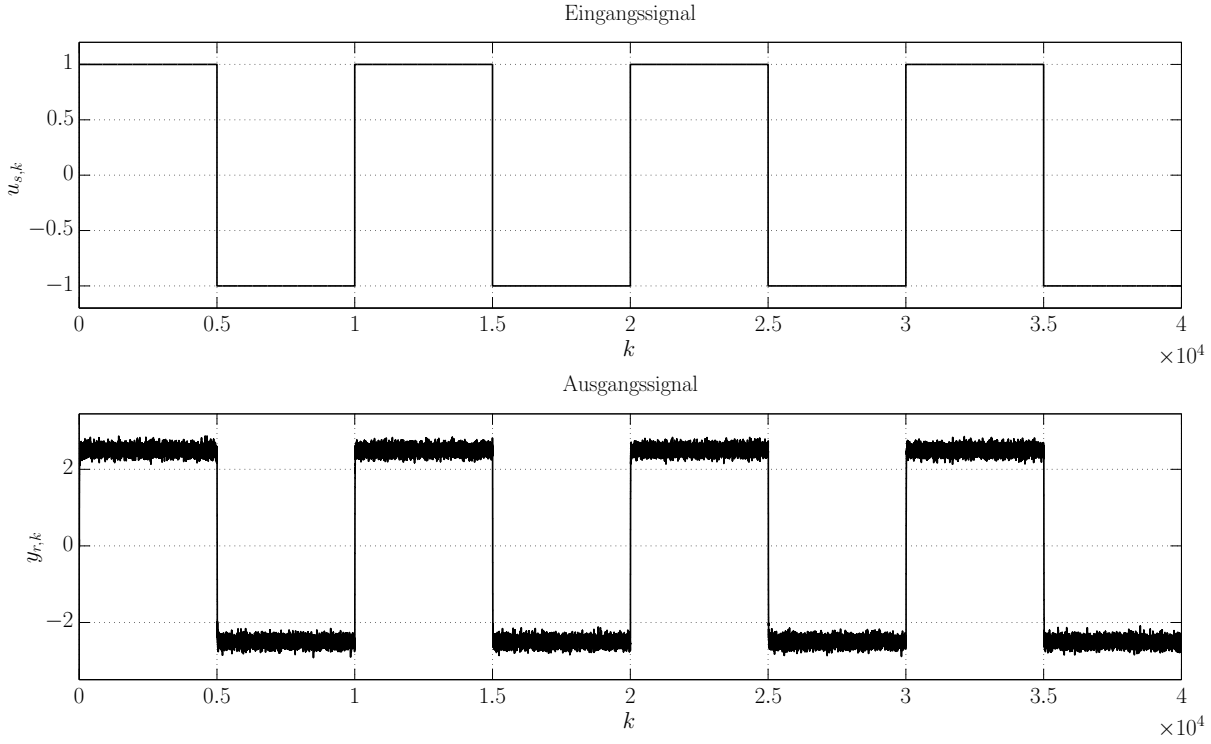

Abbildung 52: Beispiel Sprungerkennung - Ein- und Ausgangssignal

Aufgrund der konstanten Modellparameter und der leicht variierenden Totzeit werden die Vergessensfaktoren auf $\lambda_1 = 1$ und $\lambda_2 = 0.99$ gesetzt. Außerdem wird das Einschwingen der Modellparameter gemäß Kapitel 1 abgewartet ($K_1 = 7500$) und es wird direktionales Vergessen gemäß Kapitel 3 verwendet. Die minimale und maximale Totzeit wurden mit $d_{\min} = 0$ und $d_{\max} = 10$ gewählt und die Initialisierung erfolgte wie üblich entsprechend Beziehung (D.3.41). Abbildung 53 zeigt die geschätzten Modellparameter $\hat{b}_{0,k}$ und $\hat{a}_{0,k}$:

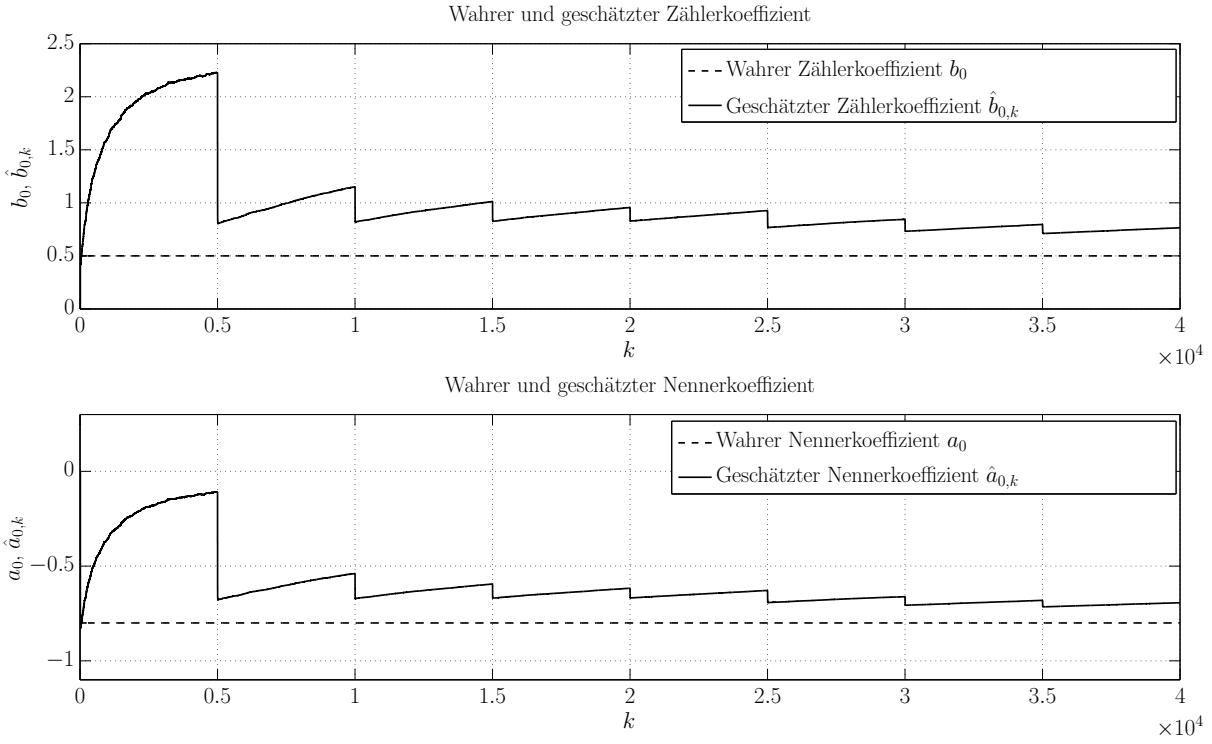

Abbildung 53: Geschätzte Modellparameter ohne Sprungerkennung

Man erkennt, dass die Modellparameter durch das direktionale Vergessen stabil bleiben, auch wenn sie von den wahren Werten ein wenig abweichen. Diese Abweichung ist durch das verhältnismäßig starke Ausgangsrauschen (siehe Abbildung 52) zu begründen. Zudem wurde der Vergessensfaktor für die RLS-Identifikation mit $\lambda_1 = 1$ gewählt und die Signale liefern durch die nur sehr sporadisch auftretenden Eingangssprünge nur selten nützliche Informationen über die Modellparameter. Der Algorithmus hat also immer nur sehr wenig Zeit, um die Parameter zu „lernen“, wodurch es verhältnismäßig lange dauert, bis die geschätzten Modellparameter den korrekten Werten nahe kommen. Da die geschätzten Parameter größenordnungsmäßig dennoch in einem Bereich liegen, der eine sinnvolle Totzeitschätzung durch den Elnaggar-Algorithmus zulässt, kann die angemerkte Abweichung an dieser Stelle hingenommen werden. Die resultierende Totzeitschätzung $\hat{d}_k$ sieht wie folgt aus:

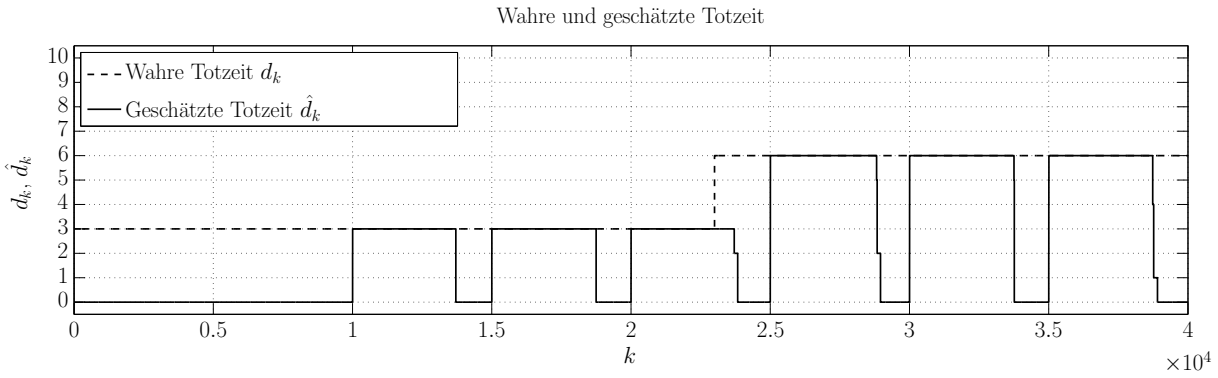

Abbildung 54: Geschätzte Totzeit ohne Sprungerkennung

Hier zeigt sich, dass die Totzeit d_k immer bei Auftreten eines Sprungs des Eingangssignals $u_{s,k}$ richtig identifiziert wird, jedoch geht die richtige Information über die Totzeit nach einiger Zeit verloren. Dies liegt daran, dass nach den Eingangssprüngen immer eine lange Zeit „nichts passiert“. In diesen Phasen werden Änderungen am Ausgang $y_{r,k}$ ausschließlich durch das Ausgangsrauschen hervorgerufen. Somit nimmt das Ausgangsrauschen auch starken Einfluss auf die Auswertung der Gütefunktionen (F.4.1). Außerdem ist zu beachten, dass der Ausgang $y_{r,k}$ im stationären Zustand ohnehin nicht von der Totzeit abhängig ist. Damit bestimmen sich die Gütefunktionen in diesem Zustand tatsächlich nur noch durch das Ausgangsrauschen. In weiterer Folge kann es wie erwähnt passieren, dass eine richtig geschätzte Totzeit bei einem Vergessensfaktor $\lambda_2 < 1$ nach einiger Zeit verloren geht. Genau dieser Fall tritt beim oben erläuterten Beispiel auf, wie in Abbildung 54 zu sehen ist.

Um das eben beschriebene Problem der durch das Ausgangsrauschen verloren gehenden Totzeitinformation zu vermeiden, bietet sich für sprungförmige Eingangsgrößen eine Abhilfe an. In Horch [37] und Isaksson & Horch [38] wird diesbezüglich eine Methode vorgeschlagen, bei welcher die Totzeit mit Hilfe von Laguerre-Approximationen des betrachteten Systems geschätzt wird. Entscheidend ist dabei, dass diese Schätzung nur anhand von Daten von Statten geht, die unmittelbar bei Auftreten eines Arbeitspunktwechsels aufgenommen werden. Daten, die während stationären Zuständen aufgezeichnet werden, werden dagegen für die Totzeitschätzung vernachlässigt. Zur Detektion eines Arbeitspunktwechsels werden dabei exponentielle Filter entsprechend der Herangehensweise nach Cao [5] vorgeschlagen. Da diese Methodik jedoch nicht während des laufenden Betriebs (on-line) umsetzbar ist, wird in Folge versucht, sich einen ähnlichen Ansatz für den Elnaggar-Algorithmus zu Nutze zu machen. So kann man die Auswertung der Gütefunktionen (F.4.1) immer nur dann aktivieren, wenn am Eingang ein Sprung auftritt bzw. gerade aufgetreten ist. Ist dies nicht der Fall, werden die Werte der Gütefunktionen belassen und somit nicht aktualisiert. Dies verhindert, dass die Gütefunktionen während eines stationären Zustands „unerwünschte“ Werte annehmen. Insbesondere erweist es sich als sinnvoll, die Gütefunkti-

onen bei Auftreten eines Eingangssprungs exakt $d_{\max} - d_{\min} + 1$ Samples auszuwerten und danach wieder abzuschalten, also zu belassen. Durch diese Wahl kann bei der Auswahl der minimalen Gütefunktion $J_k(d)$ zwischen allen möglichen Totzeiten $d \in [d_{\min}, d_{\max}]$ ausreichend gut unterschieden werden. Das heißt, dass $d_{\max} - d_{\min} + 1$ Samples genügen, um nach dem Auftreten eines Eingangssprungs die richtige Totzeit schätzen zu können. Zudem wird ein leicht angepasstes Prozedere für den Vergessensfaktor λ_2 angewendet, der in die Auswertung der Totzeit-Gütefunktionen einfließt. Es soll zwar eine Reaktionsfähigkeit auf zeitlich variierende Totzeiten gewahrt bleiben, jedoch ist es wünschenswert, dass die angesprochenen $d_{\max} - d_{\min} + 1$ Samples in den Gütefunktionen alle gleich stark gewichtet werden. Daher wird λ_2 immer bei Auftreten eines Eingangssprungs genau ein Sample lang auf den vom Benutzer angegebenen bzw. durch die Optimierung (siehe Kapitel 2) ermittelten Wert gesetzt. Dies gewährleistet, dass die Totzeitschätzungen bei früheren Eingangssprüngen weniger stark als die Totzeitschätzung beim aktuellen Eingangssprung gewichtet werden. Anschließend wird $\lambda_2 = 1$ gesetzt, um wie angedeutet alle zum aktuellen Eingangssprung gehörigen $d_{\max} - d_{\min} + 1$ Samples gleich zu gewichten.

Zur Umsetzung dieses Ansatzes stellt sich jedoch zwangsläufig die Frage, wie man einen Eingangssprung definiert. Es muss also eine Möglichkeit gefunden werden, dem Algorithmus das Auftreten eines Eingangssprungs mitzuteilen, damit dieser mit der Auswertung der Gütefunktionen beginnen kann. Das Auftreten eines Eingangssprungs muss zudem on-line, also während der Ausführung erkannt werden. Dabei ist es nicht sinnvoll, sehr kleine Änderungen im Eingangssignal schon als Eingangssprung zu definieren, da diese kleinen Änderungen am Ausgang bei Überlagerung von Rauschen oft keine wahrzunehmende Auswirkung haben. In diesem Fall hätte der Elnaggar-Algorithmus nicht die Möglichkeit, die richtige Totzeit zu erkennen. Daher wurde eine Vorgangsweise eingeführt, bei welcher während der Einschwingzeit der Modellparameter (siehe Kapitel 1) eine sinnvolle Schwelle Δu_{Th} zur Detektion eines Eingangssprunges on-line ermittelt wird. Hierfür wird die betragsmäßig maximale Eingangsänderung während der Einschwingzeit festgestellt und diese wird anschließend mit einer Konstante $0 < \mu \leq 1$ multipliziert, um die gesuchte Schwelle Δu_{Th} zu erhalten:

$$\Delta u_{Th} = \mu \max \left\{ \left| u_k - u_{k-1} \right| \right\} \quad \forall \quad 1 \leq k \leq K_1 \tag{F.4.3}$$

Nach der Einschwingzeit, also für $k > K_1$ ist es dann anhand dieser Schwelle möglich, auftretende Eingangssprünge zu detektieren. Die Konstante μ wurde dabei mit $\mu = 0.8$ festgelegt. Damit können auch etwas kleinere Eingangsänderungen als die maximale, während der Einschwingzeit vorkommende Eingangsänderung als Eingangssprung wahrgenommen werden.

Kommt man zum einführenden Beispiel zurück, dann beträgt die Schwelle Δu_{Th} für das Eingangssignal gemäß Abbildung 52 nach Vorschrift (F.4.3) $\Delta u_{Th} = 1.6$. Wendet man diese Schwelle an, und wertet die Gütefunktionen wie erwähnt nach einem Eingangssprung immer $d_{\max} - d_{\min} + 1 = 11$ Samples lang aus, dann erhält man für die geschätzten Modellparameter $\hat{b}_{0,k}$ und $\hat{a}_{0,k}$ erwartungsgemäß keine großen Änderungen:

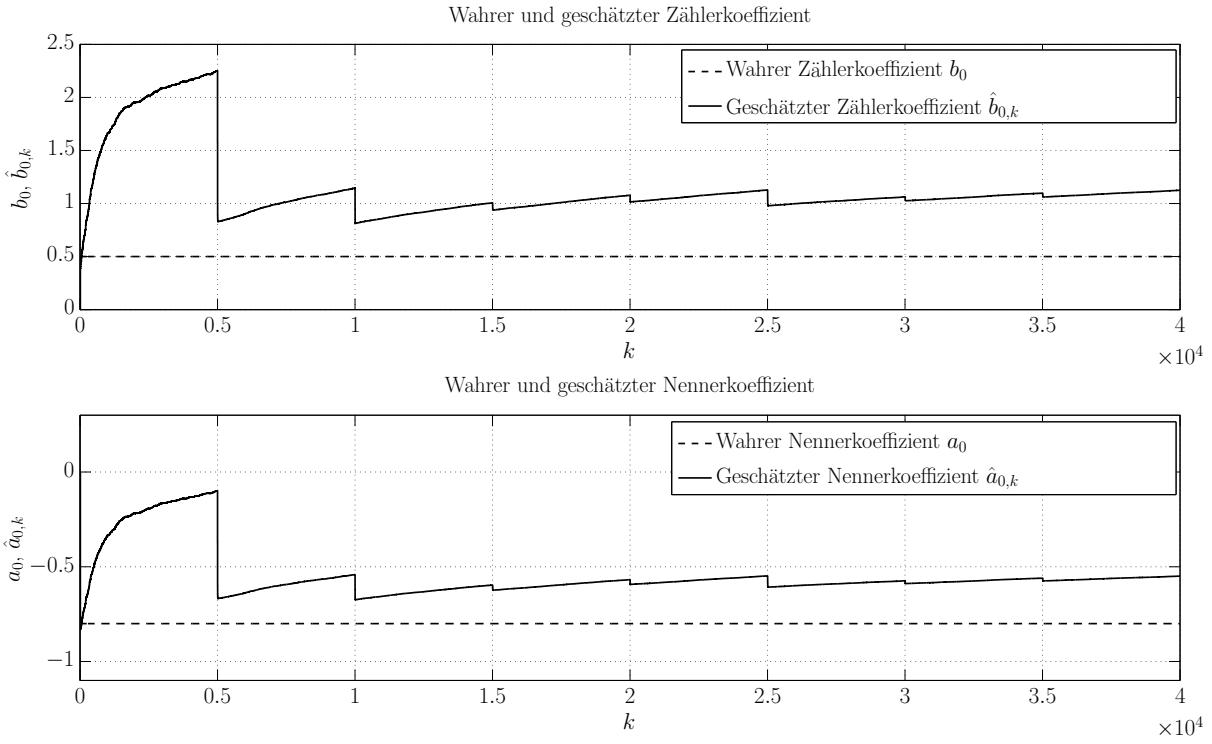

Abbildung 55: Geschätzte Modellparameter mit Sprungerkennung

Auch hier weichen die geschätzten Modellparameter von den wahren Koeffizienten ein wenig ab. Für die Totzeitschätzung $\hat{d}_k$ stellt dies jedoch wie schon bei der Totzeitschätzung ohne Sprungerkennung kein schwerwiegendes Problem dar. Die folgende Abbildung zeigt das zugehörige Ergebnis der Totzeitschätzung:

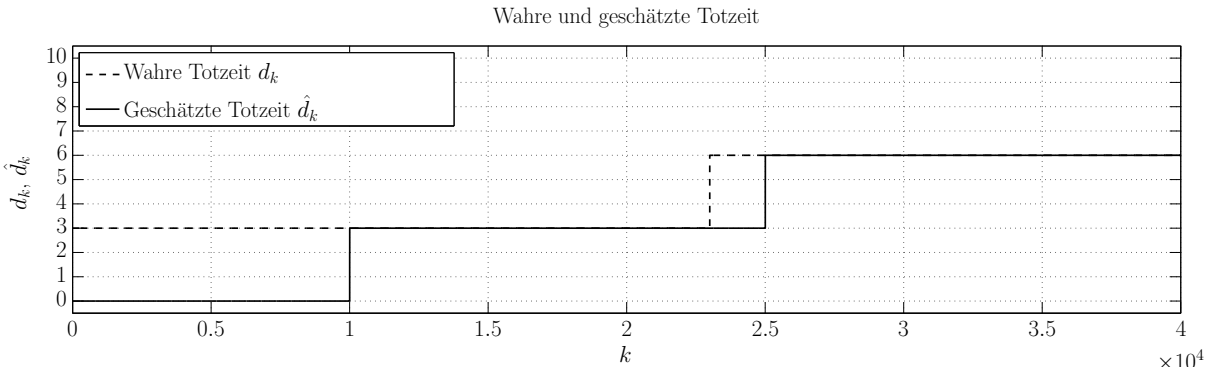

Abbildung 56: Geschätzte Totzeit mit Sprungerkennung

Man kann eine deutliche Verbesserung im Vergleich zu Abbildung 54 erkennen: Durch das „Einfrieren“ der Totzeit-Gütefunktionen $J_k(d)$ während der stationären Zustände geht die richtige Totzeitinformation nicht verloren, und die geschätzte Totzeit $\hat{d}_k$ bleibt auf ihrem richtigen Wert.

Die Sprungerkennung bezüglich der Eingangsgröße ist also ein sinnvolles Mittel, um die Totzeitschätzung bei sprungförmiger Eingangsgröße $u_{s,k}$ und Auftreten von Rauschen am Ausgang $y_{r,k}$ robuster zu machen. Dieses Mittel kann nicht nur auf der Streckenseite, sondern auch auf der Reglerseite angewendet werden. Die Ermittlung der Schwelle geschieht hierbei jedoch nicht bezüglich dem Streckeneingang $u_{s,k}$, sondern bezüglich der Referenz r_k:

$$\Delta r_{Th} = \mu \max\left\{ \left| r_k - r_{k-1} \right| \right\} \quad \forall \quad 1 \le k \le K_1 \tag{F.4.4}$$

Diese Schwelle kann nach der Einschwingzeit K_1 zur Detektion von Referenzsprüngen für den adaptierten Schätz-Algorithmus (E.2.6) auf der Reglerseite verwendet werden. Hier stellte sich heraus, dass es von Nutzen ist, bei Auftreten eines Referenzsprungs zunächst $d_{\max} - d_{\min} + 1$ Samples lang nur die Totzeit $d_{2,k}$ zu schätzen und anschließend $d_{\max} - d_{\min} + 1$ Samples lang nur die Totzeit $d_{1,k}$ zu schätzen. Damit ist gemeint, dass bei Auftreten eines Referenzsprungs erst $d_{\max} - d_{\min} + 1$ Samples lang nur die Gütefunktionen $J_{2,k}(d_2)$ für alle möglichen Totzeiten $d_2 \in [d_{\min}, d_{\max}]$ ausgewertet wird und $d_{2,k}$ geschätzt wird. Danach werden diese Gütefunktionen belassen und es werden $d_{\max} - d_{\min} + 1$ Samples lang nur die Gütefunktionen $J_{1,k}(d_1)$ zur Schätzung von $d_{1,k}$ aktualisiert. Auch auf der Reglerseite wird die von der Streckenseite bekannte, leicht veränderte Vergessensstrategie angewendet. Sowohl bei der Schätzung von d_1, als auch bei der Schätzung von d_2 wird λ_2 immer ein Sample lang auf den angegebenen oder den optimierten Wert gesetzt – anschließend folgen jeweils $d_{\max} - d_{\min}$ Samples mit $\lambda_2 = 1$.

Anzumerken ist jedoch, dass die Sprungerkennung sowie das direktionale Vergessen auf Reglerseite oft nicht unbedingt nötig sind. Dies hat den Grund, dass die Ausgangsgröße auf Streckenseite entsprechend der Struktur nach Abbildung 29 durch $u_{r,k}$ gegeben ist. Das Signal $u_{r,k}$ erhält das Koppelelement in der Praxis meist direkt von einem digital realisierten Regelalgorithmus, weshalb $u_{r,k}$ nur selten mit Rauschen überlagert ist. Im Gegensatz dazu kann auf der Streckenseite (Ausgang $y_{r,k}$) sehr wohl Ausgangsrauschen auftreten, da es sich hier meist um Daten handelt, die von einem analogen Sensor stammen.

Die Verwendung der beschriebenen Sprungerkennung kann sowohl auf Strecken- als auch auf Reglerseite mit der Optimierung der Vergessensfaktoren gemäß Kapitel 2 kombiniert werden. Dabei ist es sinnvoll, die Gütefunktion $I_{y,k}(\lambda_1, \lambda_2)$ nach Beziehung (F.2.8) jeweils nur dann auszuwerten, wenn auch eine Auswertung der Totzeitgütefunktionen $J_k(d)$ bzw. $J_{2,k}(d_2)$ und $J_{1,k}(d_1)$ stattfindet. Das heißt, dass $I_{y,k}(\lambda_1, \lambda_2)$ auf Streckenseite genau wie die Gütefunktionen $J_k(d)$ bei Auftreten eines Eingangssprungs $d_{\max} - d_{\min} + 1$ Samples lang ausgewertet wird. Auf Reglerseite ergibt sich eine Auswertungszeit von $2(d_{\max} - d_{\min} + 1)$ Samples nach einem Referenzsprung.

Zusammenfassend ist festzuhalten, dass die vorgestellte Sprungerkennung ein adäquates Mittel ist, um die Qualität der Totzeitschätzung des Elnaggar-Algorithmus bei sprungförmigen Eingangsgrößen zu verbessern. Die Schätzung wird dadurch robuster und weniger anfällig auf Ausgangsrauschen, ähnlich wie schon durch die Maßnahme des direktionalen Vergessens (Kapitel 3). Die Beschränkung dieser Maßnahme auf sprungförmige Eingangsgrößen ist durchaus praxisrelevant, da in geregelten Systemen häufig Arbeitspunktwechsel stattfinden, während die relevanten Größen zwischen diesen Arbeitspunktwechseln oftmals auf stationären Werten verharren. Diese Arbeitspunktwechsel drücken sich durch ein rechteckförmiges Referenzsignal r_k aus, was häufig zu einem Signal $u_{s,k}$ führt, bei dem ebenfalls große Sprünge auftreten. Dadurch wird es möglich, dass die Sprungerkennung nicht nur auf der Regler-, sondern auch auf der Streckenseite eingesetzt werden kann, da eine funktionierende Detektion von Sprüngen im Signal $u_{s,k}$ gewährleistet ist. Dies ist deshalb von Bedeutung, da die Sprungerkennung in der Praxis wie erwähnt insbesondere auf der Streckenseite Sinn macht.

5 Zusammenfassung

In diesem Kapitel wurden vier Maßnahmen zur Verbesserung der Anwendbarkeit des Elnaggar-Algorithmus zur On-Line-Totzeitschätzung eingeführt: Die Berücksichtigung von Einschwingvorgängen (Kapitel 1), die Optimierung der Vergessensfaktoren (Kapitel 2), das direktionale Vergessen zur Verringerung der Rauschsensibilität (Kapitel 3) und die Sprungerkennung bei sprungförmigen Eingangsgrößen (Kapitel 4). All diese Maßnahmen sollen bei der Implementierung des Algorithmus optional einsetzbar sein. Tabelle 4 und Tabelle 5 zeigen den spezifizierten Ablauf der Algorithmen sowohl auf Strecken- als auch auf Reglerseite gemäß Abbildung 29:

<table>
<tr><th>Zeitschritte</th><th>Auszuführende Schritte</th><th>Sprungerkennung</th><th>Optimierung</th></tr>
<tr><td>$k = 0$
(Initialisierung)</td><td colspan="2">$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad ; \quad \hat{\boldsymbol{\theta}}_0 = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \quad ; \quad \hat{d}_0 = d_{\min}$
Vom Benutzer festzulegen: $m, n, d_{\min}, d_{\max}, K_1, K_2$</td><td rowspan="2"></td><td>Zusätzlich festzulegen wenn Optimierung aktiviert: K_3
Ansonsten:
λ_1 und λ_2</td></tr>
<tr><td rowspan="3">$0 \le k \le K_1$
(Einschwingphase Modellparameter)</td><td colspan="2">$\hat{\boldsymbol{\phi}}_k = \begin{pmatrix} -y_{r,k-1} & \cdots & -y_{r,k-n} & u_{s,k+m-n-\hat{d}_{k-1}} & \cdots & u_{s,k-n-\hat{d}_{k-1}} \end{pmatrix}^T$
$\hat{d}_{k-1} = d_{\min} \;\forall\; k \le K_1$</td><td></td></tr>
<tr><td>Exponentielles Vergessen</td><td>Direktionales Vergessen</td><td rowspan="2">Festlegung der Schwelle Δu_{Th} für die Detektion eines Eingangssprungs:
$\Delta u_{Th} = \mu \max \left\{ \left\| u_k - u_{k-1} \right\| \right\}$</td><td rowspan="3">Parallele Durchführung dieser Schritte für alle 36 möglichen Kombinationen aus λ_1 und λ_2 gemäß Tabelle 3</td></tr>
<tr><td>$\mathbf{k}_k = \dfrac{\mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}{\lambda_1 + \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \hat{\boldsymbol{\phi}}_k}$
$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k \left(y_{r,k} - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \right)$
$\mathbf{P}_k = \dfrac{1}{\lambda_1} \left(\mathbf{P}_{k-1} - \mathbf{k}_k \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \right)$</td><td>$\overline{\mathbf{P}}_{k-1} = \begin{cases} \mathbf{P}_{k-1} + \dfrac{1-\lambda_1}{\lambda_1} \dfrac{\hat{\boldsymbol{\phi}}_k \hat{\boldsymbol{\phi}}_k^T}{\hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1}^{-1} \hat{\boldsymbol{\phi}}_k} & \text{für } \left\| \hat{\boldsymbol{\phi}}_k \right\| > \varepsilon \\ \mathbf{P}_{k-1} & \text{für } \left\| \hat{\boldsymbol{\phi}}_k \right\| \le \varepsilon \end{cases}$
$\mathbf{P}_k = \overline{\mathbf{P}}_{k-1} - \dfrac{\overline{\mathbf{P}}_{k-1} \hat{\boldsymbol{\phi}}_k \hat{\boldsymbol{\phi}}_k^T \overline{\mathbf{P}}_{k-1}}{1 + \hat{\boldsymbol{\phi}}_k^T \overline{\mathbf{P}}_{k-1} \hat{\boldsymbol{\phi}}_k}$
$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{P}_k \hat{\boldsymbol{\phi}}_k \left(y_{r,k} - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \right)$</td></tr>
<tr><td>$K_1 + 1 \le k \le K_1 + K_2$
(Einschwingphase Totzeit)</td><td colspan="2">$J_k\left(d\right) = \lambda_2 J_{k-1}\left(d\right) + \left[y_{r,k} - \boldsymbol{\phi}_k^T\left(d\right) \hat{\boldsymbol{\theta}}_k \right]^2 \quad \forall \quad d \in \left[d_{\min}, d_{\max} \right]$
$\hat{d}_k = \arg\min \left\{ J_k\left(d\right) \right\} \quad \forall \quad d \in \left[d_{\min}, d_{\max} \right]$</td><td rowspan="3">Nur während der $d_{\max} - d_{\min} + 1$ Samples nach einem detektiertem Eingangssprung; mit angepasstem Vergessensfaktor λ_2</td><td></td></tr>
<tr><td>$K_1 + K_2 + 1 \le k \le K_1 + K_2 + K_3$
(Optimierungsphase)</td><td colspan="2">$I_{y,k}\left(\lambda_1, \lambda_2\right) = I_{y,k-1}\left(\lambda_1, \lambda_2\right) + \left(y_{r,k} - \hat{\boldsymbol{\phi}}_k\left(\lambda_1, \lambda_2\right) \hat{\boldsymbol{\theta}}_k\left(\lambda_1, \lambda_2\right) \right)^2$</td><td>Nur bei aktivierter Optimierung der Vergessensfaktoren</td></tr>
<tr><td>$k = K_1 + K_2 + K_3$
(Auswahl Vergessensfaktoren)</td><td colspan="2">$\left(\lambda_{1,\text{opt}}, \lambda_{2,\text{opt}}\right) = \arg\min \left\{ I_{y,k}\left(\lambda_1, \lambda_2\right) \right\}$</td><td rowspan="2"></td></tr>
<tr><td>$k \ge K_1 + K_2 + K_3 + 1$
(Normale Totzeitschätzung)</td><td colspan="2">Geschätzte Totzeit $\hat{d}_k$ ab diesem Zeitpunkt für Weiterverwendung geeignet</td><td>Nur Algorithmus mit $(\lambda_1, \lambda_2) = (\lambda_{1,\text{opt}}, \lambda_{2,\text{opt}})$ wird weitergerechnet</td></tr>
</table>

Tabelle 4: Ablauf des Algorithmus für die Streckenseite

<table>
<tr><th>Zeitschritte</th><th colspan="2">Auszuführende Schritte</th><th>Sprungerkennung</th><th>Optimierung</th></tr>
<tr><td>$k = 0$
(Initialisierung)</td><td colspan="2">$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad ; \quad \hat{\boldsymbol{\theta}}_0 = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \quad ; \quad \hat{d}_{1,0} = \hat{d}_{2,0} = d_{\min}$
Vom Benutzer festzulegen: $m, n, d_{\min}, d_{\max}, K_1, K_2$</td><td></td><td>Zusätzlich festzulegen wenn Optimierung aktiviert: K_3
Ansonsten:
λ_1 und λ_2</td></tr>
<tr><td rowspan="3">$0 \leq k \leq K_1$
(Einschwingphase Modellparameter)</td><td colspan="2">$\hat{\boldsymbol{\phi}}_k = \begin{pmatrix} -u_{r,k-1} & \cdots & -u_{r,k-n} & r_{k+m-n-\hat{d}_{2,k-1}} - y_{s,k+m-n-\hat{d}_{1,k-1}-\hat{d}_{2,k-1}} & \cdots & r_{k-n-\hat{d}_{2,k-1}} - y_{s,k-n-\hat{d}_{1,k-1}-\hat{d}_{2,k-1}} \end{pmatrix}^T$
$\hat{d}_{1,k-1} = \hat{d}_{2,k-1} = d_{\min} \;\forall\; k \leq K_1$</td><td></td><td></td></tr>
<tr><td>Exponentielles Vergessen</td><td>Direktionales Vergessen</td><td rowspan="2">Festlegung der Schwelle Δu_{Th} für die Detektion eines Referenzsprungs:
$\Delta r_{Th} = \mu \max \left\{ \left\vert r_k - r_{k-1} \right\vert \right\}$</td><td rowspan="2">Parallele Durchführung dieser Schritte für alle 36 möglichen Kombinationen aus λ_1 und λ_2 gemäß Tabelle 3</td></tr>
<tr><td>$\mathbf{k}_k = \dfrac{\mathbf{P}_{k-1}\hat{\boldsymbol{\phi}}_k}{\lambda_1 + \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \hat{\boldsymbol{\phi}}_k}$
$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{k}_k \left(u_{r,k} - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \right)$
$\mathbf{P}_k = \dfrac{1}{\lambda_1} \left(\mathbf{P}_{k-1} - \mathbf{k}_k \hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1} \right)$</td><td>$\overline{\mathbf{P}}_{k-1} = \begin{cases} \mathbf{P}_{k-1} + \dfrac{1-\lambda_1}{\lambda_1} \dfrac{\hat{\boldsymbol{\phi}}_k \hat{\boldsymbol{\phi}}_k^T}{\hat{\boldsymbol{\phi}}_k^T \mathbf{P}_{k-1}^{-1} \hat{\boldsymbol{\phi}}_k} & \text{für } \left\vert \hat{\boldsymbol{\phi}}_k \right\vert > \varepsilon \\ \mathbf{P}_{k-1} & \text{für } \left\vert \hat{\boldsymbol{\phi}}_k \right\vert \leq \varepsilon \end{cases}$
$\mathbf{P}_k = \overline{\mathbf{P}}_{k-1} - \dfrac{\overline{\mathbf{P}}_{k-1} \hat{\boldsymbol{\phi}}_k \hat{\boldsymbol{\phi}}_k^T \overline{\mathbf{P}}_{k-1}}{1 + \hat{\boldsymbol{\phi}}_k^T \overline{\mathbf{P}}_{k-1} \hat{\boldsymbol{\phi}}_k}$
$\hat{\boldsymbol{\theta}}_k = \hat{\boldsymbol{\theta}}_{k-1} + \mathbf{P}_k \hat{\boldsymbol{\phi}}_k \left(u_{r,k} - \hat{\boldsymbol{\phi}}_k^T \hat{\boldsymbol{\theta}}_{k-1} \right)$</td></tr>
<tr><td>$K_1 + 1 \leq k \leq K_1 + K_2$
(Einschwingphase Totzeit)</td><td colspan="2">$J_{2,k}\left(d_2 \right) = \lambda_2 J_{2,k-1}\left(d_2 \right) + \left[u_{r,k} - \boldsymbol{\phi}_k^T \left(\hat{d}_{1,k-1}, d_2 \right) \hat{\boldsymbol{\theta}}_k \right]^2 \quad \forall \quad d_2 \in \left[d_{\min}, d_{\max} \right]$
$\hat{d}_{2,k} = \arg\min \left\{ J_{2,k}\left(d_2 \right) \right\} \quad \forall \quad d_2 \in \left[d_{\min}, d_{\max} \right]$
$J_{1,k}\left(d_1 \right) = \lambda_2 J_{1,k-1}\left(d_1 \right) + \left[u_{r,k} - \boldsymbol{\phi}_{1,k}^T \left(d_1, \hat{d}_{2,k} \right) \hat{\boldsymbol{\theta}}_k \right]^2 \quad \forall \quad d_1 \in \left[d_{\min}, d_{\max} \right]$
$\hat{d}_{1,k} = \arg\min \left\{ J_{1,k}\left(d_1 \right) \right\} \quad \forall \quad d_1 \in \left[d_{\min}, d_{\max} \right]$</td><td rowspan="2">Nur während der $2(d_{\max} - d_{\min} + 1)$ Samples nach einem detektiertem Referenzsprung; mit angepasstem Vergessensfaktor λ_2</td><td rowspan="2">Nur bei aktivierter Optimierung der Vergessensfaktoren</td></tr>
<tr><td>$K_1 + K_2 + 1 \leq k \leq K_1 + K_2 + K_3$
(Optimierungsphase)</td><td colspan="2">$I_{u,k}\left(\lambda_1, \lambda_2 \right) = I_{u,k-1}\left(\lambda_1, \lambda_2 \right) + \left(u_{r,k} - \hat{\boldsymbol{\phi}}_k \left(\lambda_1, \lambda_2 \right) \hat{\boldsymbol{\theta}}_k \left(\lambda_1, \lambda_2 \right) \right)^2$</td></tr>
<tr><td>$k = K_1 + K_2 + K_3$
(Auswahl Vergessensfaktoren)</td><td colspan="2">$\left(\lambda_{1,\text{opt}}, \lambda_{2,\text{opt}} \right) = \arg\min \left\{ I_{u,k}\left(\lambda_1, \lambda_2 \right) \right\}$</td><td rowspan="2"></td><td></td></tr>
<tr><td>$k \geq K_1 + K_2 + K_3 + 1$
(Normale Totzeitschätzung)</td><td colspan="2">Geschätzte Totzeiten $\hat{d}_{1,k}$ und $\hat{d}_{2,k}$ ab diesem Zeitpunkt für Weiterverwendung geeignet</td><td>Nur Algorithmus mit $(\lambda_1, \lambda_2) = (\lambda_{1,\text{opt}}, \lambda_{2,\text{opt}})$ wird weitergerechnet</td></tr>
</table>

Tabelle 5: Ablauf des Algorithmus für die Reglerseite

G Implementierung in MATLAB/SIMULINK

In Kapitel D wurde mit dem Elnaggar-Algorithmus eine modellbasierte Methode zur On-Line-Totzeitschätzung gefunden, die der Zielsetzung dieser Arbeit entspricht. Sie ist weder in Bezug auf das Eingangssignal, noch in Bezug auf die Charakteristik des Modells beschränkt. Der Elnaggar-Algorithmus konnte anschließend in Kapitel E für den angedachten Einsatzzweck innerhalb eines Koppelelements in einem geregelten System adaptiert werden. Zuletzt folgten in Kapitel F einige Maßnahmen, die eine Verbesserung der Anwendbarkeit des Algorithmus verfolgen. Die vollständigen Algorithmen sind für die Streckenseite in Tabelle 4 und für die Reglerseite in Tabelle 5 zu finden. Ziel des vorliegenden Kapitels ist eine Implementierung dieser beiden Algorithmen in MATLAB/SIMULINK.

1 Allgemeines zur Implementierung

Da die Implementierung in einer späteren Phase in ausführbaren C-Code übersetzt werden soll, wurde eine Umsetzung mit Hilfe einer *Matlab Function* in SIMULINK gewählt. Diese *Matlab Function* wurde zur Wahrung der Übersichtlichkeit in einem *Subsystem* integriert. Als Eingänge für das *Subsystem* und auch der *Matlab Function* dienen gemäß Abbildung 29 auf Streckenseite die Signale $u_{s,k}$ und $y_{r,k}$ bzw. auf Reglerseite die Signale r_k, $y_{s,k}$ und $u_{r,k}$. Um die für die Algorithmen notwendigen Verzögerungen der Signale zu realisieren, wurden sogenannte *Tapped Delay*-Blöcke verwendet. Diese erhalten ein Signal x_k als Eingang und geben entsprechende Verzögerungen $x_{k-1}, x_{k-2}, \ldots$ bis zu einer festgelegten maximalen Verzögerung aus. Dies stellt sicher, das den Algorithmen zu jedem Zeitschritt k alle benötigten Verzögerungen der in den jeweiligen Rechenschritten herangezogenen Signale zur Verfügung stehen. Um zudem die nötigen Iterationen aller in den Algorithmen vorkommenden Größen zu ermöglichen, werden diese Größen von der *Matlab Function* als Ausgänge ausgegeben und anschließend über *Unit Delay*-Blöcke als Eingänge zur *Matlab Function* zurückgeführt. Wichtig ist dabei eine korrekte Initialisierung der *Unit Delay*-Blöcke hinsichtlich der Werte und der Dimension. Als Ausgänge der angesprochenen *Subsysteme* dienen die geschätzten Totzeiten $\hat{d}_k$ (Streckenseite) bzw. $\hat{d}_{1,k}$ und $\hat{d}_{2,k}$ (Reglerseite).

2 Zeitlicher Ablauf

Die vollständigen Versionen der adaptierten Elnaggar-Algorithmen (siehe Tabelle 4 und Tabelle 5) benötigen eine Angabe der drei Parameter K_1, K_2 und K_3 in Samples. Diese kennzeichnen die Einschwingzeit der geschätzten Modellparameter, die Einschwingzeit der Totzeitschätzung und die Optimierungszeit. Da eine Angabe dieser Parameter in Samples in Bezug auf die Anwenderfreundlichkeit nicht optimal wäre, können diese stattdessen mit Hilfe einer fixen Abtastzeit T_d als absolute Zeiten T_1, T_2 und T_3 angegeben werden:

$$\begin{aligned} T_1 &= T_d K_1 \\ T_2 &= T_d K_2 \\ T_3 &= T_d K_3 \end{aligned} \tag{G.2.1}$$

Zieht man zur Abarbeitung der Algorithmen diese absoluten Zeitdauern heran, dann benötigt man zur Ablaufsteuerung eine absolute Zeit t anstatt der Anzahl der Samples k. Dies erreicht man durch einen *Discrete-Time Integrator* in SIMULINK, welcher mit einer Verstärkung T_d parametriert ist. Zudem wird dieser Integrator so konfiguriert, dass er sich über den Eingang zurücksetzen lässt. Dies ermöglicht ein manuelles Aktivieren bzw. Deaktivieren der Totzeitschätzung durch den Benutzer über einen *on/off*-Eingang des Subsystems. Liegt an diesem Eingang eine logische 1 an, dann beginnt der *Discrete-Time Integrator* mit um T_d verstärkten Integration der Konstante 1, wodurch dem Algorithmus die absolute Zeit t zur Verfügung steht. Sobald jedoch eine logische 0 anliegt, wird die gesamte Totzeitschätzung zurückgesetzt. Die absolute Zeit verharrt dann auf $t = 0$ und alle relevanten Größen nehmen ihre Initialisierungswerte an.

Durch die Verwendung der absoluten Zeit t anstatt der Sampleanzahl k ergibt sich der zeitliche Ablauf (siehe linke Spalte von Tabelle 4 und Tabelle 5) wie folgt:

<table>
<tr><td>$t = 0$</td><td colspan="2">Initialisierung</td></tr>
<tr><td>$0 \leq t \leq T_1$</td><td colspan="2">Einschwingphase Modellparameter</td></tr>
<tr><td>$T_1 < t \leq T_1 + T_2$</td><td colspan="2">Einschwingphase Totzeit</td></tr>
<tr><td>$T_1 + T_2 < t \leq T_1 + T_2 + T_3$</td><td>Optimierungsphase</td><td rowspan="2">Nur bei aktivierter Optimierung</td></tr>
<tr><td>$t = T_1 + T_2 + T_3$</td><td>Auswahl Vergessensfaktoren</td></tr>
<tr><td>$t > T_1 + T_2 + T_3$</td><td colspan="2">Normale Totzeitschätzung</td></tr>
</table>

Tabelle 6: Zeitlicher Ablauf der Totzeitschätzung in der Implementierung

3 SIMULINK-Subsysteme

Im Rahmen dieser Arbeit wurden drei SIMULINK-Subsysteme erstellt, welche die vollständigen Elnaggar-Algorithmen nach Tabelle 4 und Tabelle 5 mit dem zeitlichen Ablauf gemäß Tabelle 6 enthalten. Dabei dient jeweils ein Subsystem der Totzeitschätzung ausschließlich auf der Strecken- bzw. Reglerseite, während das dritte Subsystem beide Seiten umfasst. Zur besseren Verständlichkeit wurden die Eingangs- und Ausgangssignale der Subsysteme umbenannt, um einen klareren Bezug zur Streckenseite (*Plant*) bzw. Reglerseite (*Controller*) zu schaffen. Die Eingangssignale für das Subsystem zur Totzeitschätzung auf der Streckenseite lauten $u_{\text{Plant}} \triangleq u_{s,k}$ bzw. $y_{\text{Plant}} \triangleq y_{r,k}$ und das Ausgangssignal heißt $d_{\text{Plant}} = \hat{d}_k$. Beim Subsystem zur Totzeitschätzung auf der Reglerseite sind die Eingänge mit $r_{\text{Contr.}} \triangleq r_k$, $y_{\text{Contr.}} \triangleq y_{s,k}$ bzw. $u_{\text{Contr.}} \triangleq u_{r,k}$ bezeichnet, während die Totzeitschätzungen $d_{1,\text{Contr.}} \triangleq \hat{d}_{1,k}$ bzw. $d_{2,\text{Contr.}} \triangleq \hat{d}_{2,k}$ heißen. Beim Subsystem, dass sowohl die Totzeit auf der Streckenseite als auch die beiden Totzeiten auf der Reglerseite schätzt, sind alle soeben genannten Ein- und Ausgänge gemeinsam vorhanden. Als weiterer Eingang aller Subsysteme dient wie im vorigen Kapitel 2 beschrieben der Port *on/off* zur Aktivierung bzw. Deaktivierung der Totzeitschätzung. Abbildung 57 zeigt die drei mit MBTDE (*Model-Based Time-Delay-Estimation*) betitelten Subsysteme:

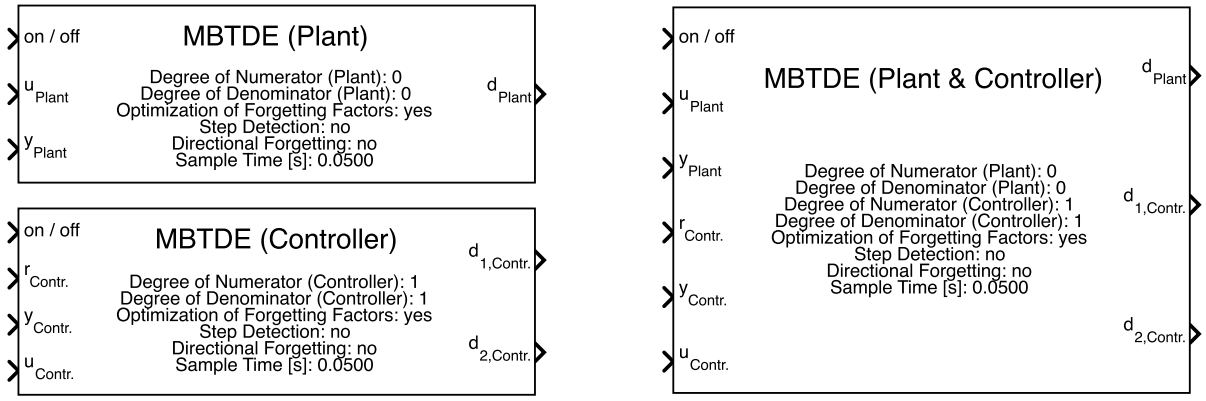

Abbildung 57: SIMULINK-Subsysteme zur Implementierung der Totzeitschätzung

Innerhalb der Subsysteme sind nun wie in Kapitel 1 und Kapitel 2 beschrieben die entsprechenden *Matlab Functions* zur Abarbeitung der Algorithmen, die *Unit Delays* zur Verzögerung der Signale und der rücksetzbare *Discrete-Time Integrator* zu finden. Abbildung 58 und Abbildung 59 zeigen das Innenleben der Subsysteme für die Strecken- und Reglerseite. Die entsprechende Abbildung für das Subsystem zur Totzeitschätzung auf beiden Seiten ist nicht dargestellt, da hier genau die beiden Strukturen dieser beiden Abbildungen gemeinsam enthalten sind.

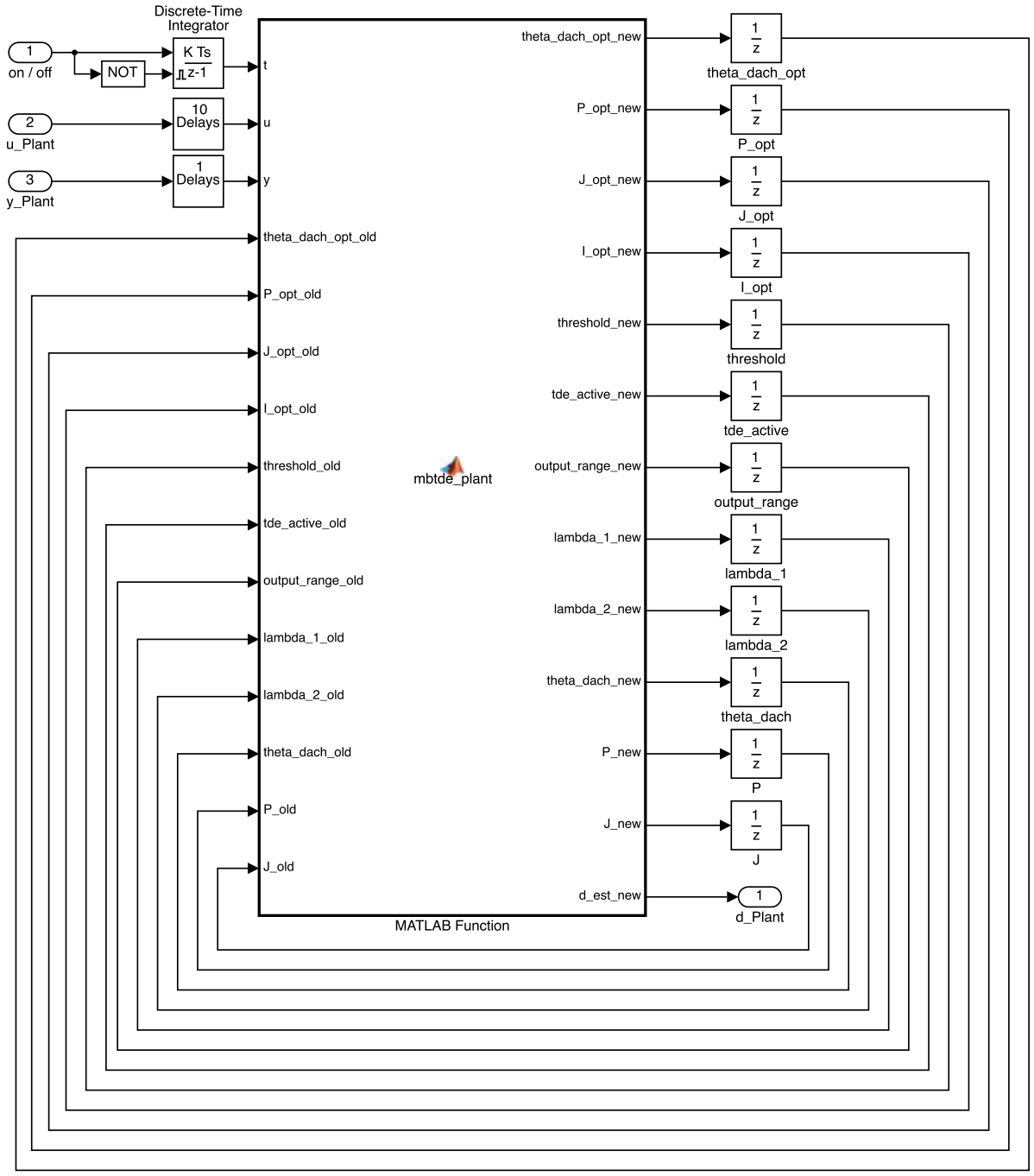

Abbildung 58: Subsystem zur Totzeitschätzung auf der Streckenseite

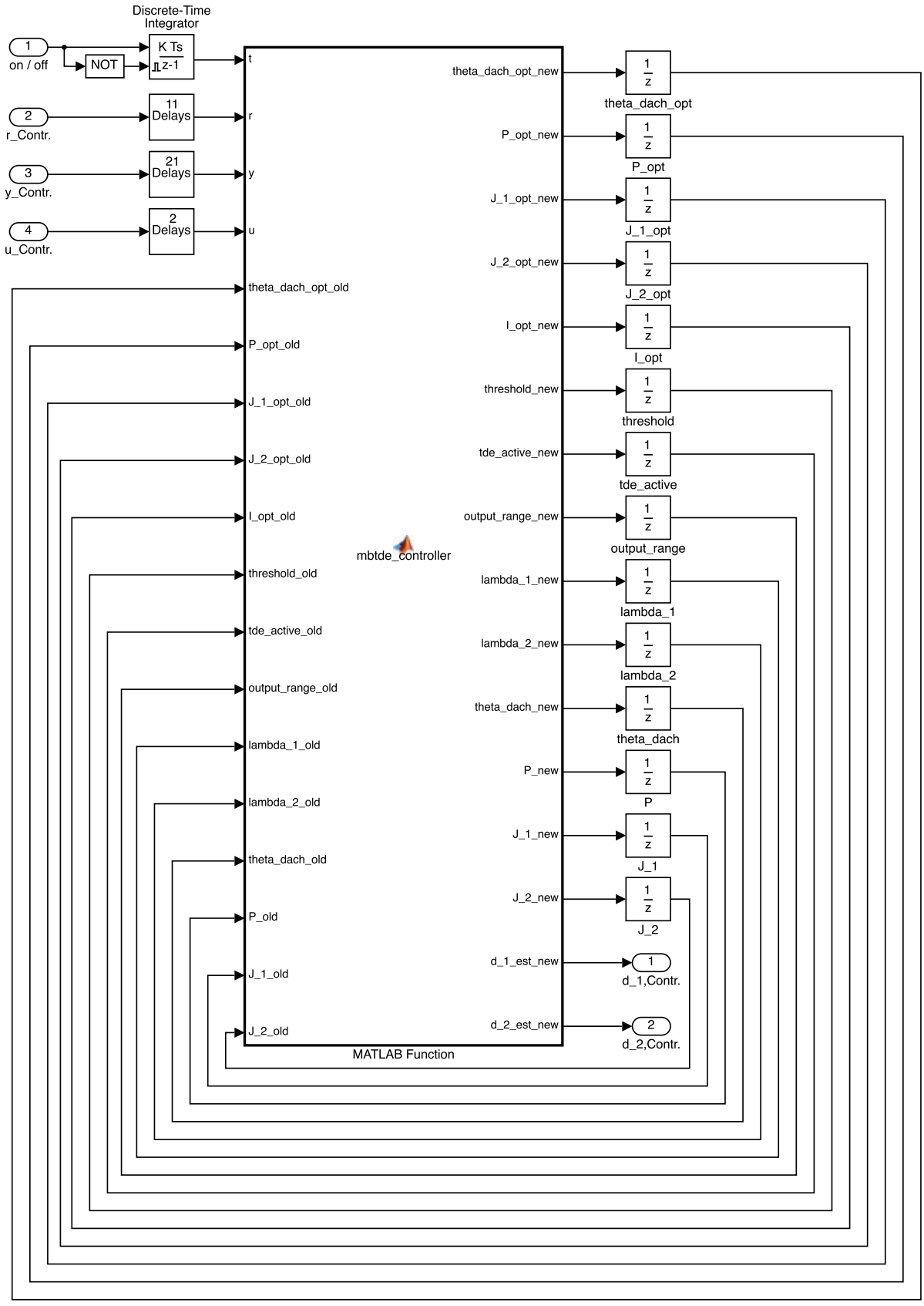

Abbildung 59: Subsystem zur Totzeitschätzung auf der Reglerseite

4 Maskierung der Subsysteme

Wie in Abbildung 57 zu sehen ist, sind die jeweiligen Parametrierungen der Algorithmen auf der Oberfläche der Subsysteme eingetragen. Diese können vom Benutzer über eine Maske festgelegt werden, welche durch einen Doppelklick auf das Subsystem erreichbar ist. Man gelangt zu einer Oberfläche, auf welcher alle Parameter, die der entsprechende Algorithmus benötigt, in Eingabefeldern manuell spezifiziert werden können. Zudem können Zusatzoptionen, wie beispielsweise das direktionale Vergessen durch anzuklickende Häkchen aktiviert werden. Die folgende Tabelle zeigt die jeweils einzustellenden Parameter:

Parameter	Bezeichnung	Streckenseite	Reglerseite	Beide
Zählergrad Strecke m_P	*Degree of Numerator (Plant)*	Ja	Nein	Ja
Nennergrad Strecke n_P	*Degree of Denominator (Plant)*	Ja	Nein	Ja
Zählergrad Regler m_C	*Degree of Numerator (Controller)*	Nein	Ja	Ja
Nennergrad Regler n_C	*Degree of Denominator (Controller)*	Nein	Ja	Ja
Minimale Totzeit $d_{\min}$	*Minimal Time Delay*	Ja	Ja	Ja
Maximale Totzeit $d_{\max}$	*Maximal Time Delay*	Ja	Ja	Ja
Einschwingzeit Modellparameter T_1	*Settling Time RLS*	Ja	Ja	Ja
Einschwingzeit Totzeit T_2	*Settling Time TDE*	Ja	Ja	Ja
Optimierung der Vergessensfaktoren	*Optimization of Forgetting Factors*	Optional aktivierbar	Optional aktivierbar	Optional aktivierbar
Optimierungszeit T_3	*Optimization Time*	Wenn Opt. aktiviert	Wenn Opt. aktiviert	Wenn Opt. aktiviert
Vergessensfaktor λ_1	*Forgetting Factor (RLS)*	Wenn Opt. deaktiviert	Wenn Opt. deaktiviert	Wenn Opt. deaktiviert
Vergessensfaktor λ_2	*Forgetting Factor (TDE)*	Wenn Opt. deaktiviert	Wenn Opt. deaktiviert	Wenn Opt. deaktiviert
Sprungerkennung	*Step Detection*	Optional aktivierbar	Optional aktivierbar	Optional aktivierbar
Direktionales Vergessen	*Directional Forgetting*	Optional aktivierbar	Optional aktivierbar	Optional aktivierbar
Abtastzeit T_d	*Sample Time*	Ja	Ja	Ja

Tabelle 7: In der Maske zu spezifizierende Algorithmus-Parameter

H Anwendungsbeispiele

In diesem Kapitel soll die in Kapitel G erläuterte Implementierung des Algorithmus zur Totzeitschätzung anhand zweier Anwendungsbeispiele getestet werden.

1 Simulationsbeispiel „Driveline“

Der erste Test behandelt ein Simulationsbeispiel, welches mit dem Co-Simulationstool ICOS durchgeführt wurde. Simuliert wird dabei die Drehzahl-Regelung einer Driveline, also eines Antriebsstrangs. Die Eingangsgröße (Stellsignal) für den Antriebsstrang ist gegeben durch das Drehmoment, Ausgangsgröße ist die Drehzahl. Als Referenz werden Drehzahlsprünge vorgegeben. Die Signale u_s, $y_{r,k}$, r_k, $y_{s,k}$ und $u_{r,k}$ gemäß Abbildung 29 wurden bei der Simulation gespeichert. Sie sehen wie folgt aus:

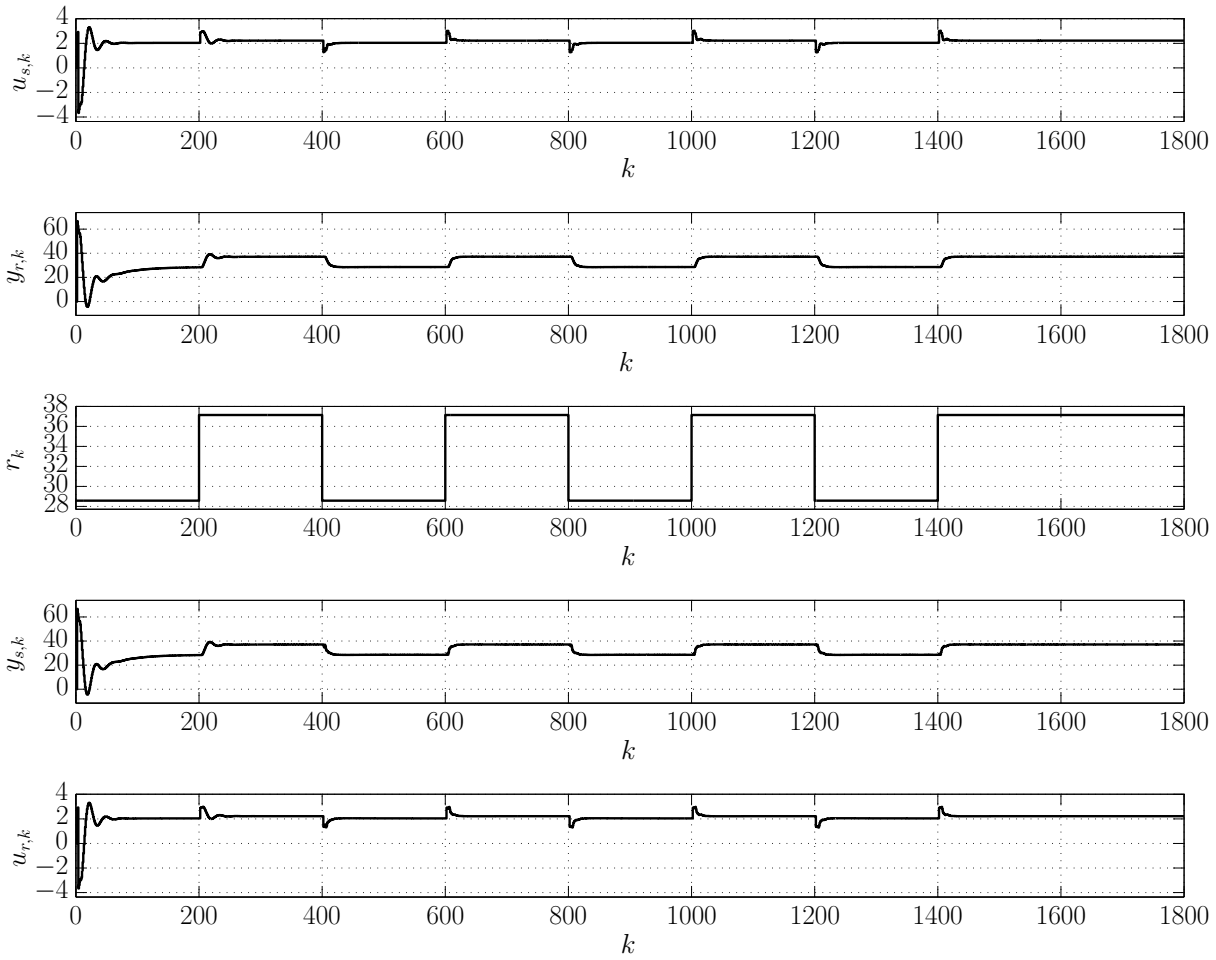

Abbildung 60: Signale für das erste Anwendungsbeispiel

Um die Totzeiten d_k , $d_{1,k}$ und $d_{2,k}$ schätzen zu können, wurde das implementierte Subsystem zur Schätzung der Totzeiten auf beiden Seiten (siehe Abbildung 57 rechts) verwendet. Da über die Regelung bekannt ist, dass es sich um einen klassischen PI-Regler handelt, wurden die Ordnungen $m_C = 1$ und $n_C = 1$ für die Reglerseite gewählt. Für die Streckenseite wird eine genauere Betrachtung der dynamischen Vorgänge im System bewusst unterlassen. Stattdessen schließt man aus der Gestalt des Signale $u_{s,k}$ und $y_{r,k}$ (siehe Abbildung 60) auf ein Verhalten, dass einem System zweiter Ordnung entspricht. Daher werden $m_P = 0$ und $n_P = 2$ gewählt. Für die minimale und maximale Totzeit wurden $d_{\min} = 0$ und $d_{\max} = 10$ festgelegt. Außerdem wurde bei einer Abtastzeit von $T_d = 0.01$ s ein zeitlicher Ablauf mit $T_1 = 3$ s (300 Samples) Einschwingzeit für die geschätzten Parameter und $T_2 = 2$ s (200 Samples) für die geschätzte Totzeit eingestellt. Zudem wurde die Optimierung der Vergessensfaktoren mit einer Optimierungszeit von $T_3 = 4$ s (400 Samples) aktiviert. Auf die Sprungerkennung und das direktionales Vergessen wurde dagegen verzichtet.

Die gespeicherten Signale wurden zum Test der Totzeitschätzung als *From Workspace*-Blöcke mit den entsprechenden Eingängen des SIMULINK-Subsystems verbunden. Die geschätzten Totzeiten $\hat{d}_k$, $\hat{d}_{1,k}$ und $\hat{d}_{2,k}$ werden anschließend mit Hilfe von *To Workspace*-Blöcken in MATLAB gespeichert. Außerdem wird die Totzeitschätzung durch eine Konstante 1 am *on/off*-Eingang dauerhaft aktiviert. Abbildung 61 zeigt den entsprechenden SIMULINK-Koppelplan:

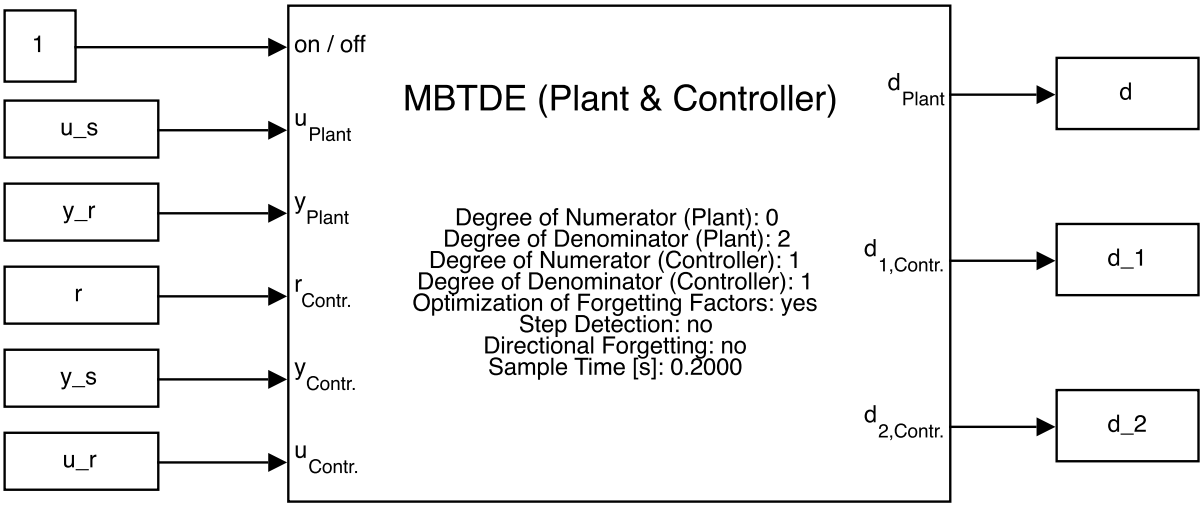

Abbildung 61: Simulink-Koppelplan für das erste Anwendungsbeispiel

Anhand der obigen Struktur werden die im geregelten System auftretenden Totzeiten mit Hilfe des Elnaggar-Algorithmus geschätzt. Es ergeben sich die folgenden Schätzungen für die drei Totzeiten:

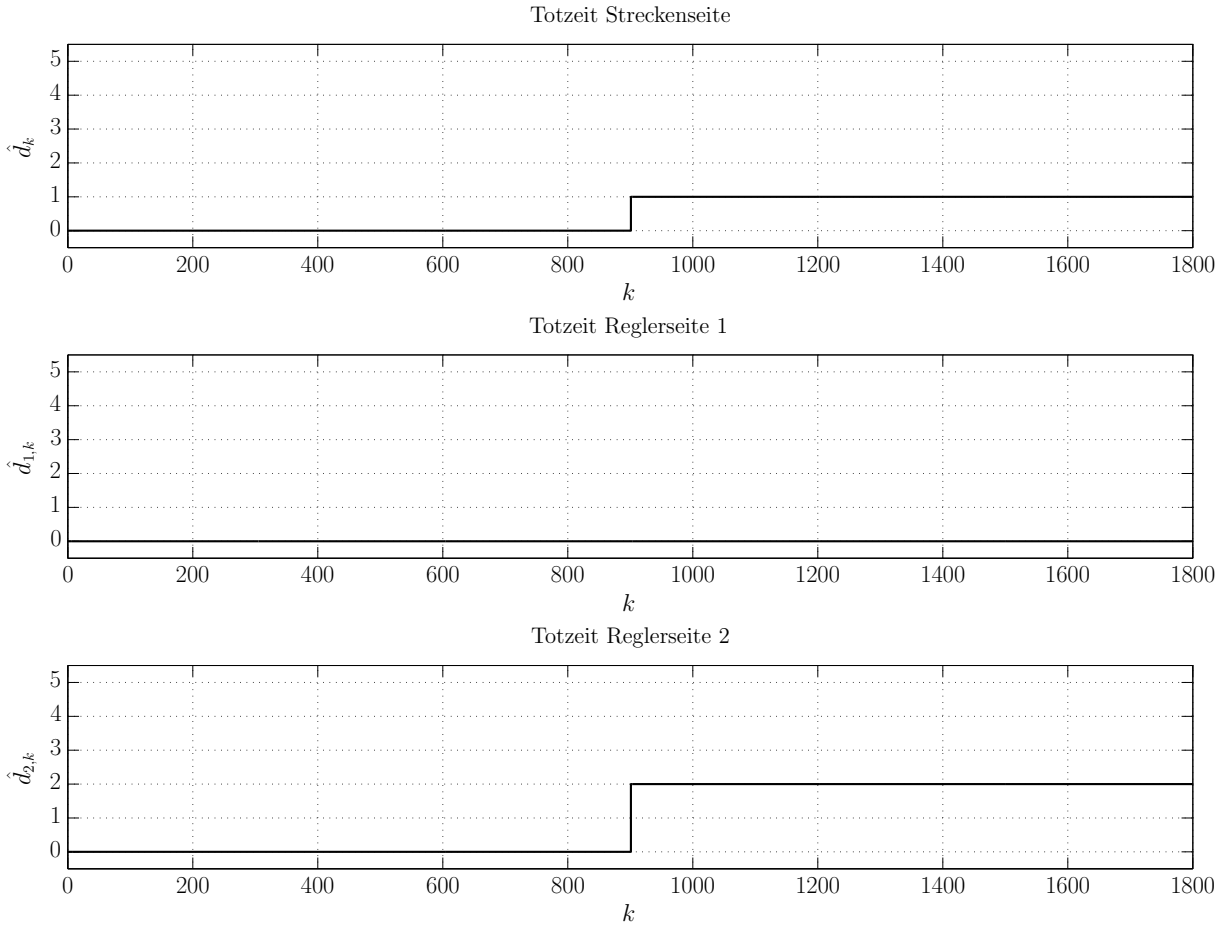

Abbildung 62: Totzeitschätzungen für das erste Anwendungsbeispiel

Man erkennt, dass sich nach den Einschwingvorgängen und der Optimierungszeit (insgesamt 900 Samples) konstant geschätzte Totzeiten ergeben:

$$\hat{d}_k = \hat{d} = 1$$

$$\hat{d}_{1,k} = \hat{d}_1 = 0 \tag{H.1.1}$$

$$\hat{d}_{2,k} = \hat{d}_2 = 2$$

Ein Abgleich mit dem Simulationsmodell in ICOS erbrachte das Ergebnis, dass alle diese drei Totzeiten richtig geschätzt werden. Die Implementierung der Totzeitschätzung funktioniert für dieses erste Anwendungsbeispiel also augenscheinlich korrekt.

2 Labormodell „Tank“

Das zweite Anwendungsbeispiel beschäftigt sich mit einem praktischen Anwendungsfall der implementierten Totzeitschätzung. Dabei wird ein quaderförmiger Tank betrachtet, welcher über eine Pumpe mit Wasser gefüllt werden kann. Zudem befindet sich am Boden des Tanks ein Loch, durch welches fortwährend Wasser abfließt. Die angesprochene Pumpe befindet sich ca. einen Meter unterhalb des Tanks und die Befüllung geschieht mit Hilfe eines Schlauchs, der das Wasser von oben in den Tank leitet. Als Eingangsgröße dient die Pumpenspannung v_k, während die Füllhöhe h_k im Tank die Ausgangsgröße darstellt. Zudem wird ein hier nicht weiter betrachteter Regler eingesetzt, um die Füllhöhe auf einen gewünschten Wert einzuregeln. Es wurde nun ein Experiment durchgeführt, bei welchem eine sprungförmige Referenz vorgegeben wurde. Da in diesem Beispiel das Augenmerk auf der durch die Schlauchlänge entstehenden Totzeit und nicht auf Totzeiten auf der Reglerseite liegt, wird nur das SIMULINK-Subsystem zur Totzeitschätzung auf der Streckenseite verwendet. Die vom Algorithmus benötigten Signale sind $u_{s,k} = v_k$ und $y_{r,k} = h_k$. Abbildung 63 zeigt diese:

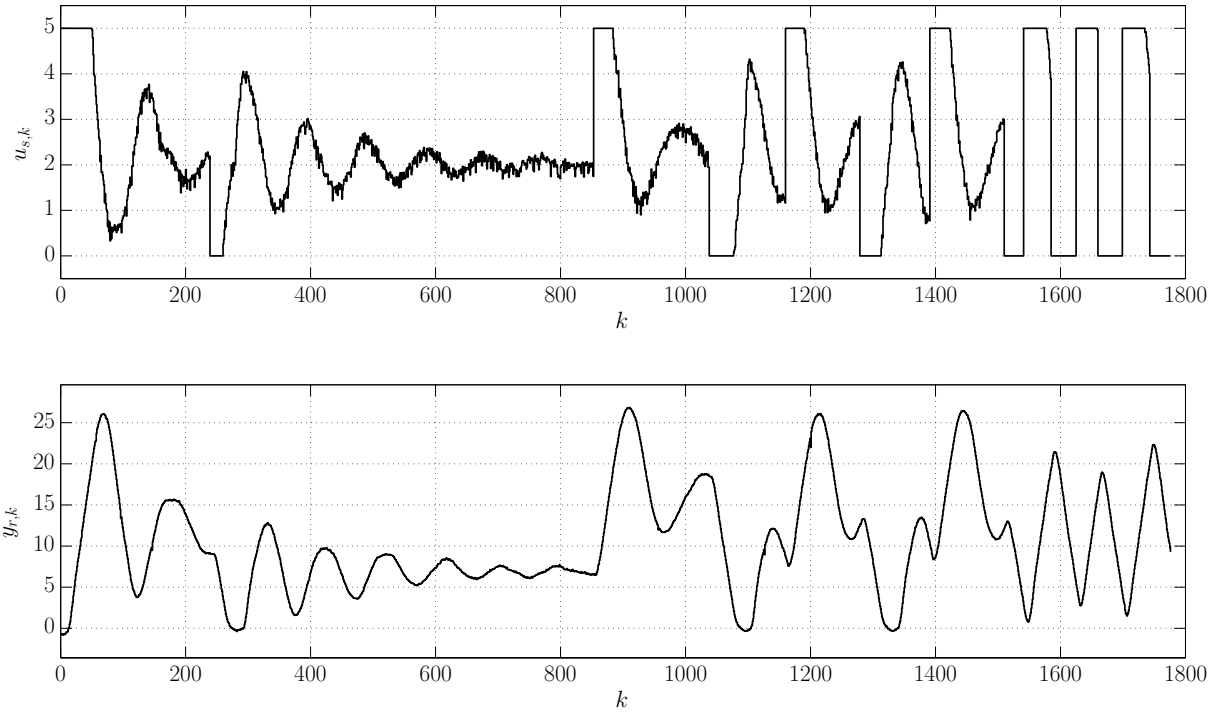

Abbildung 63: Signale für das zweite Anwendungsbeispiel

Diese beiden Signale werden wiederum über *From Workspace*-Blöcke zum Subsystem geführt, während die geschätzte Totzeit auch hier mit Hilfe eines *To Workspace*-Blocks gespeichert wird. Zudem wird die Totzeitschätzung auch hier dauerhaft aktiviert. Der zugehörige SIMULINK-Koppelplan ist in folgender Abbildung zu sehen:

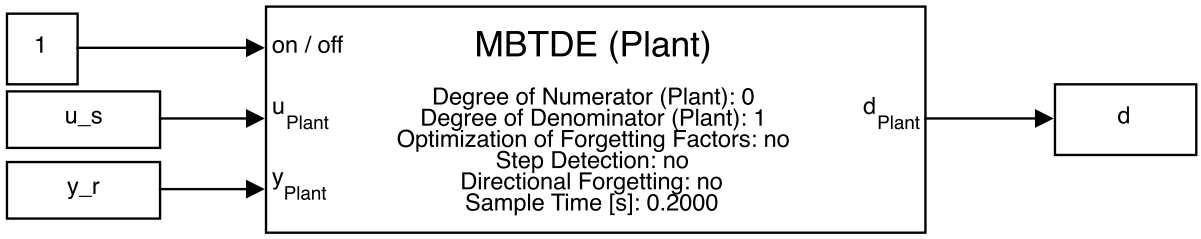

Abbildung 64: Simulink-Koppelplan für das zweite Anwendungsbeispiel

Aufgrund der in Abbildung 63 zu sehenden Signale kann man darauf schließen, dass sich das eigentlich nichtlineare dynamische Verhalten am ehesten durch ein lineares System erster Ordnung approximieren lässt. Daher wurde der Zählergrad mit $m_P = 0$ und der Nennergrad mit $n_P = 1$ gewählt. Um der Nichtlinearität des dynamischen Verhaltens folgen zu können, wurde ein Vergessensfaktor $\lambda_1 = 0.95$ des Systems festgelegt. Außerdem wurde der zweite Vergessensfaktor aufgrund einer konstant angenommenen Totzeit auf $\lambda_2 = 1$ gesetzt. Das direktionale Vergessen und die Sprungerkennung wurden nicht eingesetzt. Die weiteren in der Maske eingestellten Parameter lauten:

$$d_{\min} = 0 \; ; \; d_{\max} = 10 \; ; \; T_1 = 10 \; s \; ; \; T_2 = 10 \; s \; ; \; T_d = 0.2 \text{ s} \tag{H.2.1}$$

In Abbildung 65 ist das Ergebnis der Totzeitschätzung zu sehen:

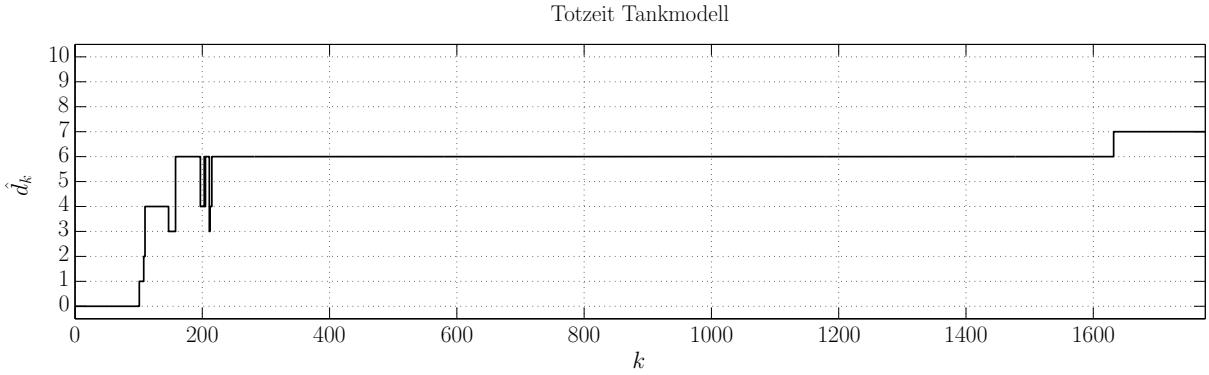

Abbildung 65: Totzeitschätzung für das zweite Anwendungsbeispiel

Man erkennt, dass sich eine geschätzte Totzeit einstellt, die zunächst den Wert $\hat{d}_k = 6$ und gegen Ende den Wert $\hat{d}_k = 7$ annimmt. Dies könnte darin begründet liegen, dass die wahre Totzeit τ des zeitkontinuierlichen Systems einem Wert entspricht, der zwischen $6 \cdot T_d$ und $7 \cdot T_d$ liegt. Weitergehende Experimente, bei welchen auch Einfüll- und Ausleerversuche ausgehend von stationären Zuständen durchgeführt wurden, bestätigten dieses Ergebnis. Die wahre Totzeit liegt tatsächlich in diesem Bereich. Auch hier liefert die Implementierung also eine solide Schätzung der Totzeit.

I Fazit und Ausblick

In dieser Arbeit wurde eine Methode gefunden welche das Potential besitzt, die Anforderungen der Zielsetzung dieser Arbeit zu erfüllen: der Elnaggar-Algorithmus. Dieser Algorithmus ist dazu fähig, anhand des Eingangs- und Ausgangssignals eines Systems auf eine Totzeit zu schließen. Es besteht dabei keine Beschränkung bezüglich der Art des Eingangssignals und der Charakteristik des Systems. Zudem können die auftretenden Totzeiten auch zeitlich variabel sein.

Der Elnaggar-Algorithmus konnte mit Hilfe kleiner Anpassungen für den vorgesehenen Zweck adaptiert werden. Dieser Zweck besteht darin, in einem Standard-Regelkreis auftretende Totzeiten innerhalb eines Koppelelements (zwischen Strecke und Regler) zu schätzen. Dabei können sowohl auf Strecken- als auch auf Reglerseite Totzeiten auftreten. In der Anwendung des Algorithmus in seiner Grundform konnten jedoch einige Probleme festgestellt werden. Diese Probleme konnte durch gezielte Maßnahmen größtenteils behoben werden. Die erste dieser Maßnahmen war die Berücksichtigung von Einschwingvorgängen bei den geschätzten Modellparametern. Dadurch wird das Verharren des Algorithmus auf einer falsch geschätzten Totzeit aufgrund falsch geschätzter Koeffizienten verhindert. Des weiteren wurde eine Optimierungsmethode für die beiden Vergessensfaktoren des Algorithmus vorgestellt. Diese ermöglicht es, dass die Vergessensfaktoren bei fehlendem Wissen über das zeitliche Verhalten von Modellparametern bzw. Totzeit vom Algorithmus automatisch festgelegt werden. Dem Anwender bzw. der Anwenderin wird dadurch die oft schwierige Frage nach einer geeigneten Wahl der Vergessensfaktoren abgenommen. Außerdem wurde eine alternative Vergessensstrategie zur Verringerung der Rauschsensibilität erläutert. Diese behebt das Problem des Verlernens der geschätzten Modellparameter beim Auftreten von Ausgangsrauschen und gleichzeitig unzureichender Anregung. Dieser Effekt kann zu falsch geschätzten Totzeiten führen und wird durch das direktionale Vergessen erheblich verringert. Die letzte Maßnahme zielt darauf ab, bei geregelten Systemen mit nur sporadisch auftretender sprungförmiger Anregung die Totzeitschätzung auf die informationshaltigen Zeitschritte (unmittelbar bei Auftreten der Sprünge) zu beschränken. Dies verhindert die Verfälschung der Schätzung durch Ausgangsrauschen in stationären Zuständen des Systems.

Es ist anzumerken, dass die entwickelte Methodik hauptsächlich zur Schätzung von Totzeiten bei linearen Systemen gedacht ist. Die Schätzung bei nichtlinearen Systemen kann jedoch auch gute Ergebnisse liefern, sofern sich das System durch eine lineare Approximation ausreichend gut abbilden lässt. In Bezug auf nichtlineare Systeme stellt insbesondere auch die Sprungerkennung ein hilfreiches Mittel zur Totzeitschätzung dar. Dies liegt darin begründet, dass sich nichtlineare Systeme bei sprungförmiger Anregung unmittelbar nach Auftreten des Sprunges in den allermeisten Fällen

sehr ähnlich zu linearen Systemen erster oder zweiter Ordnung Verhalten. Dies entspricht genau demjenigen Zeitraum, während dem die Gütefunktionen für die Totzeitschätzung bei aktivierter Sprungerkennung ausgewertet werden. Liegt ein stark nichtlineares Verhalten vor und ist das Anregungssignal zudem nicht sprungfähig, dann stößt die adaptierte Methode jedoch schnell an ihre Grenzen. Dies konnte vor allem bei nichtlinearen Regelalgorithmen (z.B. Eingangs-/Ausgangslinearisierung) festgestellt werden, wo die Einwirkung von zwei Eingängen als weitere Schwierigkeit hinzukommt.

Was bei genauerer Analyse des Algorithmus außerdem auffällt, ist die starke Wechselwirkung zwischen Schätzung der Modellparameter und Schätzung der Totzeit. Beide Schätzungen sind voneinander abhängig, da die Totzeitschätzung den Messvektor für die RLS-Identifizierung der Koeffizienten beeinflusst, während die geschätzten Koeffizienten eine Auswirkung auf den Modellfehler in den Totzeit-Gütefunktionen haben. Keine dieser beiden Schätzungen wird bevorzugt, es findet also eine gleichberechtigte Schätzung statt. Ruft man sich jedoch die Zielsetzung dieser Arbeit ins Gedächtnis, so fällt auf, dass hier einzig eine korrekte Totzeitschätzung gefordert war. Eine weiterführende Idee zur Verbesserung der Methode würde also darin bestehen, den Algorithmus so zu modifizieren, dass die Totzeitschätzung ein stärkeres Gewicht erhält, also den wichtigeren Teil gegenüber der Schätzung der Koeffizienten einnimmt. Beachtet werden muss jedoch, dass eine funktionierende Schätzung der Koeffizienten auch nicht ganz außer Acht gelassen werden darf, da sie wie erwähnt einen Einfluss auf die Totzeitschätzung selbst hat.

Literaturverzeichnis

[1] M. Horn & N. Dourdoumas, *Regelungstechnik - Rechnerunterstützter Entwurf zeitkontinuierlicher und zeitdiskreter Regelkreise*. München, Deutschland: Pearson Studium, 2004.

[2] H. Padé, „Sur la représentation approchée d'une fonction par des fractions rationnelles", *Annales scientifiques de l'École Normale Supérieure*, Bd. 3, H. 9, S. 3-93, 1892.

[3] O.J.M. Smith, „Closer control of loops with dead time", *Chemical Engineering Progress*, Bd. 53, H. 3, S. 217-219, 1957.

[4] J.E. Normey-Rico & E.F. Camacho, *Control of Dead-Time Processes*. London, England: Springer-Verlag London Limited, 2007.

[5] S. Cao & R.R. Rhinehart, „An efficient method for on-line identification of steady state", *Journal of Process Control*, Bd. 5, H. 6, S. 363-374, 1995.

[6] C.H. Knapp & G.C. Carter, „The Generalized Correlation Method for Estimation of Time Delay", *IEEE Transactions on Acoustics, Speech and Signal Processing* , Bd. 24, H. 4, S. 320-327, 1976.

[7] D. C. von Grünigen, *Digitale Signalverarbeitung*. Egg, Schweiz: FO Publishing, 2008.

[8] G. Moschytz & M. Hofbauer, *Adaptive Filter - eine Einführung in die Theorie mit Aufgaben und MATLAB-Simulationen auf CD-ROM*. New York, Vereinigte Staaten: Springer-Verlag Berlin Heidelberg, 2000.

[9] D.G. Luenberger, *Linear and Nonlinear Programming*. New York, Vereinigte Staaten: Springer Science+Business Media, 2008.

[10] A. Elnaggar, G. Dumont & A.-L. Elshafei, „Delay Estimation Using Variable Regression", *American Control Conference*, Boston, 1991, S. 2812-2817.

[11] A. Elnaggar, G. Dumont & A.-L. Elshafei, „New Method for Delay Estimation", *Proceedings of the 29th Conference on Decision and Control*, Honolulu, 1990, S. 1629-1630.

[12] M. Isaksson, *A Comparison of some approaches to Time-Delay-Estimation*. Lund Institute of Technology, Schweden, Masterarbeit 1997.

[13] S. Bedoui, M. Ltaief & K. Abderrahim, „New Results on Discrete-time Delay Systems Identification", *International Journal of Automation and Computing*, Bd. 9, H. 6, S. 570-577, 2012.

[14] L. Ljung, „Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems", *IEEE Transactions on Automatic Control*, Bd. 24, H. 1, S. 36-50, 1979.

[15] T. Söderström, L. Ljung & I. Gustavsson, „A theoretical analysis of recursive

identification methods", *Automatica*, Bd. 14, H. 3, S. 231-244, 1978.

[16] L. Ljung, *System Identification - Theory for the User*. New York, Vereinigte Staaten: Prentice Hall PTR, 2009.

[17] M.A. Woodbury, *The stability of out-input Matrices*. Chicago, Vereinigte Staaten: University of Chicago Press, 1949.

[18] K.Y. Wong & M.M. Bayoumi, „A self-tuning control algorithm for systems with unknown time delay", *Proceedings of the IFAC Identification and System Parameter Estimation Conference* , Washington D.C., 1982, S. 1193-1198.

[19] J. Roe, G. Ruiyao & A. O'Dwyer, „Identification of a time-delayed process model using an overparameterisation method", *Proceedings of the China-Ireland International Conference on Information and Communications Technologies*, Dublin, 2007.

[20] R. De Keyser, „Adaptive Dead-Time Estimation", *IFAC Adaptive Systems in Control and Signal Processings*, Lund, 1986, S. 385-389.

[21] A. Elnaggar, G. Dumont & A.-L. Elshafei, „Recusrsive Estimation for System of unknown Delay", *Proceedings of the 28th Conference on Decision and Control*, Tampa, 1989, S. 1809-1810.

[22] A. Elnaggar, G. Dumont & A.-L. Elshafei, „System Identification and Adaptive control Based on a Variable Regression for Systems Having Unknown Delay", *Proceedings of the 29th Conference on Decision and Control*, Honolulu, 1990, S. 1445-1450.

[23] S. Björklund & L. Ljung, *A Review of Time-Delay Estimation Techniques*. Universität Linköping, Schweden, Bericht 2003.

[24] S. Bedoui, M. Ltaief & K. Abderrahim, „A New Generalized Vector Observation for Discrete-Time Delay Systems Identification", *European Control Conference*, Zürich, 2013, S. 1922-1927.

[25] S. Bedoui, M. Ltaief & K. Abderrahim, „Nonlinear Approach For The Identification of Discrete Time Delay Systems", *20th Mediterranean Conference on Control & Automation*, Barcelona, 2012, S. 36-41.

[26] S. Bedoui, M. Ltaief & K. Abderrahim, *Online Identification of Multivariable Discrete Time Delay Systems Using a Recursive Least Square Algorithm*. Universität Gabes, Tunesien, Bericht 2013.

[27] G. Stettinger, M. Benedikt, M. Horn & J. Zehetner, „Modellbasierte Echtzeit-Co-Simulation: Überblick und praktische Anwendungsbeispiele", *e & i Elektrotechnik und Informationstechnik*, Bd. 132, H. 3, 2015.

[28] L. Ljung & T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, Vereinigte Staaten: MIT Press, 1983.

[29] J.J. Milek, *Stabilized Adaptive Forgetting in Recursive Parameter Estimation*. Eidgenössische Technische Hochschule Zürich, Schweiz, Dissertation 1995.

[30] F.J. Kraus, *Das Vergessen in rekursiven Parameterschätzverfahren*.

Eidgenössische Technische Hochschule Zürich, Zürich, Dissertation 1986.

[31] R. Kulhávy & M. Kárny, „Tracking of slowly varying Parameters by Directional Forgetting", *Proceedings of the ninth IFAC World Congress*, Budapest, 1984, S. 687-692.

[32] T. Hägglund, *New Estimation Techniques for Adaptive Control*. Universität Lund, Schweden, Dissertation 1983.

[33] R. Kulhávy, „Restricted Exponential Forgetting in Real-time Identification", *Automatica*, Bd. 23, H. 5, S. 589-600, 1987.

[34] V. Bobál, J. Böhm, J. Fessl & J. Machácek, *Digital Self-Tuning Controllers*. London, England: Springer-Verlag London Limited, 2005.

[35] S. Bittanti, P. Bolzern & M.C. Campi, „Convergence and Exponential Convergence of Identification Algorithms with Directional Forgetting Factor", *Automatica*, Bd. 26, H. 5, S. 929-932, 1990.

[36] L. Cao & H. Schwartz, „A directional forgetting algorithm based on the decomposition of the information matrix", *Automatica*, Bd. 36, H. 11, S. 1725-1731, 2000.

[37] A. Horch, *Condition Monitoring of Control Loops*. Royal Institute of Technology Stockholm, Schweden, Dissertation 2000.

[38] A.J. Isaksson, A. Horch & G.A. Dumont, „Event-Triggered deadtime estimation from closed-loop Data", *Proceedings of the American Control Conference*, Arlington, 2001, S. 3280-3285.

Eidesstattliche Erklärung

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

Graz, am ....

(Unterschrift)