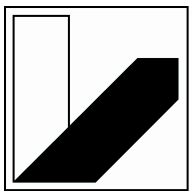

UNIVERSITÄT BAYREUTH

FAKULTÄT FÜR MATHEMATIK, PHYSIK UND INFORMATIK

MATHEMATISCHES INSTITUT

Identifikationsbasierte Beobachter auf beweglichem Horizont

Bachelorarbeit

von

Simon Pirkelmann

Datum:
20. April 2014

Aufgabenstellung und Betreuung:
Prof. Dr. Lars Grüne
Dipl.-Math. Thomas Jahn

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Die Kontrolltheorie beschäftigt sich mit dynamischen Systemen, auf deren Zustand durch sogenannte Eingangsgrößen bzw. Kontrollen Einfluss genommen werden kann. Das reicht vom einfachen Beispiel einer Heizungsregelung, um eine bestimmte Raumtemperatur zu erreichen, bis zum komplexen System eines Stromnetzes, bei dem die Verbraucher sowohl Energie aus dem Netz entnehmen als auch zuführen können z. B. durch Photovoltaikanlagen auf dem Dach. Hier muss durch geeignete Steuerung der Ladung von Zwischenspeichern (Batterien) für eine möglichst konstante Netzauslastung gesorgt werden.

Zum Erreichen eines gewünschten Systemzustands gibt es dabei verschiedene Methoden, wie beispielsweise Feedbackregelung oder Modellprädiktive Regelung (MPC). [5, 7]

Alle diese Techniken haben gemeinsam, dass für die Berechnung einer Kontrolle Kenntnis über den aktuellen Systemzustand vorausgesetzt ist. Das lässt sich auch ganz intuitiv erklären: Um ein Ziel zu erreichen, muss man natürlich erst einmal wissen, wo man sich gerade befindet.

Das Problem dabei ist jedoch, dass die messbaren Variablen oft nur einen kleinen Anteil der Variablen bilden, die zur Modellierung eines Systems nötig sind. Der vollständige Zustand ist also im Allgemeinen nicht bekannt. An dieser Stelle setzt ein Beobachter an, der aus dem gemessenen, unvollständigen Systemzustand eine Approximation des tatsächlichen Zustands erstellt.

Im Fall linearer Kontrollsysteme ist es relativ einfach einen Beobachter zu finden. Der Luenberger Beobachter ist unter denkbar einfachen Voraussetzungen anwendbar, notwendig ist lediglich die asymptotische Beobachtbarkeit des Systems. Die Konstruktion des Beobachters erfolgt dabei so, dass der Beobachterfehler durch eine lineare exponentiell stabile Differentialgleichung beschrieben wird. Dadurch konvergiert der Beobachter auch für sehr schlechte Startschätzungen gegen den tatsächlichen Zustand und der geschätzte Zustand wird mit der Zeit immer genauer. Außerdem reagiert der Beobachter gut auf Ungenauigkeiten und Störungen des Systems. Mit dem Einsatz des Kalman Filters ist hier sogar eine optimale Zustandsschätzung möglich. [5]

Auch bei nichtlinearen Kontrollsystemen gibt es Möglichkeiten die Zustandsschätzung durchzuführen. Das Konzept des Kalman Filters kann mittels Linearisierung auf ein nichtlineares System ausgedehnt werden. Man spricht dann vom erweiterten Kalman Filter (EKF), der sich als De-facto-Standard in vielen technischen Anwendungen etabliert hat. Im Unterschied zu seinem linearen Gegenstück liefert der erweiterte Kalman Filter aber im Allgemeinen keine optimale Zustandsschätzung. [1]

1.2 Ziel der Arbeit

Tatsächlich gibt es Fälle, in denen der erweiterte Kalman Filter keine brauchbaren Ergebnisse erzielt. Betrachten wir beispielsweise das System

$$\frac{d}{dt}\begin{pmatrix} c_A \\ c_B \\ c_C \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 1 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} k_1 c_A - k_{-1} c_B c_C \\ k_2 c_B^2 - k_{-2} c_C \end{pmatrix}$$

mit Ausgang

$$y = RT \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} c_A \\ c_B \\ c_C \end{pmatrix},$$

durch das die Konzentration von Substanzen A, B und C in einem chemischen Batchreaktor beschrieben wird. [7]

In Abbildung 1.1 sind die beobachteten Konzentrationen zusammen mit ihren exakten Verläufen dargestellt. Es wurde bewusst eine schlechte Startschätzung für den Beobachter verwendet. Wie man erkennt, kann der erweiterte Kalman Filter den Zustand dieses Systems nicht rekonstruieren. Die Schätzung konvergiert gegen einen physikalisch unmöglichen Zustand mit negativen Konzentrationen für A und B.

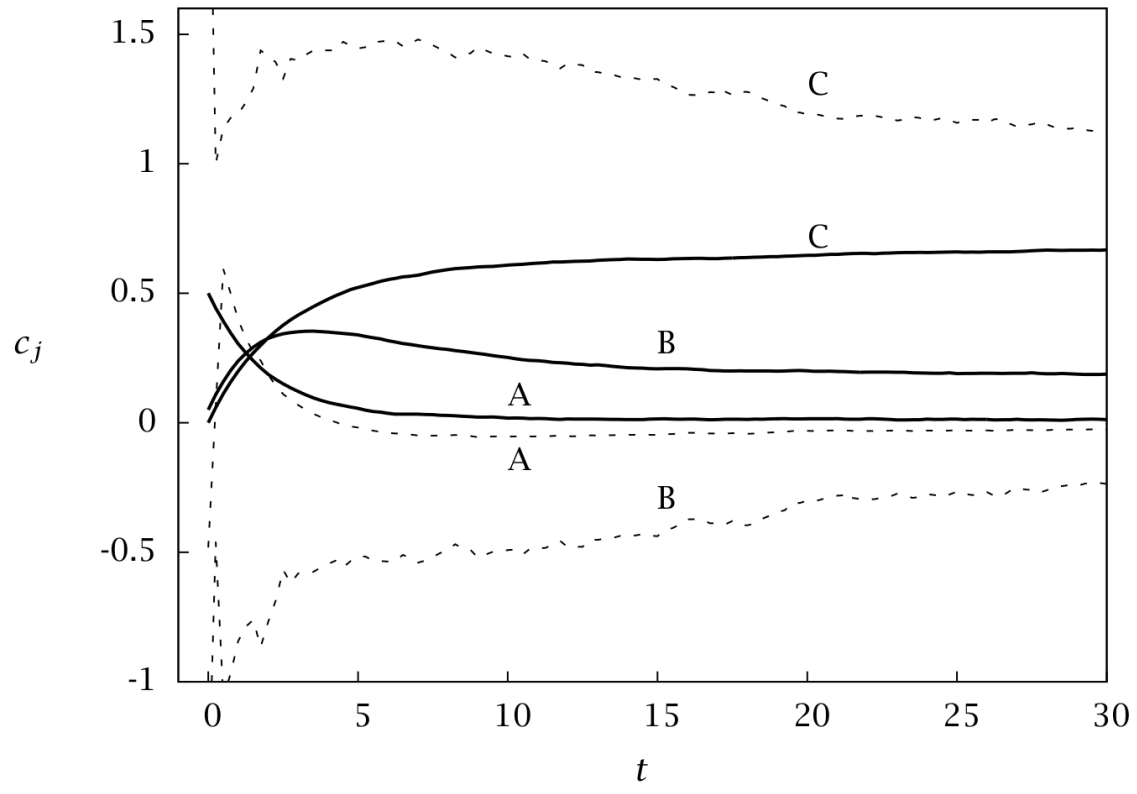

Abbildung 1.1: Verlauf des exakten Zustands (durchgezogene Linien) und der Schätzung des erweiterten Kalman Filters (gestrichelte Linien). (Quelle: [7])

Eine korrekte Zustandsschätzung erhält man hier durch den Einsatz von Moving Horizon Estimation (MHE). Bei diesem Ansatz wird der Schätzfehler mittels der Methode der kleinsten Quadrate minimiert. Dabei werden die letzten N Messungen in die Berechnung des Zustands mit einbezogen. [7]

Abbildung 1.2 zeigt das Ergebnis von MHE unter Berücksichtigung eines Fensters von 10 Messwerten. Offenbar erholt sich der Beobachter gut vom anfänglich schlechten Schätzwert.

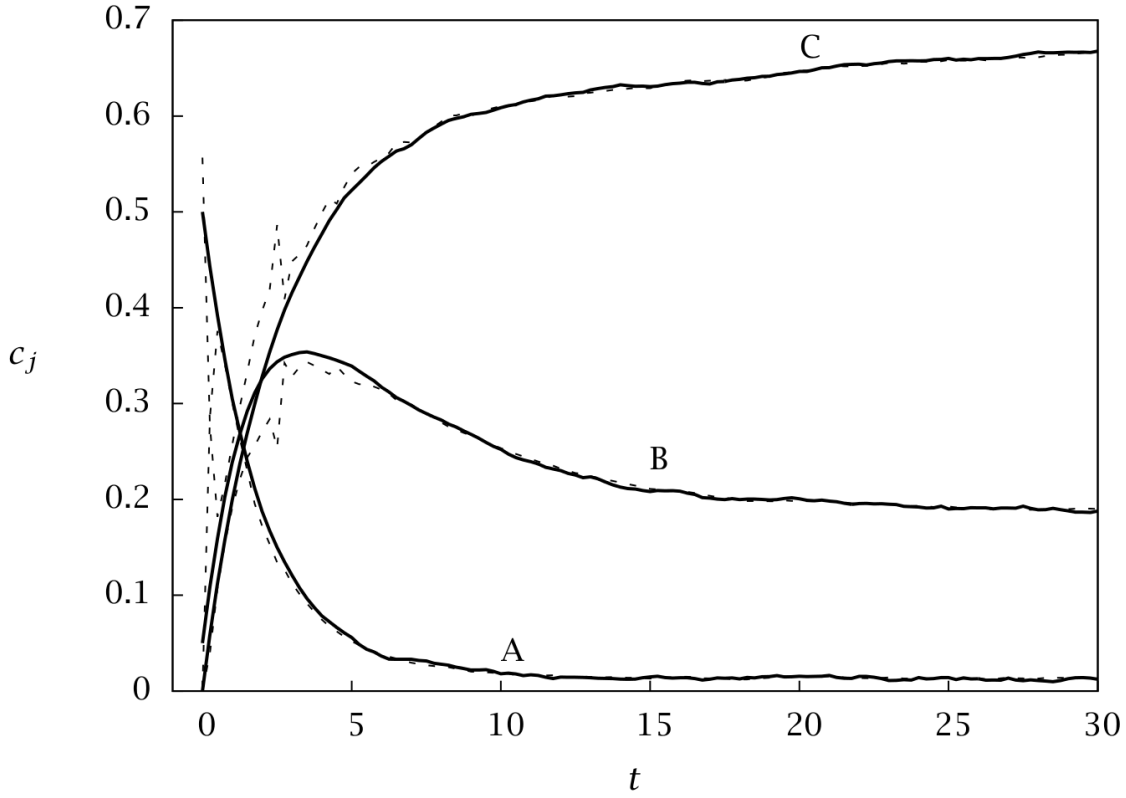

Abbildung 1.2: Verlauf des exakten Zustands (durchgezogene Linien) und der Schätzung mit MHE (gestrichelte Linien). (Quelle: [7])

Die Zustandsrekonstruktion mittels MHE besitzt jedoch den Nachteil, dass in jedem Zeitschritt ein nichtkonvexes Optimierungsproblem gelöst werden muss. [2]

Beim Einsatz in Systemen mit hohen Abtastraten oder limitierten Rechenkapazitäten kann das zu Problemen führen. Es stellt sich daher die Frage, ob es möglich ist, einen Beobachter zu entwickeln, der, wie bei MHE, eine Vergangenheit von N Messwerten verwendet, jedoch in der Anwendung ohne die aufwendige Lösung eines Optimierungsproblems auskommt.

In dieser Arbeit soll nun ein Ansatz vorgestellt werden, der dies verspricht. Statt fortwährender Optimierung wird dabei offline eine Funktion identifiziert, mit der eine Reihe von Messungen auf den vollständigen Zustand abgebildet wird. Diese Funktion kann dann als Beobachter auf beweglichem Horizont eingesetzt werden.

1.3 Aufbau

Im ersten Teil der Arbeit beschäftigen wir uns mit der grundlegenden Theorie des Beobachters auf beweglichem Horizont. Außerdem wird auf die algorithmische Umsetzung eingegangen.

Der zweite Teil orientiert sich an einer Diplomarbeit über das inverse Pendel auf einem Wagen (siehe [3]). Darin wurde ein präziseres mathematisches Modell des Pendels eingeführt und darauf aufbauend unter anderem die Implementierung eines Beobachters vorgeschlagen. Wir werden die Ergebnisse des ersten Teils hier anwenden, um einen Beobachter zu konstruieren. Die Genauigkeit der Zustandsschätzung wird anhand von Simulationen und praktischem Einsatz überprüft.

Anschließend werden die erzielten Resultate diskutiert und auf Vorteile und Schwächen des Beobachters eingegangen.

Zum Schluss folgt eine kurze Zusammenfassung und es werden einige weitergehende Ideen vorgestellt.

2 Beobachter auf beweglichem Horizont

Der hier betrachtete Ansatz zur Konstruktion eines Beobachters nutzt Systeminformationen aus der Vergangenheit für die Zustandsschätzung. Die Zeitspanne, aus der Daten in die Schätzung einfließen, bezeichnen wir als Horizont. Da sich mit fortschreitender Zeit auch der Horizont verschiebt, spricht man vom Beobachter auf beweglichem Horizont.

2.1 Idee

Es seien im Folgenden stets $n, n_y \in \mathbb{N}, n_u \in \mathbb{N}_0$ und $f : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^n$ ein Vektorfeld, sowie eine Funktion $C : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$. Wir betrachten das nichtlineare zeitdiskrete System definiert durch:

$$x(k+1) = f(x(k), u(k)) \tag{2.1}$$

mit Ausgang

$$y(k) = C(x(k), u(k)). \tag{2.2}$$

Dabei bezeichnet $x \in \mathbb{R}^n$ den Zustand, $u \in \mathbb{R}^{n_u}$ die Kontrolle und $y \in \mathbb{R}^{n_y}$ den gemessenen Ausgang des Systems. Die Zahl k beschreibt den Zeitpunkt hk für die Abtastrate $h > 0$.

Die grundsätzliche Annahme ist, dass eine Horizontlänge $N \in \mathbb{N}$ und eine Funktion $\mathcal{F} : \mathbb{R}^{N(n_y+n_u)} \to \mathbb{R}^n$ existieren, so dass aus dem Horizont $Z(k)$ der letzten N gemessenen Zustände und Kontrollen der tatsächlichen Zustand des Systems rekonstruiert werden kann, dass also gilt:

$$x(k) \approx \mathcal{F}(Z(k))$$

Die Historie $Z(k)$ hat dabei die Form:

$$Z(k) = \begin{pmatrix} y(k) \\ u(k) \\ y(k-1) \\ u(k-1) \\ \vdots \\ y(k-N+1) \\ u(k-N+1) \end{pmatrix} \in \mathbb{R}^{N(n_y+n_u)=:n_z}$$

Eine skizzenhafte Darstellung des Horizonts $Z(k)$ und des daraus geschätzten Zustandes $x(k)$ ist in Abbildung 2.1 gegeben.

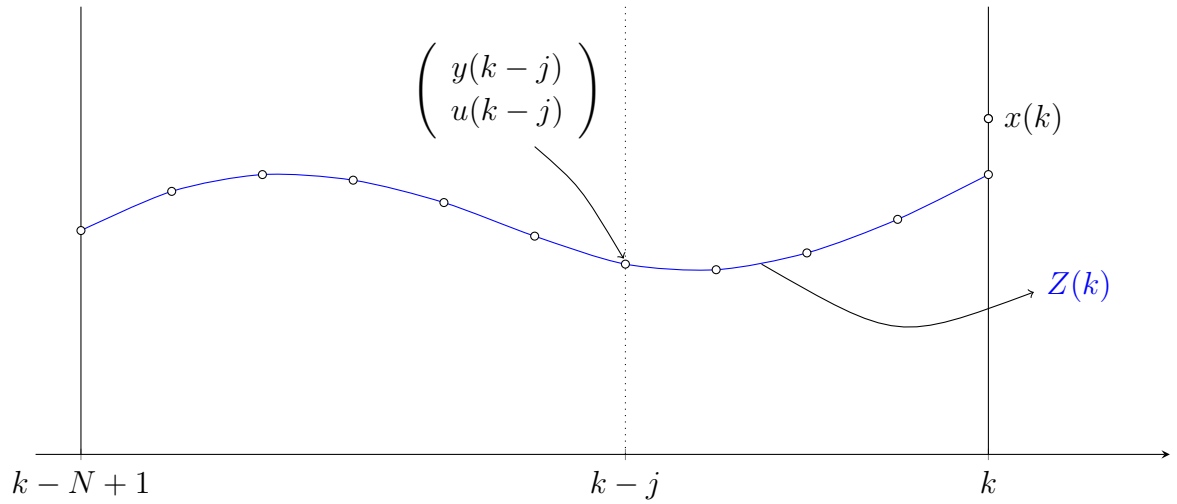

Abbildung 2.1: Zusammensetzung des Horizonts und geschätzter Zustand $x(k)$.

Ziel ist es nun, die Funktion $\mathcal{F}$ approximativ zu bestimmen, um damit einen Beobachter des Systems basierend auf dem Horizont $Z(k)$ zu erhalten. Man spricht hier auch von Identifikation der Funktion $\mathcal{F}$.

Zur Lösung dieses Problems stellen wir zunächst fest, dass es genügt, für jede Komponente $r_i = x_i(k)$ des Zustandes $x(k)$ eine Funktion F_i zu identifizieren, die den Horizont $Z(k)$ auf den Zustand r_i abbildet. Der Beobachter für den gesamten Zustand ergibt sich dann durch:

$$x(k) \approx \mathcal{F}(Z(k)) = \begin{pmatrix} F_1(Z(k)) \\ \vdots \\ F_n(Z(k)) \end{pmatrix}$$

Wir beschäftigen uns also mit dem Problem, eine Funktion $F : \mathbb{R}^{n_z} \to \mathbb{R}$ zu finden, die einen Zusammenhang zwischen r (einer Komponente des Zustandsvektors) und dem Horizont Z herstellt:

$$r \approx F(Z) \quad , \quad r \in \mathbb{R} \quad , \quad Z \in \mathbb{R}^{n_z} \tag{2.3}$$

Speziell wird der Fall betrachtet, dass F sich darstellen lässt durch:

$$F(Z) = \Gamma^{-1}(Z^T L), \quad \Gamma(\cdot) \text{ streng monoton wachsend}, \quad L \in \mathbb{R}^{n_z} \tag{2.4}$$

Die Existenz der Funktion Γ^{-1} ist aufgrund der Monotonie von Γ garantiert.

Diese Eigenschaft der Funktion F kann man sicher nicht für beliebige Systeme erwarten. Wir schränken uns auf Funktionen F ein, die diese Eigenschaft erfüllen, da sich diese Struktur gut für die effiziente Lösung mit einem quadratischen Optimierungsproblem (QP-Problem) eignet. Leider lässt sich a priori nicht sagen, ob ein System die Eigenschaft (2.4) erfüllt. Die Effizienz des QP-Problems ermöglicht es aber, eine große Menge von Parametern zu testen, um zu sehen, ob man eine brauchbare Lösung erhält.[1]

[1] vgl. Remark 2 in [2]

Im Fall, dass (2.4) erfüllt ist, erhält man zusammen mit (2.3):

$$\Gamma(r) \approx \Gamma(F(Z)) = Z^T L$$

F zu finden, ist unter den gemachten Annahmen also äquivalent dazu, einen Vektor $L \in \mathbb{R}^{n_z}$ und eine streng monoton wachsende Funktion Γ zu finden, so dass:

$$\Gamma(r) \approx Z^T L \tag{2.5}$$

Im nächsten Schritt approximieren wir die Funktion Γ mittels Basisfunktionen. Wir betrachten die Funktionenbasis B der Dimension $n_b = 2n_m,\ n_m \in \mathbb{N}$, die folgendermaßen definiert ist:

$$B(\eta) = \begin{pmatrix} 1 & B_1^{(2)}(\eta) & \dots & B_1^{(n_m)}(\eta) & B_2^{(1)}(\eta) & \dots & B_2^{(n_m)}(\eta) \end{pmatrix} \in \mathbb{R}^{1 \times n_b}$$

Hierbei sind $B_1^{(i)} : [0,1] \to [0,1]$ und $B_2^{(j)} : [0,1] \to [0,1]$ gegeben durch:

$$B_1^{(i)}(\eta) := (1+\alpha_i)\frac{\eta}{1+\alpha_i \eta} \quad , \quad i \in \{2, \dots, n_m\}$$
$$B_2^{(j)}(\eta) := \frac{\eta}{1+\alpha_j - \alpha_j \eta} \quad , \quad j \in \{1, \dots, n_m\}$$

mit $\alpha_l := exp(\beta(1-l)) - 1$ für ein $\beta \in \mathbb{R}\backslash\{0\}$.

Eine grafische Darstellung dieser Basisfunktionen für $n_b = 12$ und Parameter $\beta = \frac{5}{6}$ findet sich in Abbildung 2.2. Wie man sieht, sind die Basisfunktionen (mit Ausnahme der konstanten Funktion) alle streng monoton wachsend. Es liegt nahe, diese Funktionenbasis zu verwenden, da wir damit eine streng monoton wachsende Funktion darstellen wollen.

Um die Funktion $\Gamma(r)$ mit den Basisfunktionen ausdrücken zu können, müssen wir noch sicherstellen, dass sich r im betrachteten Definitionsbereich $[0,1]$ der Basisfunktionen befindet. Dies gewährleisten wir, indem r mit einer Funktion η geeignet transformiert wird. Seien dazu r_{min} und r_{max} der minimale bzw. maximale Wert, den r auf den Trainingsdaten annimmt[2]. Wir definieren:

$$\eta(r) := \frac{r - r_{min}}{r_{max} - r_{min}} \tag{2.6}$$

Nach diesen Vorüberlegungen können wir Γ als Linearkombination aus Basisfunktionen darstellen. Sei dazu $\mu \in \mathbb{R}^{n_b}$, so dass gilt:

$$\Gamma(r) = B(\eta(r))\mu = \sum_{j=1}^{n_b} \mu_j B^{(j)}(\eta(r)) \tag{2.7}$$

Kombinieren wir nun die Gleichungen (2.5) und (2.7) so erhalten wir:

$$Z^T L - B(\eta(r))\mu \approx 0 \tag{2.8}$$

[2]Wie man diese Werte erhält, werden wir im folgenden Abschnitt betrachten.

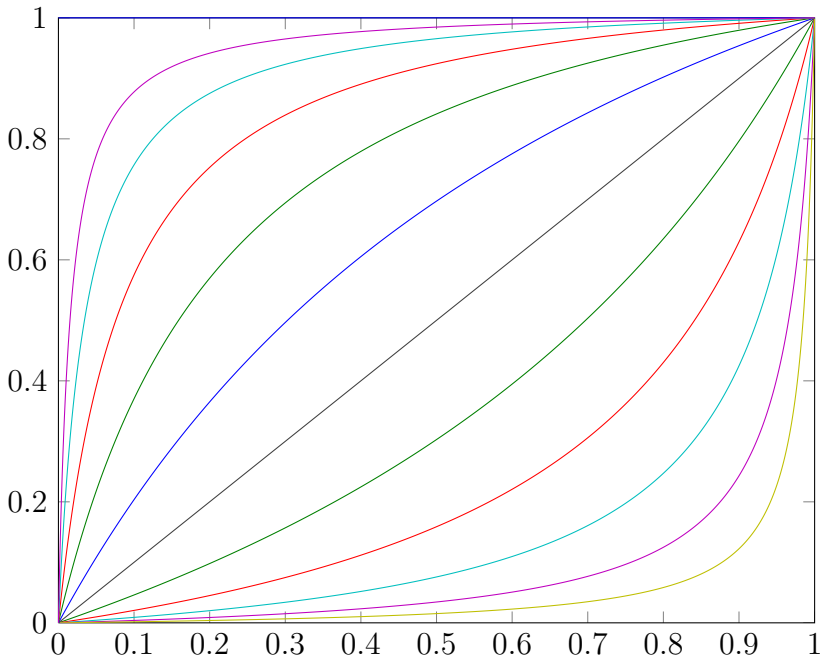

Abbildung 2.2: Darstellung der Basisfunktionen für eine Basisgröße von $n_b = 12$ und Parameter $\beta = \frac{5}{6}$.

Diese Gleichung bildet die Grundlage des Verfahrens zur Identifikation von F. Wir wollen für eine große Anzahl von Paaren (r, Z) aus einer Menge von Referenzdaten $\mathcal{E}$ die Vektoren L und μ so bestimmen, dass (2.8) möglichst gut erfüllt ist.

Wir erhalten daraus das Kleinste-Quadrate-Problem:

$$\min_{L,\mu} \sum_{(r,Z)\in\mathcal{E}} \Big(Z^T L - B(\eta(r))\mu \Big)^2 \tag{2.9}$$

Zusätzlich müssen folgende Nebenbedingungen gelten:

1) Γ muss streng monoton wachsend sein. Dies erreicht man, indem man die Ableitung mit einer Konstante $\varepsilon > 0$ nach unten beschränkt:

$$\forall \eta \in [0,1] \quad \Big[\frac{dB}{d\eta}(\eta)\Big]\mu \geq \varepsilon \tag{2.10}$$

2) Um die triviale Lösung $L = 0$, $\mu = 0$ auszuschließen, muss eine Normalisierungsbedingung[3] erfüllt sein:

$$\Big[\int_0^1 B(\eta)d\eta\Big]\mu = \frac{1}{2}\Big(r_{min} + r_{max}\Big) \tag{2.11}$$

Dabei bezeichnet

$$\frac{dB}{d\eta}(\eta) = \begin{pmatrix} 0 & \frac{d}{d\eta}B_1^{(2)}(\eta) & \dots & \frac{d}{d\eta}B_1^{(n_m)}(\eta) & \frac{d}{d\eta}B_2^{(1)}(\eta) & \dots & \frac{d}{d\eta}B_2^{(n_m)}(\eta) \end{pmatrix}$$

[3] vgl. auch Remark 3 in [2]

die Ableitung der einzelnen Basisfunktionen in jeder Komponente und analog

$$\int_0^1 B(\eta)d\eta = \begin{pmatrix} 1 & \int_0^1 B_1^{(2)}(\eta)d\eta & \dots & \int_0^1 B_1^{(n_m)}(\eta)d\eta & \int_0^1 B_2^{(1)}(\eta)d\eta & \dots & \int_0^1 B_2^{(n_m)}(\eta)d\eta \end{pmatrix}$$

das Integral der Basisfunktionen.

2.2 Umsetzung

In diesem Abschnitt werden wir die bisherigen Ergebnisse nutzen, um ein numerisches Verfahren zur Konstruktion des Beobachters herzuleiten.

2.2.1 Formulierung als quadratisches Optimierungsproblem

Für die numerische Umsetzung ist die bisherige Form des Problems noch nicht direkt geeignet. Wir überführen es deshalb zunächst in ein allgemeines quadratisches Optimierungsproblem (QP-Problem):

$$\begin{aligned} \min_x \quad & f(x) = \frac{1}{2}x^T Q x + c^T x \\ & Ax \leq b \\ & Ex = d \end{aligned} \tag{2.12}$$

Wegen

$$\begin{aligned} & \min_{L,\mu} \sum_{(r,Z)\in\mathcal{E}} \Big(Z^T L - B(\eta(r))\mu\Big)^2 \\ =& \min_{L,\mu} \sum_{(r,Z)\in\mathcal{E}} \left(\Big(Z^T, -B(\eta(r))\Big)\begin{pmatrix} L \\ \mu \end{pmatrix}\right)^2 \\ =& \min_{L,\mu} \sum_{(r,Z)\in\mathcal{E}} \Big(Z^T, -B(\eta(r))\Big)\begin{pmatrix} L \\ \mu \end{pmatrix}\Big(Z^T, -B(\eta(r))\Big)\begin{pmatrix} L \\ \mu \end{pmatrix} \\ =& \min_{L,\mu} \sum_{(r,Z)\in\mathcal{E}} \Big(L^T, \mu^T\Big)\begin{pmatrix} Z \\ -B(\eta(r))^T \end{pmatrix}\Big(Z^T, -B(\eta(r))\Big)\begin{pmatrix} L \\ \mu \end{pmatrix} \\ =& \min_{L,\mu} \Big(L^T, \mu^T\Big)\left[\sum_{(r,Z)\in\mathcal{E}} \begin{pmatrix} Z \\ -B(\eta(r))^T \end{pmatrix}\Big(Z^T, -B(\eta(r))\Big)\right]\begin{pmatrix} L \\ \mu \end{pmatrix} \end{aligned}$$

ist (2.9) mit $x := \begin{pmatrix} L \\ \mu \end{pmatrix} \in \mathbb{R}^{n_z+n_b}$ bereits eine Zielfunktion der gewünschten Form. Wir setzen:

$$Q := 2\sum_{(r,Z)\in\mathcal{E}} \begin{pmatrix} Z \\ -B(\eta(r))^T \end{pmatrix}\Big(Z^T, -B(\eta(r))\Big) \in \mathbb{R}^{(n_z+n_b)\times(n_z+n_b)} \tag{2.13}$$

Da keine linearen Anteile vorhanden sind, ist $c := 0$.

Bemerkung 1 (Konvexität des QP-Problems)
Beachte, dass die Matrix Q positiv semidefinit ist. Wir schreiben kurz $z^T = \Big(Z^T, -B(\eta(r))\Big)$, $\langle \cdot, \cdot \rangle$ bezeichne das Standardskalarprodukt. Für $x \in \mathbb{R}^{n_z+n_b}$ gilt dann:

$$\begin{aligned} x^T Q x &= 2 \sum_{(r,Z)\in\mathcal{E}} x^T z \; z^T x \\ &= 2 \sum_{(r,Z)\in\mathcal{E}} \langle x, z\rangle \, \langle z, x\rangle \\ &= 2 \sum_{(r,Z)\in\mathcal{E}} \langle x, z\rangle^2 \\ &\geq 0 \end{aligned}$$

Insbesondere ist die Zielfunktion f des QP-Problems konvex. Diese Klasse von Optimierungsproblem ist in der Schwierigkeit vergleichbar mit einem linearen Programm und es gibt eine Reihe von effizienten Lösungsverfahren. [6]

Weil es numerisch nicht möglich ist, die Nebenbedingung (2.10) für alle $\eta \in [0,1]$ zu überprüfen, betrachten wir stattdessen ein Gitter $0 = \eta_1 < \cdots < \eta_q = 1$, $q \in \mathbb{N}$. Man fordert nun, dass die Ableitung an jeder Stelle des Gitters größer als ein $\varepsilon > 0$ ist. Für ein hinreichend feines Gitter folgt so die Monotonie der Funktion Γ.
Es ergeben sich die linearen Ungleichungen:

$$\begin{pmatrix} \frac{dB}{d\eta}(\eta_1) \\ \vdots \\ \frac{dB}{d\eta}(\eta_q) \end{pmatrix} \mu \geq \begin{pmatrix} \varepsilon \\ \vdots \\ \varepsilon \end{pmatrix}$$

$$\Leftrightarrow \quad \begin{pmatrix} -\frac{dB}{d\eta}(\eta_1) \\ \vdots \\ -\frac{dB}{d\eta}(\eta_q) \end{pmatrix} \mu \leq \begin{pmatrix} -\varepsilon \\ \vdots \\ -\varepsilon \end{pmatrix} =: b \in \mathbb{R}^q$$

Der Vektor x enthält an erster Stelle die Komponenten des Vektors L, daher muss man in der Matrix A noch eine Nullmatrix der Dimension $q \times n_z$ voranstellen und erhält:

$$A := \begin{pmatrix} 0 & \cdots & 0 & -\frac{dB}{d\eta}(\eta_1) \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & -\frac{dB}{d\eta}(\eta_q) \end{pmatrix} \in \mathbb{R}^{q\times(n_z+n_b)}$$

Nebenbedingung (2.11) ist bereits eine lineare Gleichung, es ist also $d := \frac{1}{2}(r_{min}+r_{max}) \in \mathbb{R}$. Auffüllen mit einer Nullmatrix der Dimension $1 \times n_z$ liefert direkt:

$$E := \begin{pmatrix} 0 & \cdots & 0 & \int_0^1 B(\eta) d\eta \end{pmatrix} \in \mathbb{R}^{1\times(n_z+n_b)}$$

Das Integral jeder Basisfunktion kann dabei einmal im Voraus explizit ausgewertet werden und muss nicht während der Lösung des QP-Problems fortwährend berechnet werden.

Damit haben wir das Kleinste-Quadrate-Problem (2.9) mit Nebenbedingungen (2.10) und (2.11) in ein QP-Problem der Form (2.12) überführt.

Zur Lösung dieses Problems steht eine ganze Reihe von Techniken aus der mathematischen Optimierung zur Verfügung. Im MATLAB-Programm wird ein Innere-Punkte-Verfahren zur Lösung von konvexen Optimierungsproblemen verwendet. Eine ausführliche Beschreibung des verwendeten Algorithmus findet sich z. B. in [6].

2.2.2 Generierung der Lerndaten

Zum Aufstellen der Matrix Q aus (2.13) müssen wir noch klären, wie man die Lerndaten $\mathcal{E}$ erhält. Der Grundgedanke hier ist, das System (2.1) für eine Vielzahl von Startwerten und Kontrollfolgen zu simulieren, um so aus den Lösungstrajektorien Paare (Z, r) zu erhalten, die zum Training des neuronalen Netzes verwendet werden.

Betrachte dazu eine Menge von Startwerten

$$\mathbb{X} := \prod_{i=1}^{n} [x_i^{min}, x_i^{max}].$$

Analog schränken wir die betrachteten Kontrollwerte auf die Menge

$$\mathbb{U} := \prod_{i=1}^{n_u} [u_i^{min}, u_i^{max}]$$

ein. Wir wählen nun n_g Startwerte $x_0^{(j)} \in \mathbb{X}$ aus. Diese Auswahl kann man etwa durch ein gleichmäßiges Gitter über die Intervalle $[x_i^{min}, x_i^{max}]$ treffen. Außerdem wählt man für jeden Startwert eine Folge von Kontrollwerten $\tilde{u}^{(j)} \in \mathbb{U}^{N_s}$ für einen Zeitraum von $N_s \geq N$ Abtastperioden.

Man simuliert nun das System mit Anfangszeit $t_0 = 0$ für jedes Paar $(x_0^{(j)}, \tilde{u}^{(j)})$ über die Länge von N_s Abtastperioden und erhält so eine Menge von Lösungen:

$$\left\{\chi^{(j)}\right\}_{j=1}^{n_g} := \left\{x(k, t_0, x_0^{(j)}, \tilde{u}^{(j)}), k \in \{1, \ldots, N_s\}\right\}_{j=1}^{n_g}$$

In Abbildung 2.3 wird das Vorgehen für eine kleine Anzahl von erzeugten Lösungen illustriert.
Aus jeder generierten Lösung $\chi^{(j)}$ mit zugehöriger Kontrolle $\tilde{u}^{(j)}$ gewinnt man mittels der Funktion des Ausgangs (2.2) den Horizont $Z^{(j)}(k)$ und den damit verbundenen exakten Zustand $r^{(j)}(k) := x(k, t_0, x_0^{(j)}, \tilde{u}^{(j)})_{(p)}$ für jedes $k \in \{N, \ldots, N_s\}$. Der Index p verweist hierbei auf die Komponente des Zustandsvektors, die beobachtet werden soll.

Abbildung 2.4 zeigt, welche Teile der Lösung $\chi^{(1)}$ aus Abbildung 2.3 zu den einzelnen

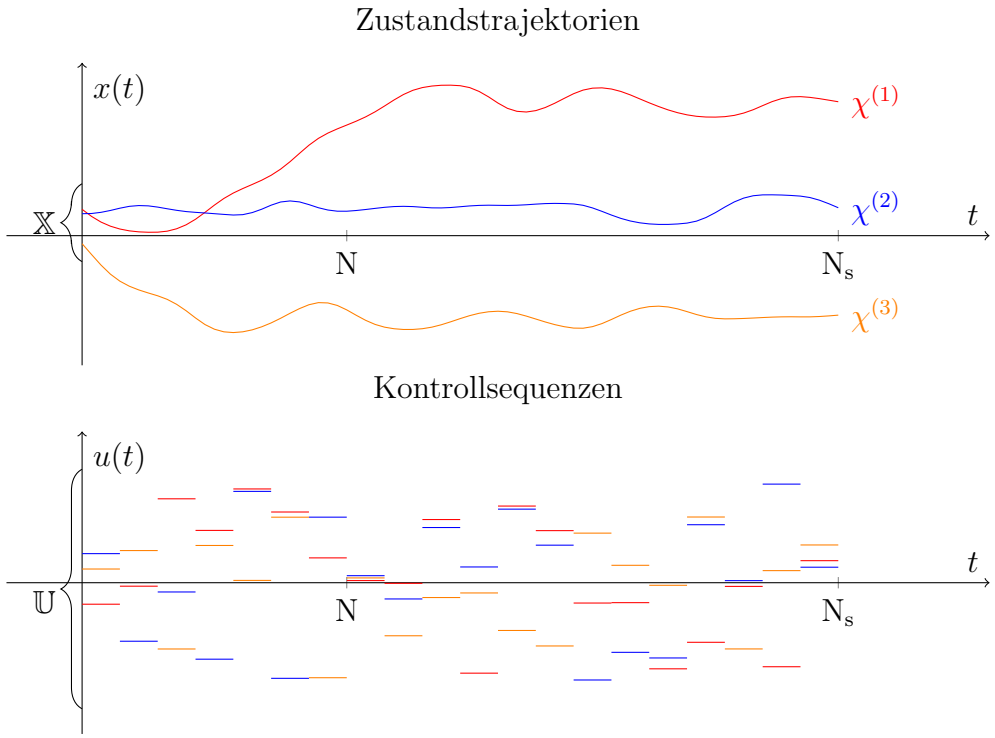

Abbildung 2.3: Simulation des Systems zur Gewinnung der Lerndaten für eine kleine Anzahl von Startwerten und Kontrollfolgen.

Trainingspaaren $(Z^{(1)}(k), r^{(1)}(k))$ führen. Daraus ergibt sich die Defintion der Menge der Lerndaten $\mathcal{E}$ als:

$$\mathcal{E} := \left\{ \left(r^{(j)}(k), Z^{(j)}(k) \right) \right\}_{(j,k)\in\{1,\dots,n_g\}\times\{N,\dots,N_s\}}$$

Die Kardinalität dieser Menge ist offenbar gegeben durch:

$$n_E := n_g \cdot (N_s - N + 1)$$

Zusätzlich können aus den Lösungstrajektorien die Werte r_{min} und r_{max} bestimmt werden, die die Transformation η in (2.6) festlegen.

Dafür muss lediglich das Maximum bzw. Minimum des p-ten Eintrags aller Lösungen an jedem Zeitpunkt gebildet werden:

$$r_{min} := \min_{(j,k)\in\{1,\dots,n_g\}\times\{0,\dots,N_s\}} x(k, t_0, x_0^{(j)}, \tilde{u}^{(j)})_{(p)}$$

$$r_{max} := \max_{(j,k)\in\{1,\dots,n_g\}\times\{0,\dots,N_s\}} x(k, t_0, x_0^{(j)}, \tilde{u}^{(j)})_{(p)}$$

Es sei angemerkt, dass die erhaltenen Werte lediglich den Wertebereich der Zustandskomponente für die betrachteten Startwerte und Kontrollen und auch nur auf dem endlichen Intervall $[0, hN_s]$ beschreiben. Es können also durchaus Zustände auftreten, die diesen Bereich verlassen, wodurch der Beobachter nicht mehr sinnvoll einsetzbar ist. Der Bereich $[r_{min}, r_{max}]$ der beobachteten Zustandskomponente wird deshalb in der Praxis

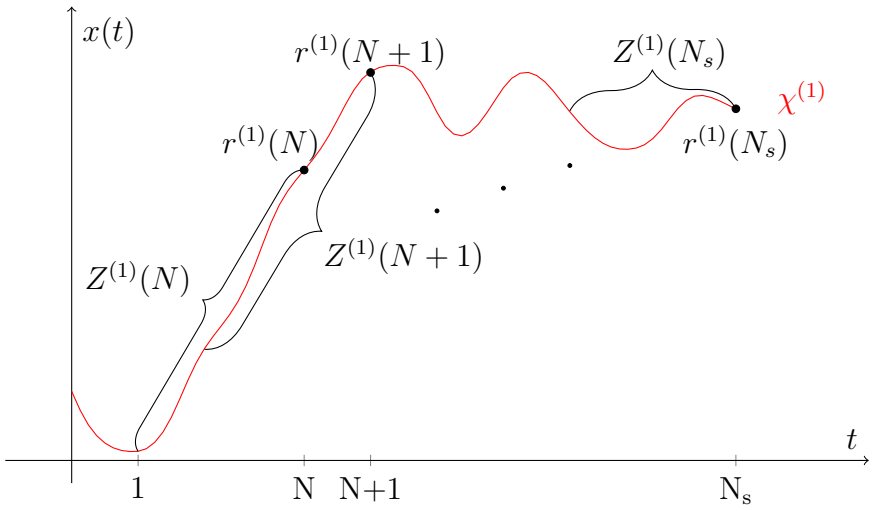

Abbildung 2.4: Zustandekommen der einzelnen Horizonte $Z^{(1)}(k)$, $k \in \{N, \ldots, N_s\}$ aus einer Lösungstrajektorie $\chi^{(1)}$. Die Darstellung veranschaulicht, welche Teilstücke von $\chi^{(1)}$ in welchen Horizont eingehen. Außerdem ist jeweils der zugehörige exakte Zustand $r^{(1)}(k)$ markiert.

noch durch eine Toleranz vergrößert. Dadurch ist der Beobachter auch in solchen Fällen verwendbar, bei denen der Zustand am Rand des betrachteten Bereichs liegt.

Ein etwas anderer Ansatz bei der Wahl der Startwerte wird im MATLAB-Programm verwendet. Statt eines gleichmäßigen Gitters über $\mathbb{X}$ wird für jedes Intervall $[x_i^{min}, x_i^{max}]$ eine Betaverteilung festgelegt. Die einzelnen Komponenten des Startwertes werden dann entsprechend dieser Verteilungen zufällig gewählt.

Der Grund dieser Entscheidung ist, dass in der Praxis das Verhalten eines Systems in einem bestimmten Bereich, wie z. B. in der Umgebung um ein Gleichgewicht, oft von besonderem Interesse ist. Durch geeignete Parameterwahl der Betaverteilung kann man Erwartungswert und Varianz so setzen, dass im kritischen Bereich viele Startwerte liegen. Man hofft, durch die höhere Informationsdichte dort eine sehr präzise Zustandsschätzung zu erhalten. In größerer Entfernung vom Erwartungswert ist dagegen eine weniger genaue Zustandsapproximation ausreichend.

Auch für die Festlegung der Kontrollwerte ist es möglich eine Verteilung über $\mathbb{U}$ zu verwenden. Die erhaltenen Kontrollfolgen weisen dadurch jedoch ein Verhalten auf, das in der Realität oft nicht auftritt. Alternativ kann man daher eine eigene Funktion für die Generierung der Kontrollfolgen angeben, die real auftretende Kontrollen genauer nachbildet.

2.2.3 Verwendung des Beobachters

Die Lösung des QP-Problems aus Abschnitt 2.2.1 liefert uns die Vektoren L und μ, die die Funktion F gemäß (2.4) implizit[4] modellieren. Es ist noch nicht klar, wie man die-

[4]Implizit, weil μ die Funktion Γ beschreibt, aber F durch Γ^{-1} definiert ist.

se Ergebnisse verwendet, um aus einer Historie den geschätzten Zustand zu berechnen. Darauf wollen wir jetzt genauer eingehen.

Man nutzt wieder die Eigenschaft (2.8) aus. Wir betrachten die Funktion:

$$G_Z(r) := Z^T L - B(\eta(r))\mu$$

Für eine Nullstelle $\hat{r}$ der Funktion G_Z gilt nach der Konstruktion aus Abschnitt 2.1 $F(Z) \approx \hat{r}$. Zur Verwendung des Beobachters löst man also für die gegebene Historie Z das Nullstellenproblem $G_Z(\hat{r}) = 0$ und erhält so den zugehörigen Zustand $\hat{r}$.

In der Praxis finden wir zunächst eine Nullstelle $\hat{\theta}$ der Funktion

$$\tilde{G}_Z(\theta) := Z^T L - B(\theta)\mu.$$

Man muss nun noch die Transformation (2.6) rückgängig machen und erhält so den Zustand

$$\hat{r} = \eta^{-1}(\hat{\theta}) = r_{min} + \hat{\theta}(r_{max} - r_{min}).$$

Im MATLAB-Programm wird zur Lösung dieses eindimensionalen Nullstellenproblems eine Kombination aus Bisektionsverfahren und Newton-Verfahren verwendet. Als Startintervall wählt man $[0, 1]$, da die Basisfunktionen nur auf diesem Bereich definiert sind. Mit dem Bisektionsverfahren wird zunächst ein Startwert für das Newton-Verfahren gefunden. Das ist nötig, da sichergestellt werden muss, dass die Iterationsfolge des Newton-Verfahrens das Intervall $[0, 1]$ nicht verlässt. Anschließend wird zum Newton-Verfahren gewechselt, um von dessen höherer Konvergenzordnung zu profitieren. Details zu den erwähnten Methoden finden sich in [4].

Damit haben wir die nötigen theoretischen Grundlagen des Beobachters erarbeitet. Im nun folgenden Kapitel werden wir anhand eines Beispiels den praktischen Einsatz demonstrieren.

3 Anwendung des Beobachters für das inverse Pendel auf dem Wagen

Wir werden nun als Anwendungsbeispiel für den Beobachter das inverse Pendel auf dem Wagen betrachten. Nach einer kurzen Einführung in das Modell wird der Beobachter explizit konstruiert. Anschließend verwenden wir diesen, um bei dem am Lehrstuhl vorhandenen Pendel die Zustandsschätzung durchzuführen.

3.1 Pendelmodell

Beim einachsigen inversen Pendel handelt es sich um eine Stange, die frei drehbar an einem Wagen befestigt ist. Der Wagen kann in der Horizontalen mittels eines Motors bewegt werden. Die Steuerung des Wagen erfolgt über eine Kontrolle u, die die Beschleunigung des Motors festlegt. Die Auslenkung des Pendels, gemessen in Relation zur aufrechten Position, wird durch den Winkel Φ angegeben. Die Bezeichnung *inverses* Pendel rührt daher, dass man das Ziel verfolgt, das Pendel aufrecht stehend zu balancieren. Dies kann durch passende Steuerung des Wagens erreicht werden.

In Abbildung 3.1 ist der Aufbau eines inversen Pendels schematisch dargestellt.

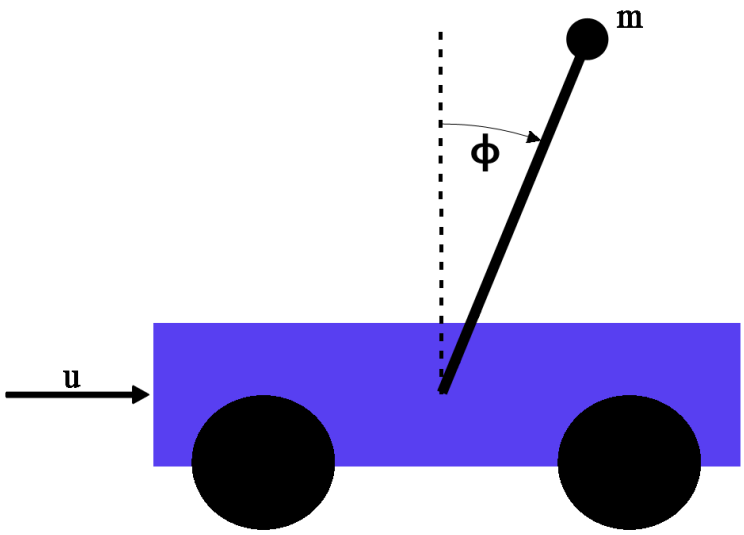

Abbildung 3.1: Skizzenhafte Darstellung eines inversen Pendels auf einem Wagen. (Quelle: [3])

3.1.1 Das Kontrollsystem

Zur mathematischen Beschreibung des inversen Pendels wird das verbesserte Modell aus [3] verwendet, bei dem der Motor des Wagens als PT_1-Glied approximativ modelliert ist:

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= -K_1\ x_2(t) + K_2\ (g \sin x_1(t) + x_5(t) \cos x_1(t)) \\
\dot{x}_3(t) &= x_4(t) \\
\dot{x}_4(t) &= x_5(t) \\
\dot{x}_5(t) &= \frac{1}{T}(K\ u(t) - x_5(t))
\end{aligned} \tag{3.1}$$

Die einzelnen Komponenten des Zustandsvektors $x(t) \in \mathbb{R}^5$ haben folgende Bedeutung:

$x_1(t)$	:	Winkelauslenkung Φ zwischen Pendel und aufrechter Position, positiv im Uhrzeigersinn
$x_2(t)$	:	Winkelgeschwindigkeit
$x_3(t)$	:	Position des Wagens
$x_4(t)$	:	Geschwindigkeit des Wagens
$x_5(t)$	:	Verzögerte Beschleunigung des Wagens

Als Kontrolle $u(t) \in \mathbb{R}$ wird die Sollbeschleunigung des Wagens vorgegeben.

In die Konstanten K_1 und K_2 fließen verschiedene Faktoren wie Reibung oder Eigenschaften der Pendelstange ein, auf die wir nicht näher eingehen. Die Konstante $g := 9.81$ beschreibt die Gewichtskraft.

Durch die Zeitkonstante T wird die Verzögerung modelliert, mit der eine gewählte Beschleunigung vom Motor des Wagens umgesetzt wird. Die Konstante K ist der Verstärkungsfaktor.

Zur Messung des Systemzustands stehen Sensoren zur Ermittlung des Auslenkungswinkels $x_1(t)$ und der Wagenposition $x_3(t)$ zur Verfügung. Der Ausgang $y(t)$ des Systems ist also gegeben durch:

$$y(t) = C(x(t), u(t)) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} x(t) = \begin{pmatrix} x_1(t) \\ x_3(t) \end{pmatrix} \tag{3.2}$$

An dieser Stelle wird der Beobachter ansetzen, um den vollständigen Systemzustand aus den Messdaten zu rekonstruieren.

3.1.2 Modellkonstanten

Die im Modell vorhandenen Konstanten wurden in [3] mit einem Programm bereits geschätzt. Darin werden die Parameter K_1, K_2, T und K mittels Optimierung so bestimmt, dass das Modell des Systems möglichst gut mit einer Reihe von Messungen übereinstimmt.

Mittlerweile ist am Pendel des Lehrstuhls jedoch eine Stange anderer Länge und verschiedenen Gewichts eingebaut. Die Konstanten entsprechen also nicht mehr den aktuellen physikalischen Gegebenheiten.

Für die Bestimmung der korrekten Parameter wurden neue Messreihen mit verschiedenen Beschleunigungsprofilen durchgeführt und das Programm mit den gewonnen Daten nochmals ausgeführt. Es ergaben sich die Werte:

$$\begin{aligned} K_1 &:= 0.116529163967995 \\ K_2 &:= 3.954189381512292 \\ T &:= 0.007032146489483 \\ K &:= 0.977640071078069 \end{aligned}$$

Damit haben wir ein mathematisches Modell, mit dem das real existierende Pendel sehr genau beschrieben wird. Dieses wollen wir im Folgenden zur Implementierung des Beobachters nutzen.

3.2 Implementierung des Beobachters

Zur Umsetzung des Beobachters für das Pendel wählen wir eine Schrittweite von

$$h = 0.050\ s = 50\ ms.$$

Diese Wahl geschieht mit dem Hintergedanken, dass der Beobachter in zukünftigen Projekten in einem MPC-Regler verwendet werden soll. In diesem Fall könnte beispielsweise alle 50 ms eine Kontrolle ausgehend vom geschätzten Zustand berechnet werden.

Die Länge des Horizonts legen wir zunächst auf

$$N = 3$$

fest, der Beobachter wird also Messwerte und Kontrollen aus den vergangenen 100 ms berücksichtigen. Beachte, dass auch die Messwerte zum aktuellen Zeitpunkt in die Berechnung des gegenwärtigen Zustands eingehen. Hier tritt natürlich eine leichte Verzögerung auf, das heißt der geschätzte Zustand ist zum Zeitpunkt der Berechnung bereits veraltet.

Für die Generierung der Lerndaten müssen die Bereiche festgelegt werden, in denen sich die Startzustände des Systems und die Kontrollen bewegen können. Wir orientieren uns dabei an den real auftretenden Größen.

Der Auslenkungswinkel des Pendel wird im Bogenmaß angegeben und bewegt sich im Bereich von $[-\pi, \pi]$. Die maximale Fahrstrecke des Wagens liegt bei ca. 70 cm. Da die Messung der Wagenposition relativ zur Mitte dieser Strecke erfolgt, können also nur Werte zwischen $-0.35\ m$ und $0.35\ m$ auftreten. In der Bedienungsanleitung des Pendels wird die maximale Geschwindigkeit des Wagens mit 6 m/s, die maximale Beschleunigung des Motors mit 8 m/s^2 angegeben. Für die Winkelgeschwindigkeit lagen keine Daten

vor, die Simulation des Pendelmodells lässt Werte zwischen -15 und 15 als sinnvoll erscheinen. Wir setzen daher die Menge der Startwerte für die Simulation auf

$$\mathbb{X} = [-\pi, \pi] \times [-15, 15] \times [-0.35, 0.35] \times [-6, 6] \times [-8, 8]$$

und die Menge der zulässigen Kontrollen auf

$$\mathbb{U} = [-8, 8].$$

Wir simulieren das System mit $n_g = 500$ zufälligen Startwerten aus $\mathbb{X}$ und zugehörigen, ebenfalls zufällig generierten Kontrollfolgen der Länge $N_s = 30$ mit Werten aus $\mathbb{U}$, also jeweils über eine Zeitspanne von $hN_s = 1500\ ms$. Damit erhalten wir Simulationsdaten für einen Beobachter mit maximaler Horizontlänge 30.

Zum Training des neuronalen Netzes erzeugen wir aus diesen Daten 500 Paare (Z, r). Die Identifikation erfolgt vorerst unter Verwendung von $n_b = 12$ Basisfunktionen mit Parameter $\beta = \frac{5}{6}$. Für die erste Nebenbedingung des resultierenden QP-Problems wird ein Gitter der Größe $l = 1000$ verwendet.

3.3 Anwendung des Beobachters in der Simulation

Zur grafischen Analyse betrachten wir als Beispiel das vorgegebene Kontrollprofil aus Abbildung 3.2 über eine Zeitspanne von fünf Sekunden. Als Startwert wird $x_0 = (\pi, 0, 0, 0, 0)^T$ gewählt, was dem ruhig herabhängenden Pendel und still stehenden Wagen entspricht. In den folgenden Abbildungen sind ausschließlich die Komponenten x_1 bis x_4 dargestellt. Die x_5-Komponente zeigt im Wesentlichen nur die gesetzte Kontrolle mit der Verzögerung bedingt durch das PT_1-Glied, daher wird auf die Darstellung verzichtet.

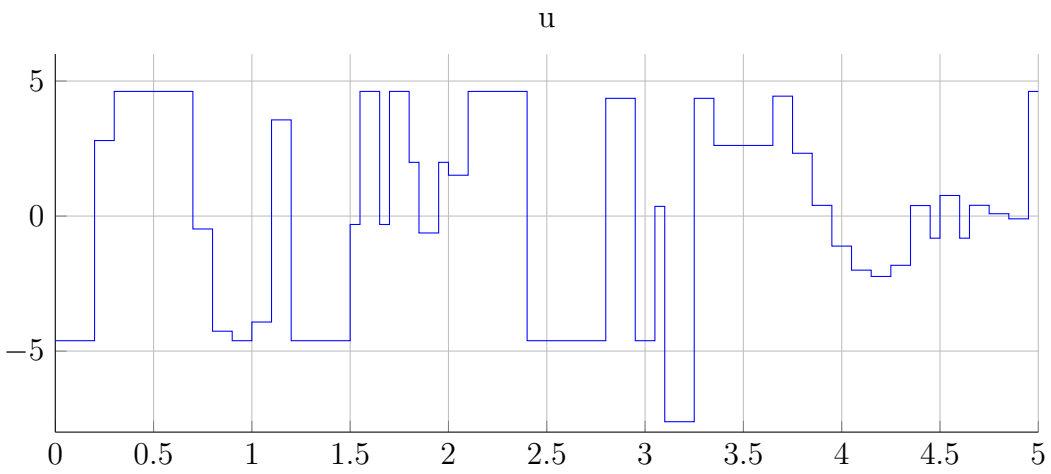

Abbildung 3.2: Verwendete Kontrollsequenz für die Simulationen.

Das Ergebnis beim Einsatz des Beobachters in der Simulation ist in Abbildung 3.3 dargestellt. Die schwarze Linie markiert hier wie im Folgenden den exakten Zustand, der geschätzte Zustand ist stets rot gestrichelt.

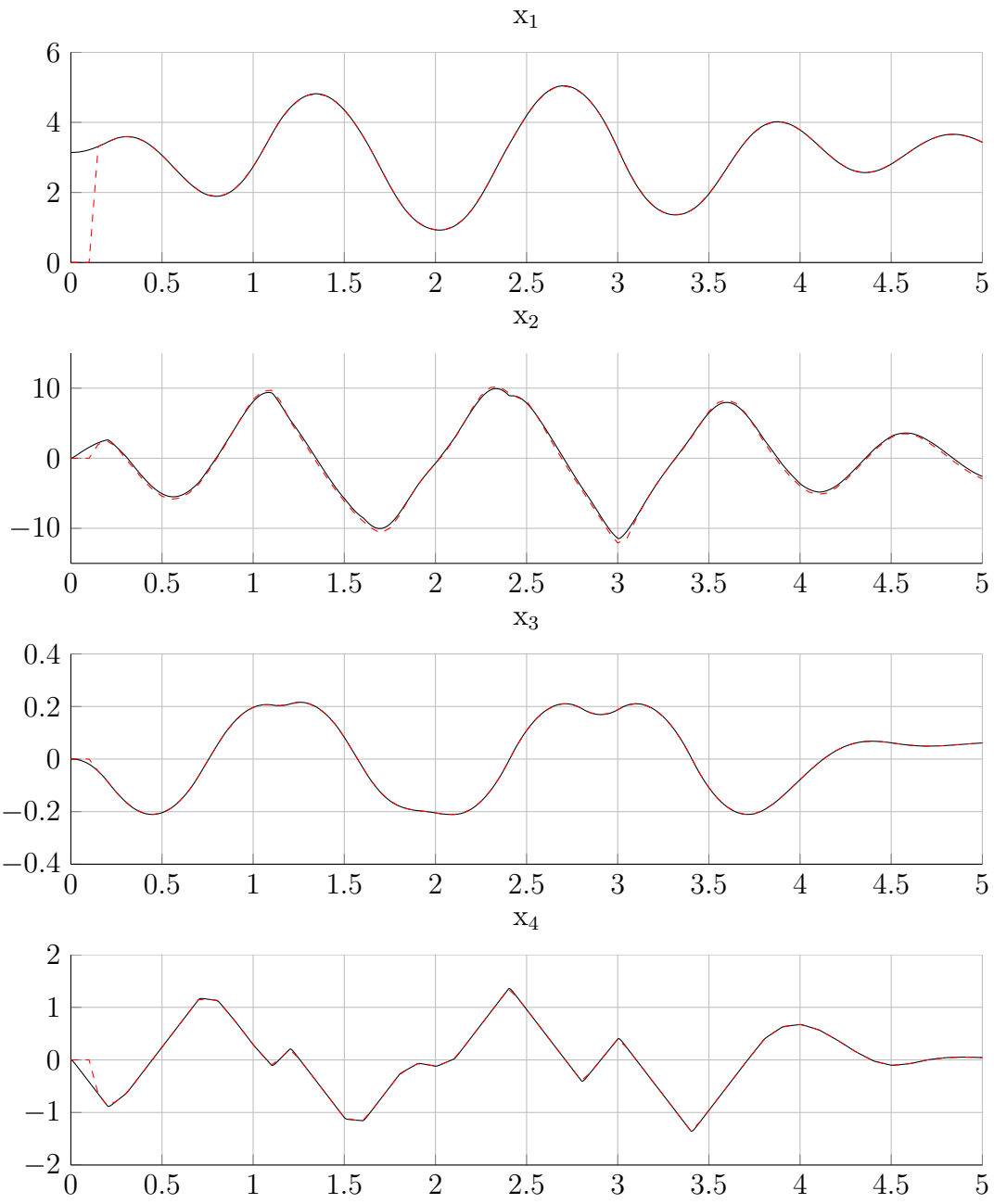

Abbildung 3.3: Beobachter für eine Horizontlänge von $N = 3$. Der Zustand wird präzise approximiert, in der x_2-Komponente sind bei genauerer Betrachtung aber noch Abweichungen zu erkennen.

Wie man erkennt, liefert der erhaltene Beobachter zumindest rein optisch eine gute Zustandsschätzung. Der große Sprung am Anfang erklärt sich dadurch, dass erst nach 150 ms erstmals genug Vergangenheitswerte für eine Zustandsschätzung vorhanden sind. Der Beobachter muss sozusagen erst „warmlaufen“.

Während der geschätzte Zustand für den Winkel, sowie Position und Geschwindigkeit des Wagens schon sehr präzise ist, treten bei genauerer Betrachtung der Winkelgeschwindigkeit aber noch sichtbare Abweichungen auf. Abbildung 3.4 zeigt die x_2-Komponente noch einmal gesondert.

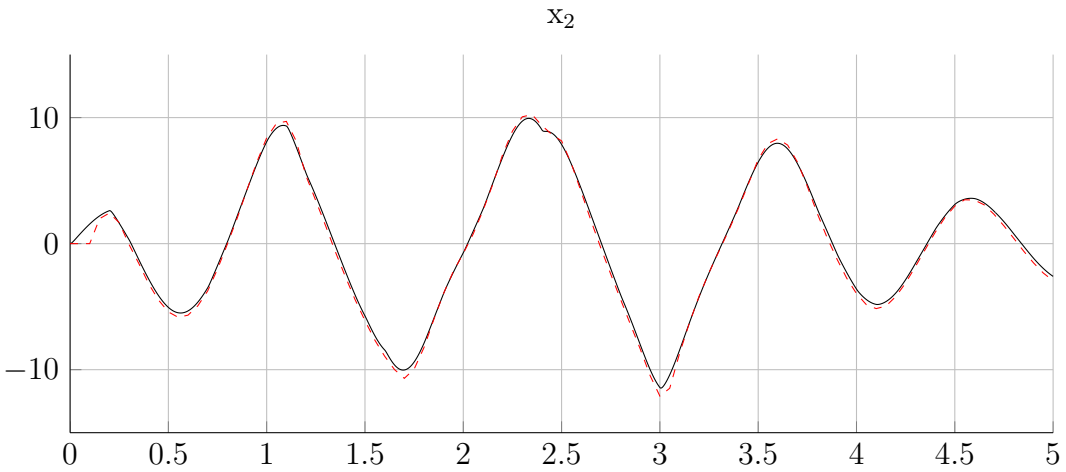

Abbildung 3.4: Vergleich der x_2-Komponente mit dem Ergebnis der Zustandsschätzung für $N = 3$. Vor allem an den Extrema kann man noch leichte Abweichungen erkennen.

Denkbar wäre nun die Anzahl der Trainingspaare zu erhöhen, um ein besseres Ergebnis zu erhalten. Die Idee dahinter ist, dass der Zustandsraum durch nur 500 Referenzpaare möglicherweise noch nicht ausreichend abgedeckt ist. Es stellt sich aber heraus, dass die Zustandsschätzung ab einer gewissen Mindestanzahl dadurch nicht mehr besser wird und unter Umständen sogar ungenauer.

Ein Erklärung dafür könnte sein, dass der maximale Eigenwert der Matrix Q des QP-Problems mit zunehmender Anzahl immer größer wird. Gleichzeitig gibt es aber auch Eigenwerte, die nahe bei 0 sind. Die Differenz zwischen maximalem und minimalem Eigenwert nimmt also betragsmäßig zu. Das wirkt sich negativ auf die Genauigkeit des Ergebnisses des Optimierungsproblems aus.

Stattdessen verdoppeln wir die Horizontlänge auf $N = 6$. Dadurch erhält man auch hier eine wesentlich bessere Schätzung, wie in Abbildung 3.5 zu sehen ist. Eine weitere Verlängerung des Horizonts macht das Ergebnis im Fall des Pendels noch genauer. Man sollte aber bedenken, dass dadurch der Speicherbedarf des Beobachters größer wird, da mehr Messwerte festgehalten werden müssen. Zudem kann die Generierung der Lerndaten für ausgedehnte Horizontlängen sehr lange dauern. Auch verlängert sich dadurch die „Aufwärmphase“ des Beobachters.

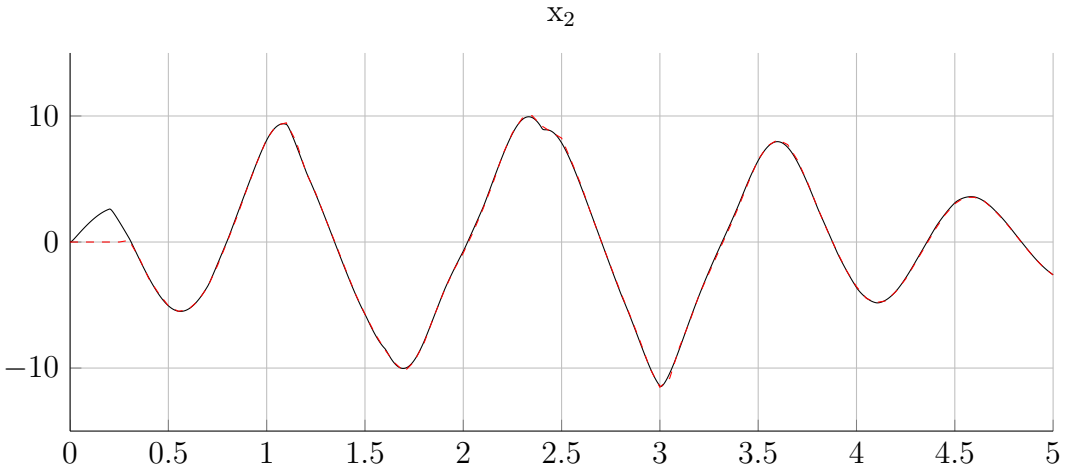

Abbildung 3.5: Vergleich der x_2-Komponente mit dem Ergebnis der Zustandsschätzung für $N = 6$. Durch den längeren Horizont wird die Zustandsapproximation genauer. Der Beobachter liefert aber erst nach 300 ms den ersten Zustand.

3.3.1 Eigenschaften der Funktionenbasis

Wir wollen außerdem die Auswirkung von Veränderungen der Größe der Funktionenbasis bei konstanter Horizontlänge untersuchen. Zusätzlich stellt sich die Frage, wie der Parameter β der Basisfunktionen gewählt werden sollte. Eine definitive Antwort ist hier schwer zu geben. Grob gesprochen sollte man β so festlegen, dass bei der graphischen Darstellung der Basisfunktionen das Quadrat $[0, 1] \times [0, 1]$ gut ausgefüllt wird und der Abstand der einzelnen Basisfunktionen voneinander einigermaßen gleichmäßig ist (vgl. Abbildung 3.6). Mit der Wahl von $\beta = \frac{10}{n_b}$ erhält man für kleine Basisgrößen (< 30) meist eine passende Verteilung. Wir werden β im Folgenden stets so wählen und nicht explizit angeben.

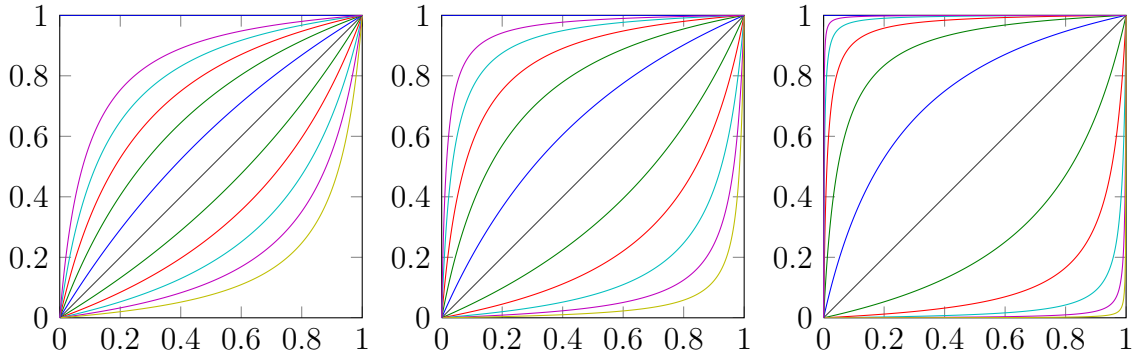

Abbildung 3.6: Vergleich von $n_b = 12$ Basisfunktionen mit unterschiedlichen Parametern $\beta = \frac{1}{2}$ (links), $\beta = \frac{5}{6}$ (mittig) und $\beta = \frac{3}{2}$ (rechts). Beim linken Bild decken die Basisfunktionen nicht genug Fläche ab, rechts sind die Abstände nicht gleichmäßig. Das Bild in der Mitte zeigt eine gute Verteilung der Basisfunktionen.

Die Anzahl der Basisfunktionen bestimmt direkt die Dimension des Optimierungsproblems, das zur Identifikation des Beobachters gelöst werden muss. Außerdem nimmt mit steigender Anzahl auch die Zahl von Funktionsauswertungen zu, die zur Lösung des Nullstellenproblems beim Einsatz des Beobachters nötig sind. Es ist also wünschenswert eine möglichst kleine Basis zu verwenden.

Im betrachteten Beispiel des inversen Pendels ist der Einfluss der Basisgröße eher gering. Schon die kleinste denkbare Basisgröße von $n_b = 2$ liefert eine sinnvolle Approximation. Wie in Abbildung 3.7 zu sehen ist, sind aber vor allem in der x_2-Komponente noch leichte Abweichungen zu erkennen.

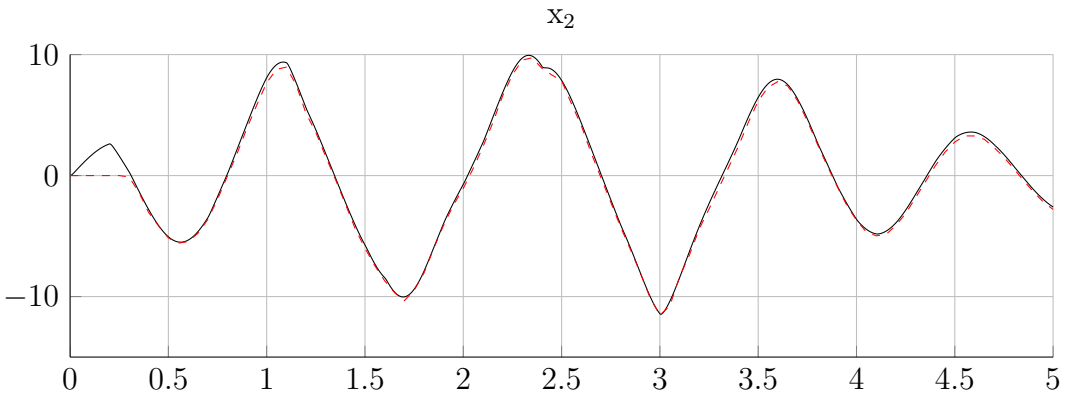

Abbildung 3.7: Vergleich der x_2-Komponente mit dem Ergebnis der Zustandsschätzung für $N = 6$ und $n_b = 2$. Trotz der geringen Basisgröße wird der Zustand gut rekonstruiert.

Ab einer Basisgröße von ca. $n_b = 10$ sind mit bloßem Auge auch hier keine nennenswerten Verbesserungen mehr feststellbar. Deshalb wurde zur besseren Vergleichbarkeit der mittlere Fehler bei der Schätzung des Zustands in Abhängigkeit von der Basisgröße in Tabelle 3.1 zusammengefasst. Die Darstellung beschränkt sich auf die x_2-Komponente, da dort erfahrungsgemäß die größten Abweichungen auftreten, was sich auf die Nichtlinearität der 2. Komponente des Vektorfelds (3.1) zurückführen lässt.

Man erkennt, dass der Fehler zunächst deutlich abnimmt, aber dann keine große Verbesserung mehr eintritt.

Wie erwähnt wirkt sich die Größe der Basis in jedem Iterationsschritt des Bisektions- und Newton-Verfahrens durch die Funktionsauswertungen der einzelnen Basisfunktionen auf die benötigte Zeit für die Zustandsschätzung aus. Dagegen beeinflusst die Länge des Horizonts die Geschwindigkeit der Ausführung kaum. Der Grund dafür ist, dass bei der Lösung des Nullstellenproblems zur Zustandsberechnung nur einmal das Skalarprodukt $Z^T L$ aus Horizont Z und Vektor L gebildet werden muss.

Für die praktische Anwendung ist es daher interessant zu wissen, wie stark sich die Verwendung von mehr Basisfunktionen auf die Laufzeit niederschlägt. Deshalb wurde eine Reihe von Zeitmessungen beim Einsatz des Beobachters durchgeführt.

Tabelle 3.2 enthält die durchschnittliche Zeit in Abhängigkeit von der Basisgröße, die der Beobachter auf beweglichem Horizont benötigt, um den Systemzustand beim

Tabelle 3.1: Mittlerer Fehler der Zustandsschätzung von x_2 in Abhängigkeit der Basisgröße n_b.

n_b	mittlerer Fehler
2	0.23676098
4	0.22626512
6	0.29434189
8	0.16040419
10	0.0868136
12	0.08297110
14	0.07631983
16	0.07926079
18	0.07783417
20	0.07710436
22	0.07837954
24	0.07653727

inversen Pendel zu rekonstruieren. Die Daten repräsentieren dabei den Zeitaufwand im MATLAB-Programm, das kaum Optimierungen enthält.

Tabelle 3.2: Zeit in Abhängigkeit der Basisgröße, die der Beobachter im Schnitt für die Berechnung des Zustands benötigt.

n_b	Zeit in s
2	0.00557407
4	0.00644897
6	0.00659955
8	0.00664260
10	0.00660610
12	0.00646121
14	0.00644940
16	0.00651158
18	0.00648375
20	0.00639173
22	0.00650773
24	0.00648914

Man erkennt, dass die Zustandsschätzung im Schnitt stets um die 6.5 ms dauert. Einzig bei der Verwendung von nur 2 Basisfunktionen spart man etwa 1 ms. In der Praxis wird man daher abwägen müssen, ob man auf die verbesserte Genauigkeit zugunsten des geringeren Aufwands bei der Nullstellenberechnung verzichtet.

Durch eine Implementierung in C++ kann man eine deutlich schnellere Berechnung erwarten, sodass die Rechenzeit gegenüber der Abtastrate des Systems in den Hintergrund tritt und man sich stattdessen auf die Präzision der Schätzung konzentrieren kann.

3.3.2 Verlassen des Wertebereichs

Bei der Zustandsschätzung kann das Problem auftreten, dass eine Zustandskomponente des Systems den Bereich verlässt, der durch r_{min} bzw. r_{max} beschränkt wird, also auf dem der Beobachter definiert wurde. Da beim Pendel am Lehrstuhl Lichtschranken vorhanden sind, die verhindern, dass der Wagen zu weit nach rechts oder links fährt, kann die Position des Wagens niemals außerhalb der angegebenen Grenzen liegen, hier treten also keine Probleme auf. Auch von den Geschwindigkeiten können wir annehmen, dass sie sich stets in einem festen Bereich bewegen.

Anders sieht es dagegen bei der Auslenkung des Pendels aus. Wir haben für die Simulation angenommen, dass der Winkel der Startwerte immer im Intervall $[-\pi, \pi]$ liegt. Es wird aber noch nicht berücksichtigt, dass das Pendel überschlagen kann. Weil der Winkelsensor die physikalisch identischen Zustände mit Winkel $\Phi = \pi$ und $\Phi = 3\pi$ unterscheidet, verlässt der Zustand des Systems nach einem Überschlag des Pendels also den für den Beobachter bekannten Bereich. Zwar werden auch in der Simulation mit genügend zufälligen Kontrollen Überschläge des Pendels vorkommen, was man auch daran sieht, dass im Beispiel der maximale Wertebereich des Winkels durch $r_{min} = -10.3824$ und $r_{max} = 12.6424$ festlegt wird. Nach mehreren Überschlägen, wie in Abbildung 3.8, ist aber keine zuverlässige Schätzung mehr möglich.

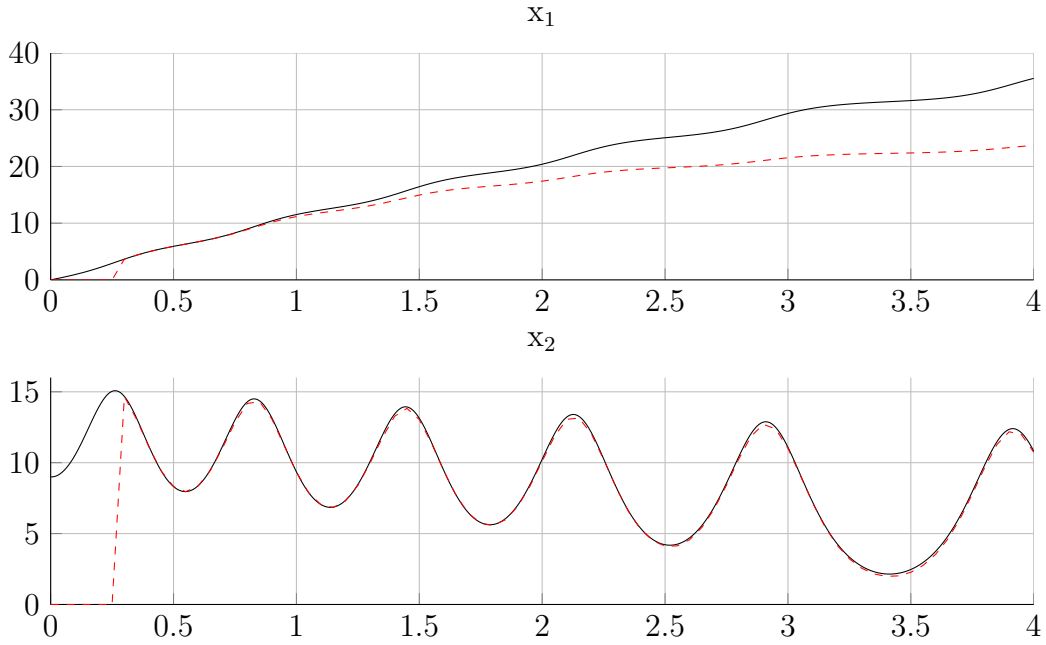

Abbildung 3.8: Abweichung der Zustandschätzung bei Verlassen des Bereichs auf dem der Beobachter trainiert wurde. Die Darstellung beschränkt sich auf den Winkel des Pendels und die Winkelgeschwindigkeit. Die Kontrolle wurde konstant gleich Null gewählt.

Wie man sieht, wird der Winkel ab einer gewissen Grenze (r_{max}) nicht mehr korrekt identifiziert. Auch für die anderen Zustandskomponenten nimmt dann die Genauigkeit des Beobachters ab.

Wir können die Problematik beheben, indem wir den Horizont so transformieren, dass die gemessenen Winkel wieder im Trainingsbereich des Beobachters liegen. Dafür werden sämtliche Winkelkomponenten im Horizont um ein ganzzahliges Vielfaches von 2π verschoben, wofür die Periodizität von Sinus und Kosinus ausgenutzt wird. Für die Berechnung der beobachteten x_1-Komponente muss die Transformation rückgängig gemacht werden, um den *richtigen* Winkel zu erhalten.

So ergibt sich wieder eine passende Zustandsrekonstruktion (vgl. Abbildung 3.9). Auch für die anderen Komponenten steigt die Genauigkeit.

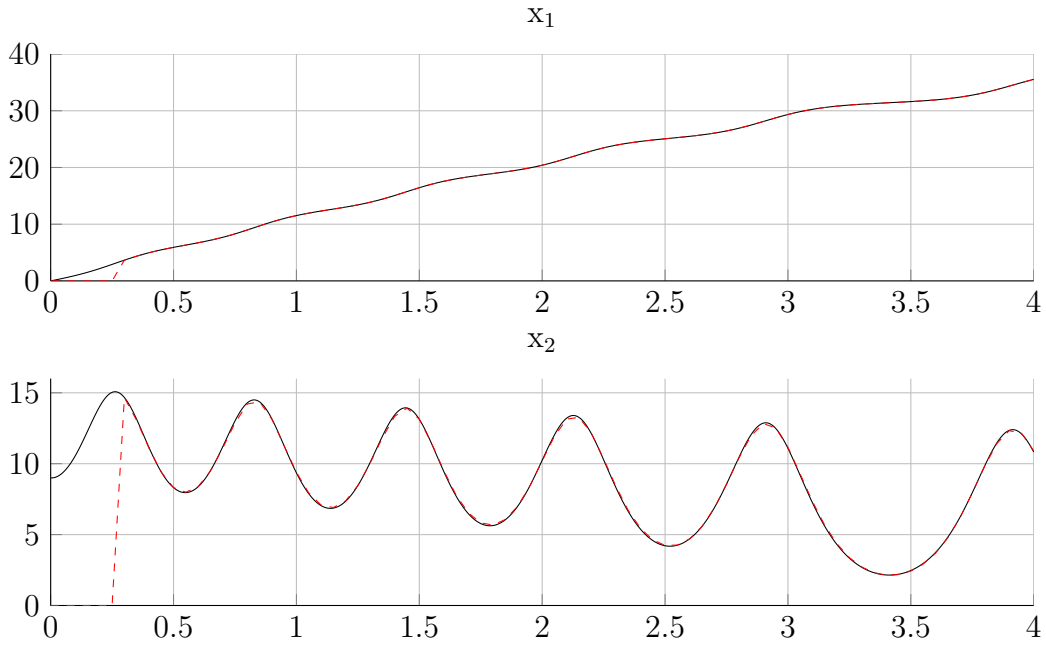

Abbildung 3.9: Hier wurden die Winkelkomponenten des Horizonts nach $[r_{min}, r_{max}]$ verschoben. Damit liefert der Beobachter wieder eine korrekte Rekonstruktion des Zustands.

3.3.3 Verhalten bei gestörtem Ausgang

In der Simulation gingen wir bisher davon aus, dass der Ausgang (3.2) des Systems die exakten Winkel- und Positionsdaten des Pendels liefert. Tatsächlich treten in der Praxis aber stets Messfehler auf, die noch nicht berücksichtigt werden. Wir untersuchen deshalb, wie der bisherige Beobachter auf solche Störungen reagiert und welche Anpassungen für den Einsatz unter realen Bedingungen gemacht werden müssen.

Für die Nachbildung eines gestörten Systemausgangs nehmen wir an, dass ein normalverteiltes weißes Messrauschen ω_1 mit Standardabweichung $\sigma_1 = 0.025$ für den Winkel und ω_2 mit Standardabweichung $\sigma_2 = 0.01$ für die Position vorliegt, jeweils mit Erwartungswert 0:

$$y(t) = C(x(t), u(t)) = \begin{pmatrix} x_1(t) \\ x_3(t) \end{pmatrix} + \begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix}$$

Die Stärke der Streuung entspricht dabei nicht der realen auftretenden Abweichung, es soll an dieser Stelle nur die allgemeine Problematik veranschaulicht werden.

In Abbildung 3.10 sind typische verrauschte Messwerte aus der Simulation des Pendels zusammen mit exaktem Winkel und korrekter Position des Wagens dargestellt.

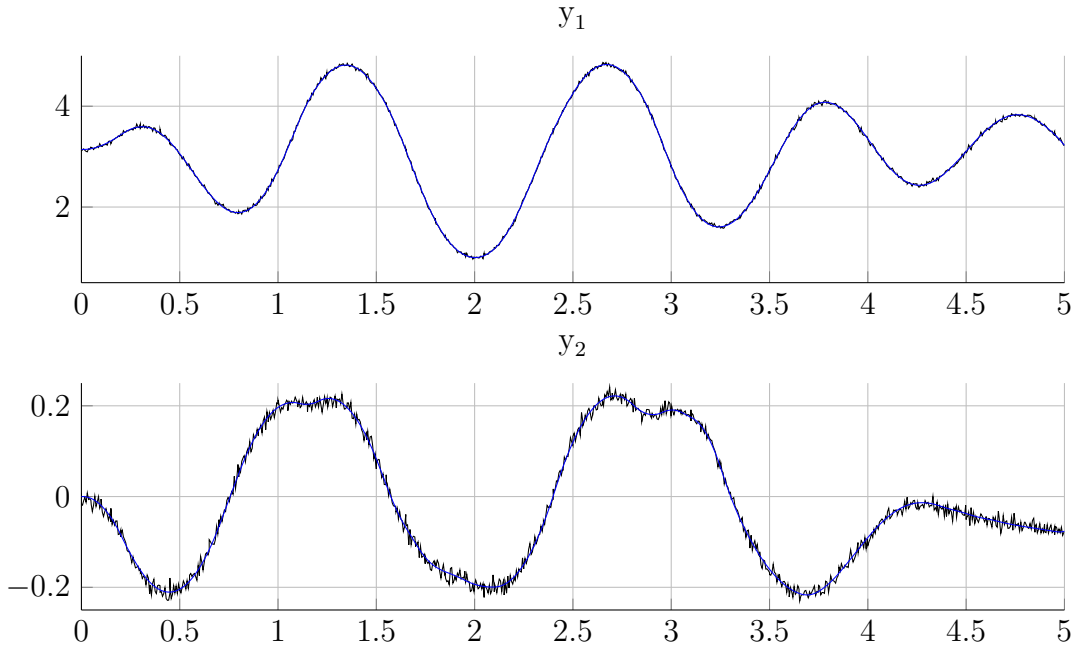

Abbildung 3.10: Typisches Messrauschen bei Systemen mit gestörtem Ausgang. Wahre Winkel- und Positionsdaten sind in blau eingezeichnet.

Der mit idealisiertem Ausgang trainierte Beobachter liefert beim Einsatz mit gestörtem Ausgang ein völlig unbrauchbares Resultat (siehe Abbildung 3.11). Bei den beobachteten Zustandskomponenten x_1 bis x_3 ist keine sinnvolle Approximation erkennbar. In der x_4-Komponente ist das Ergebnis besser, aber auch dort sind noch große Unterschiede vorhanden.

Auch eine deutliche Verlängerung des Horizonts und die Verwendung von mehr Trainingsdaten und deutlich größerer Funktionenbasis bringt hier nur kleine Verbesserungen. In Abbildung 3.12 werden x_1- und x_4-Komponente zwar sichtbar besser geschätzt, vor allem beim eigentlich interessanten Zustand x_2 versagt der Beobachter aber weiterhin.

Abhilfe schafft hier, den Beobachter bereits mit gestörtem Ausgang zu trainieren, indem man die Störung mit in die Simulation integriert. Auf diese Weise wird die Ungenauigkeit des Ausgangs bereits beim Training des Beobachters berücksichtigt. So ergibt sich schon bei einer Horizontlänge von $N = 6$ und $n_b = 12$ Basisfunktionen ein sichtbar besseres Ergebnis als beim Training mit idealem Ausgang (vergleiche Abbildung 3.13). Besonders in der x_2-Komponente sind die Abweichungen vom tatsächlichen Zustand aber noch deutlich zu sehen.

Ein größerer Horizont und mehr Trainingsdaten können den Fehler reduzieren, wie in Abbildung 3.14 zu sehen ist. Eine ähnlich genaue Schätzung wie bei exaktem Ausgang wird jedoch nicht erreicht.

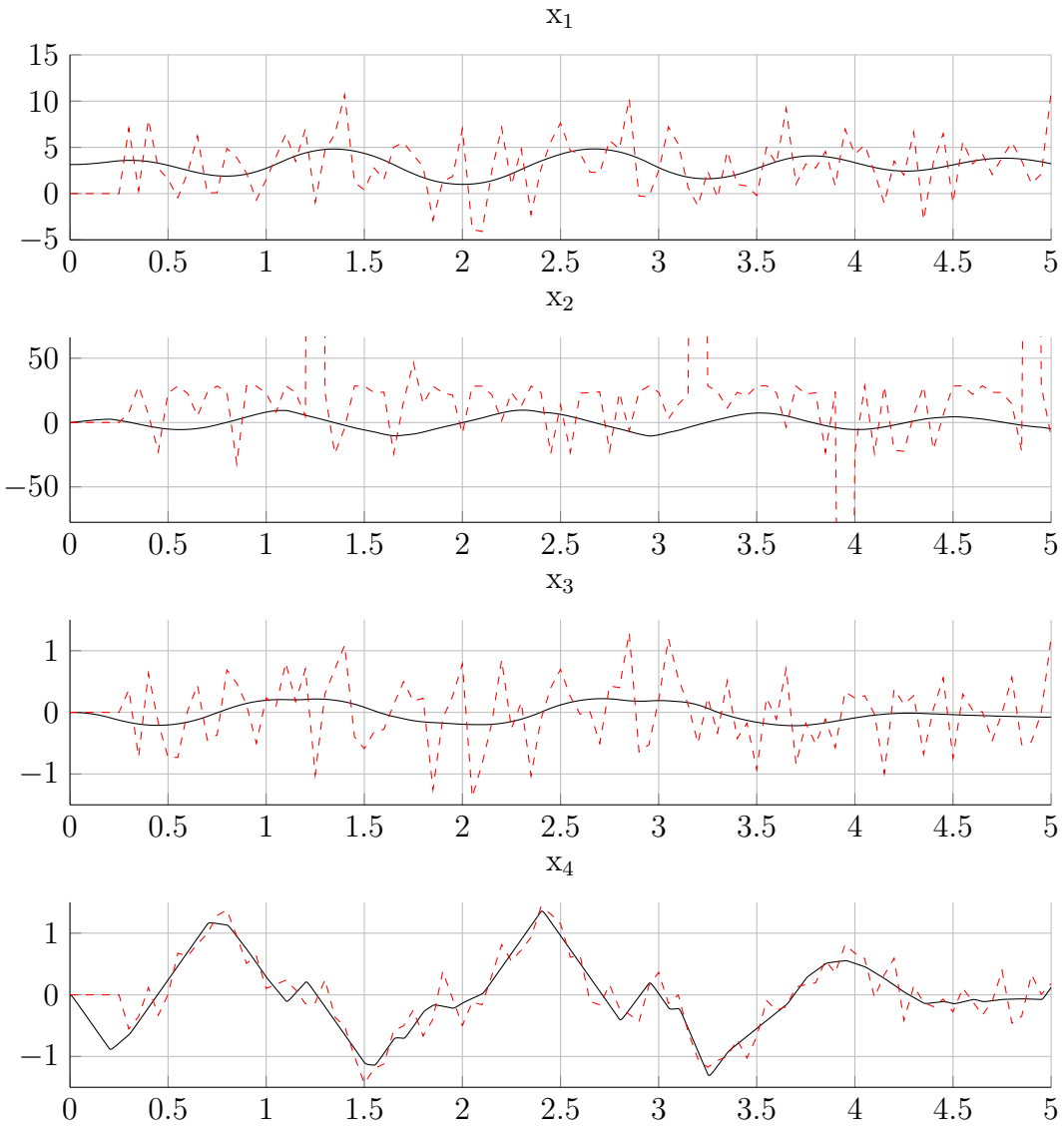

Abbildung 3.11: Der Beobachter aus dem vorherigen Abschnitt wurde ohne Störung trainiert. Beim Einsatz mit verrauschten Messwerten sind große Fehler in der Schätzung erkennbar.

Interessant ist auch, wie der Beobachters bei gestörten Systemen auf eine Verringerung der Schrittweite reagiert. Diese Untersuchung ist dadurch motiviert, dass die Sensoren des Pendels am Lehrstuhl etwa alle 0.13 ms einen Messwert liefern. Im Moment wird also nur ein sehr kleiner Bruchteil der Messwerte tatsächlich verwendet. Mit kleinerer Schrittweite findet die Zustandsberechnung insgesamt häufiger statt. Durch die zusätzliche Information würde man eine genauere Schätzung erwarten.

Es zeigt sich aber, dass mit kleinerer Schrittweite Effekte auftreten, die die Zustandsschätzung verschlechtern. Abbildung 3.15 zeigt den Beobachter mit der veränderten Abtastrate von 5 ms, es werden also effektiv zehnmal so viele Messwerte genutzt.

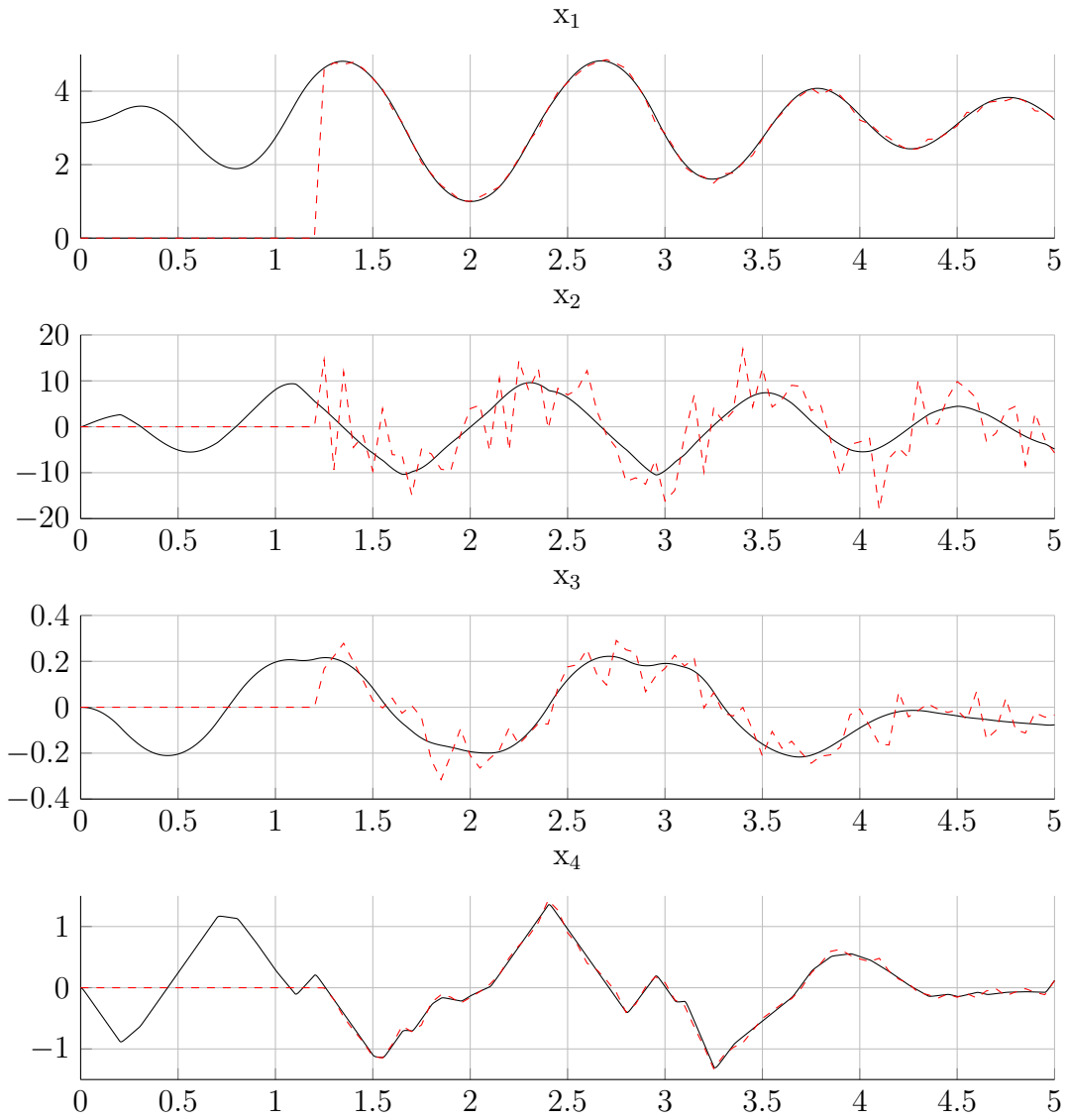

Abbildung 3.12: Der ohne Störung trainierte Beobachter mit $N = 25$ und $n_b = 24$ beim Einsatz mit Störung. Trotz des deutlich verlängerten Horizonts und mehr Basisfunktionen ist die Schätzung in der x_2- und x_3-Komponente noch fehlerhaft.

Wegen der kleineren Schrittweite wirken sich die zufälligen Abweichungen der Messung vom realen Zustand offenbar stärker aus. Dadurch sinkt die Qualität der Schätzung vor allem für die Geschwindigkeiten spürbar. Das Verhalten deutet auf eine schlechte Konditionierung des Problems hin.

Im praktischen Einsatz werden sehr kleine Schrittweiten ohnehin schwer umzusetzen sein, da auch beachtet werden muss, dass die Zustandsberechnung nicht länger dauern darf als die Abtastrate des Systems.

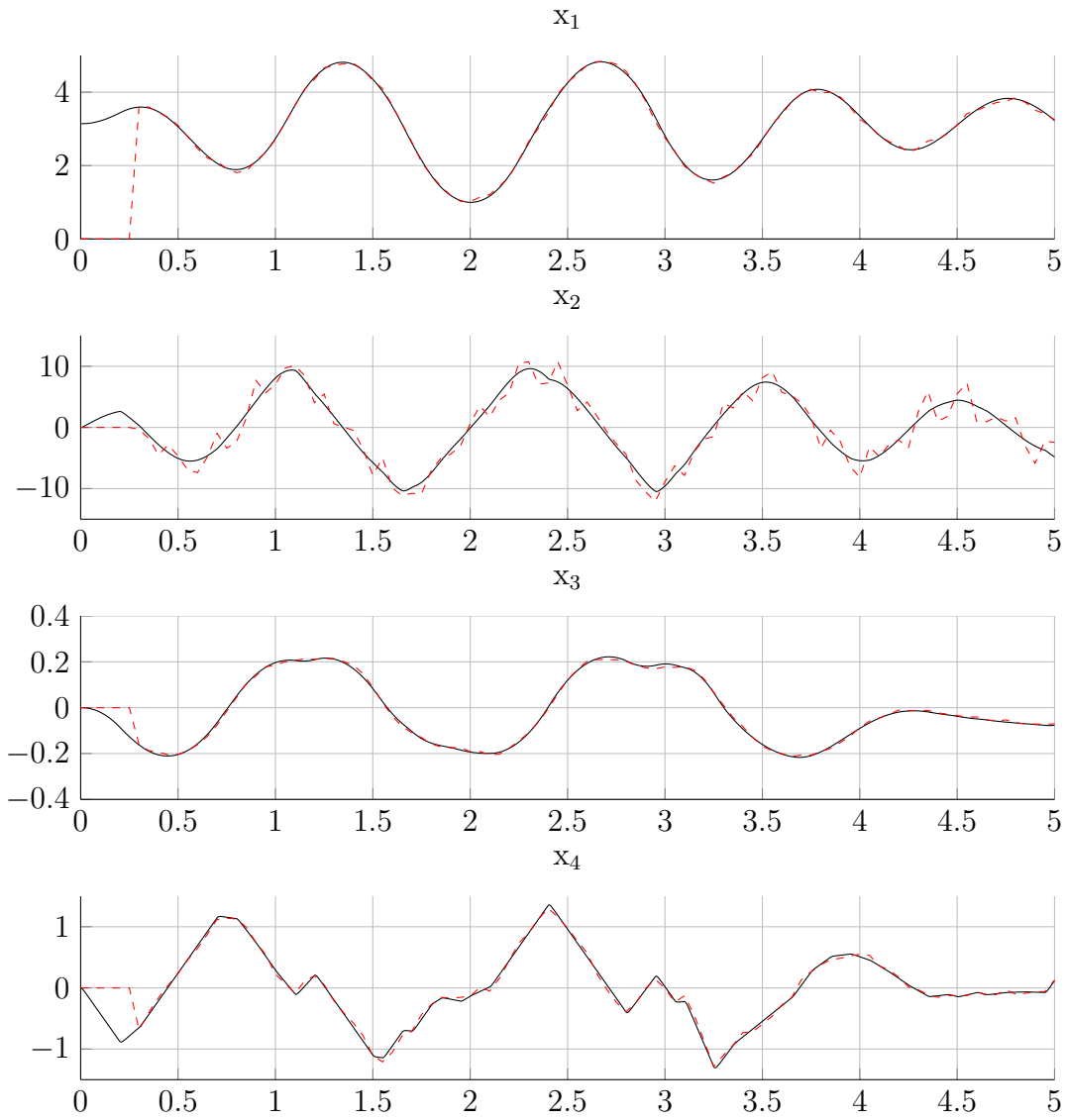

Abbildung 3.13: Ergebnis eines Beobachters mit $N = 6$ und $n_b = 12$, bei dem die Störung des Ausgangs bereits im Training berücksichtigt wurde. Die Verbesserung gegenüber den Beobachtern, die ohne Störung trainiert wurden, ist gut erkennbar. Die x_2-Komponente weist jedoch noch Ungenauigkeiten auf.

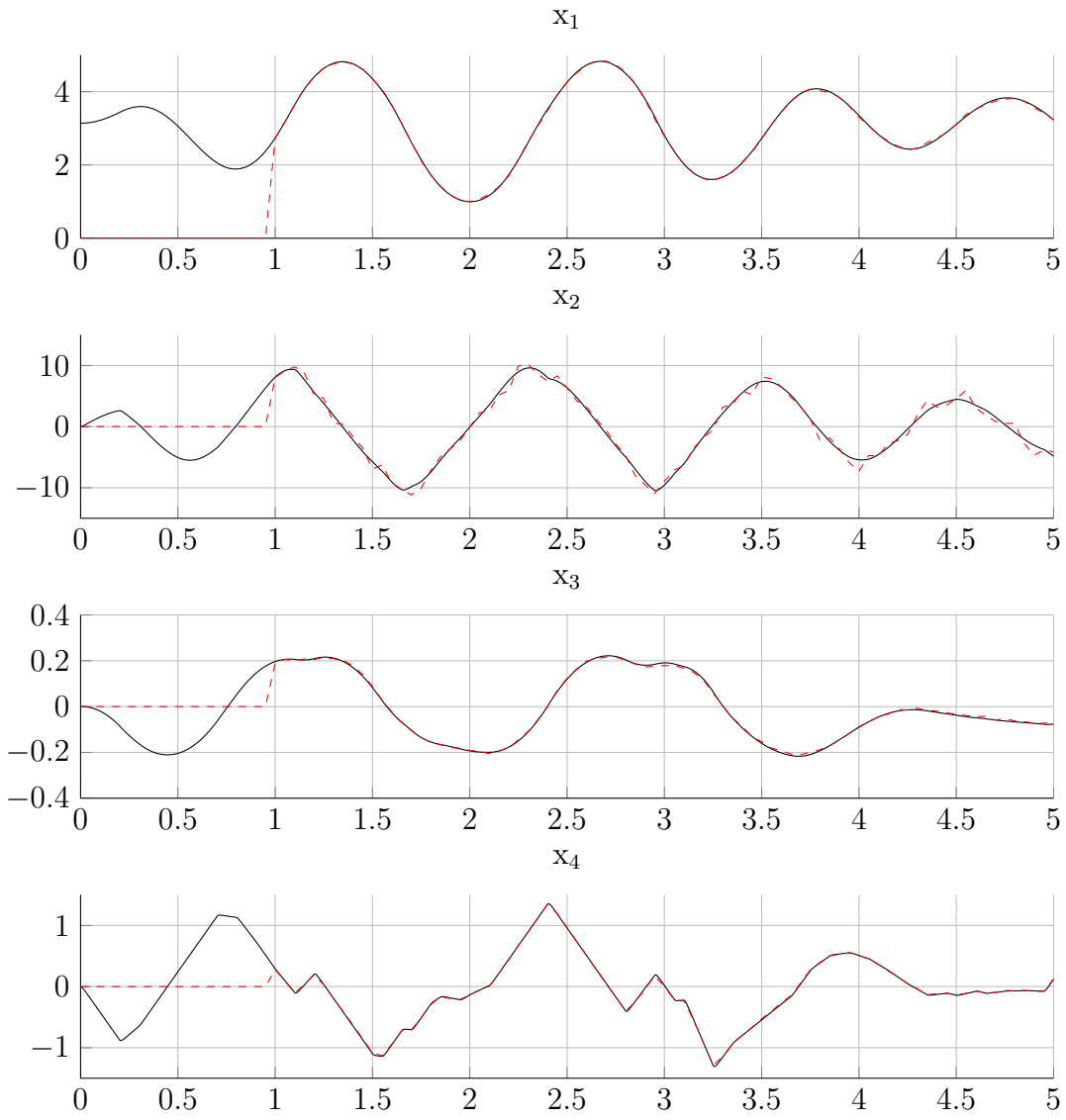

Abbildung 3.14: Ergebnis für einen Beobachter mit $N = 20$ und $n_b = 20$. Der längere Horizont und die größere Basis verbessern das Ergebnis, es sind trotzdem noch leichte Fehler sichtbar.

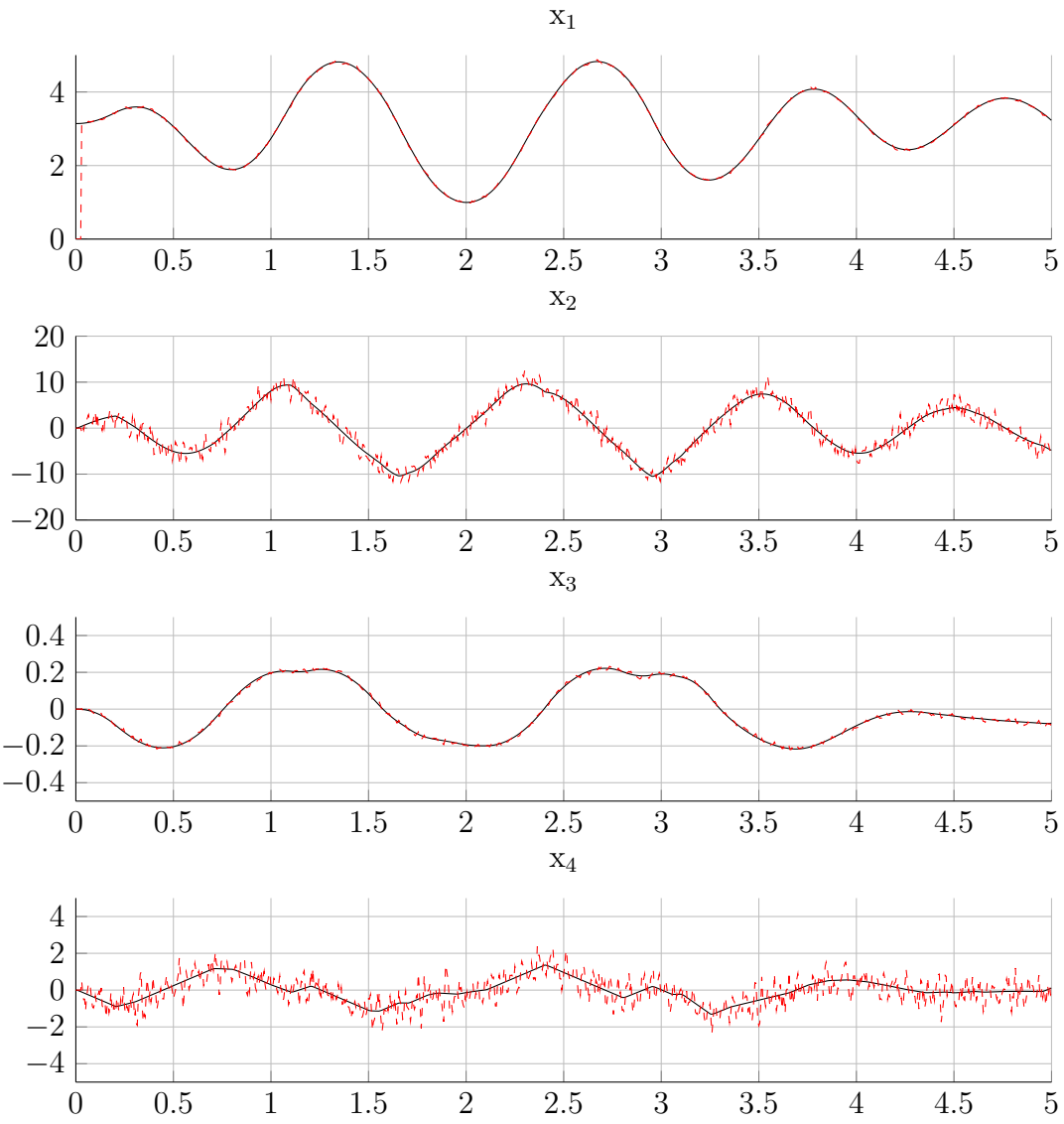

Abbildung 3.15: Der Beobachter für $N = 6$ und $n_b = 12$ mit einer verringerten Abtastrate des Systems von 5 ms. Die Genauigkeit der Schätzung nimmt gegenüber der längeren Abtastrate deutlich ab.

3.4 Beobachter in der Praxis

Wir wollen nun den Beobachter auch am inversen Pendel des Lehrstuhls einsetzen. Dazu wurde am Pendel eine Reihe von Messungen mit vorgegebenen Kontrollfolgen durchgeführt. Mit den aufgezeichneten Daten des Systemausgangs wurde anschließend der beobachtete Zustand berechnet und mit dem tatsächlichen Zustand verglichen. Man stößt hier allerdings noch auf einige kleinere Probleme, die beim Einsatz in der Simulation nicht auftreten.

Zum einen erhält man nicht exakt alle 50 ms einen Messwert. Stattdessen hat man eine Reihe von Messwerten, die, wie bereits erwähnt, im Schnitt etwa 0.13 ms auseinander liegen. Wir nehmen hier einfach den letzten Messwert unmittelbar vor der 50 ms-Marke in den Horizont auf (vgl. Abbildung 3.16). Alternativ könnte man auch die Messwerte in einem Intervall um den aktuellen Zeitpunkt interpolieren, damit man einen genaueren Messwert erhält. Die leichte zeitliche Abweichung bei der Wahl des letzten Wertes bereitet aber in der Praxis keine Probleme, der Messwert ist in der Regel nah genug am tatsächlichen Wert.

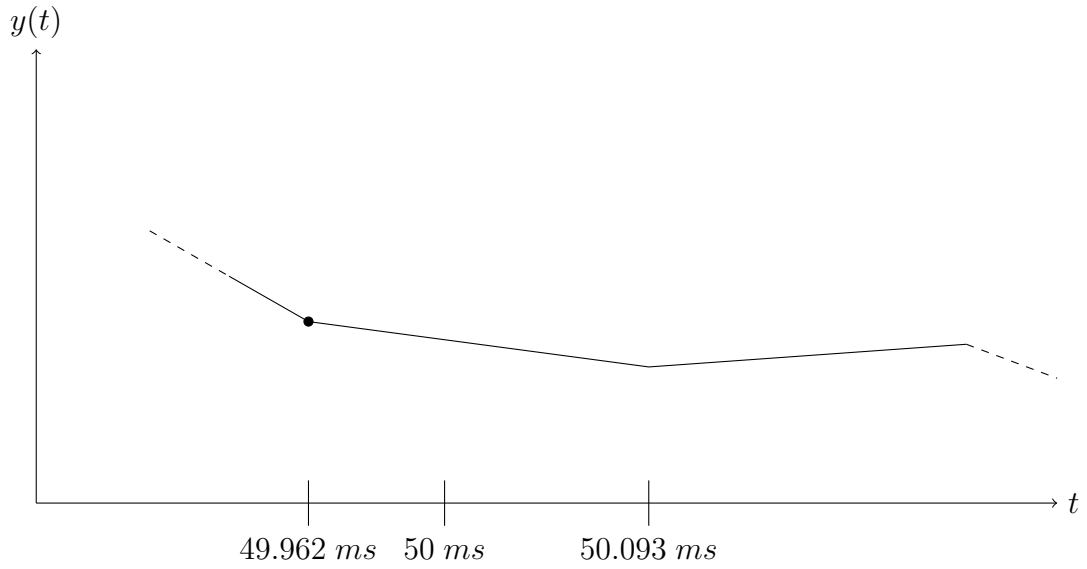

Abbildung 3.16: Der markierte Messwert wird anstelle des unbekannten „richtigen“ Wertes in den Horizont aufgenommen.

Außerdem wird die Kontrollsequenz nicht genau wie vorgegeben umgesetzt. Einzelne Kontrollen werden ein wenig länger oder kürzer angewendet, als die vorgegebene Zeitspanne (siehe Tabelle 3.3). Dies kann dazu führen, dass im Horizont eine falsche Kontrolle zu einem Messwert gespeichert wird. Ist nämlich in einem Intervall die Kontrolle zu kurz, so ist zum Zeitpunkt der Messung schon eine neue Kontrolle gesetzt, die eigentlich erst zum nächsten Zeitschritt gehört. Wir nehmen daher nur genau die vorgegebenen Kontrollen in den Horizont auf, auch wenn diese leicht verzögert sind.

Eine weitere Schwierigkeit ergibt sich dadurch, dass die Standardabweichung des Fehlers bei der Messung von Position und Winkel nicht bekannt ist, die für eine realitätsnahe

Tabelle 3.3: Zeitverzögerte Umsetzung der Kontrolle am Pendel des Lehrstuhl. Die einzelnen Beschleunigungsphasen sind nicht genau 50 ms lang, sondern können etwas länger oder kürzer ausfallen. Außerdem wird die erste Kontrolle erst gut 1 ms nach Beginn der Zeitmessung angewendet.

Zeit in s	Beschleunigung in $\frac{m}{s^2}$	Dauer in s
+0.001228016	+1.000000000	+0.050395403
+0.051623419	+2.000000000	+0.049995349
+0.101618768	-1.500000000	+0.049990694
+0.151609462	+1.000000000	+0.049990718
+0.201600179	+3.000000000	+0.049991313
+0.251591492	-1.000000000	+0.049995463
+0.301586956	-1.700000000	+0.049991075
+0.351578031	-2.300000000	+0.050200431
⋮	⋮	⋮

Simulation aber notwendig ist. Wir nehmen zur Vereinfachung an, dass die Standardabweichung gerade der maximalen Auflösung der Sensoren entspricht. Für die Position liegt diese bei ca. 0.000268609 $m \approx 0.3\ mm$, der Winkelsensor hat eine Auflösung von etwa 0.002617994 $rad \approx 0.15°$. Um Modellfehler[1] und andere nicht beachtete Effekte abfangen zu können, erhöhen wir die theoretische Messgenauigkeit noch um etwa 30 Prozent. Der Ausgang des Systems wird wie im vorherigen Abschnitt durch Addition entsprechend verteilter Zufallsgrößen simuliert.

Zum Vergleich der Ergebnisse des Beobachters mit dem tatsächlichen Zustand liegen zunächst nur die gemessenen Winkel- und Positionsdaten des Pendels vor. Um auch die Genauigkeit der beobachteten Geschwindigkeiten einschätzen zu können, müssen wir diese erst numerisch bestimmen. Die Geschwindigkeit ist gerade die Änderung des Winkels bzw. der Position über die Zeit, also können wir diese über die Ableitung der Kurven der Messwerte berechnen. Da allerdings nur diskrete Messpunkte vorliegen, erhalten wir lediglich eine Approximation der Geschwindigkeiten.

Eine einfache Methode zur Berechnung der Steigung ist, die Messwerte lokal durch ein Polynom zu approximieren (vgl. Abbildung 3.17) und dieses anschließend abzuleiten. Wir verwenden dazu die MATLAB-Routine `polyfit`, um ein quadratisches Polynom zu erhalten, welches im kleinste-Quadrate-Sinn optimal ist. Mit `polyder` berechnen wir die Ableitung dieses Polynoms, die nur noch an den Zeitpunkten der Messung ausgewertet werden muss. Statt alle Messwerte auf einmal zu verarbeiten, betrachten wir einzelne Abschnitte mit jeweils 200 Messwerten, die sich am Rand überlappen. Dadurch vermeidet man zu starke Sprünge an den Intervallgrenzen.

Für die Geschwindigkeit des Wagens erhält man so eine anschaulich recht gute Approximation. Die Winkelgeschwindigkeit weist vereinzelt noch ein etwas unregelmäßiges

[1] Das aktuelle Modell des inversen Pendels ist zwar schon sehr genau, es wird aber beispielsweise die Haftreibung noch nicht berücksichtigt. Auch die Konstanten sind nicht völlig exakt.

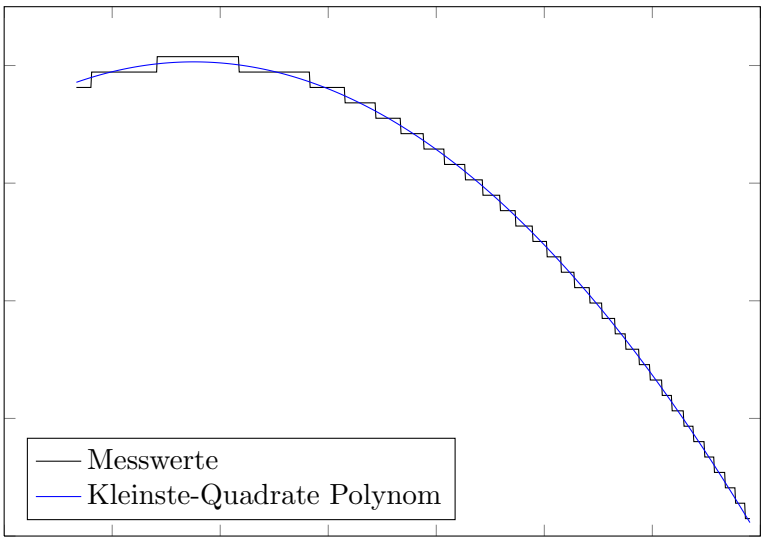

Abbildung 3.17: Polynomfitting der Messwerte mit der Methode der kleinsten Quadrate. Aus der Ableitung des Polynoms kann eine numerische Approximation der Geschwindigkeit errechnet werden.

Verhalten auf, das sich auf die etwas grobe Auflösung des Winkelsensors zurückführen lässt. Der ungefähre Verlauf ist aber dennoch gut erkennbar.

Für den Test des Beobachters wurden Messungen mit verschiedenen Kontrollfolgen durchgeführt. Wir wollen hier beispielhaft die Ergebnisse eines Versuchs vorstellen, bei dem der Wagen abwechselnd nach links und rechts beschleunigt wird und zwischendurch auch immer wieder stehen bleibt. Die verwendete Kontrollsequenz ist in Abbildung 3.18 dargestellt.

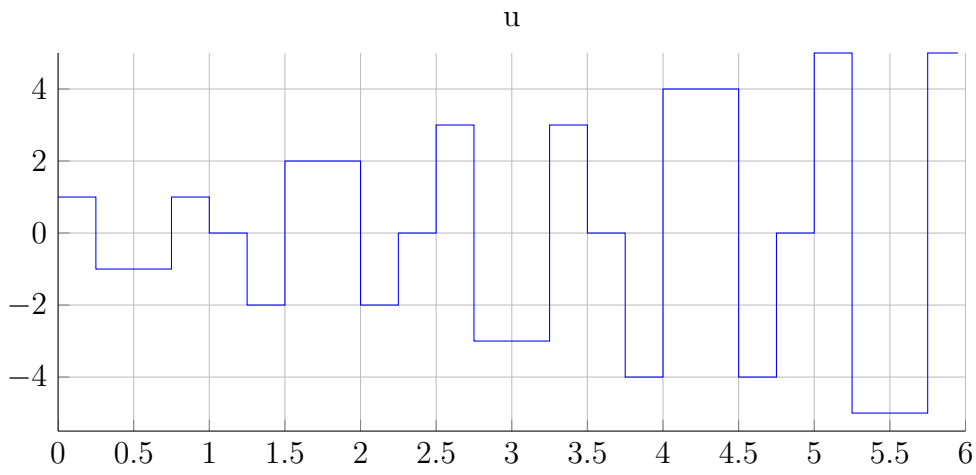

Abbildung 3.18: Verwendete Kontrollsequenz in der praktischen Anwendung des Beobachters für das inverse Pendel.

Abbildung 3.19 zeigt das Ergebnis der Zustandsschätzung für einen Beobachter mit

Horizontlänge $N = 6$ und $n_b = 12$ Basisfunktionen und zum Vergleich die gemessenen Zustandskomponenten x_1 und x_3, sowie die daraus berechneten Geschwindigkeiten x_2 und x_4.

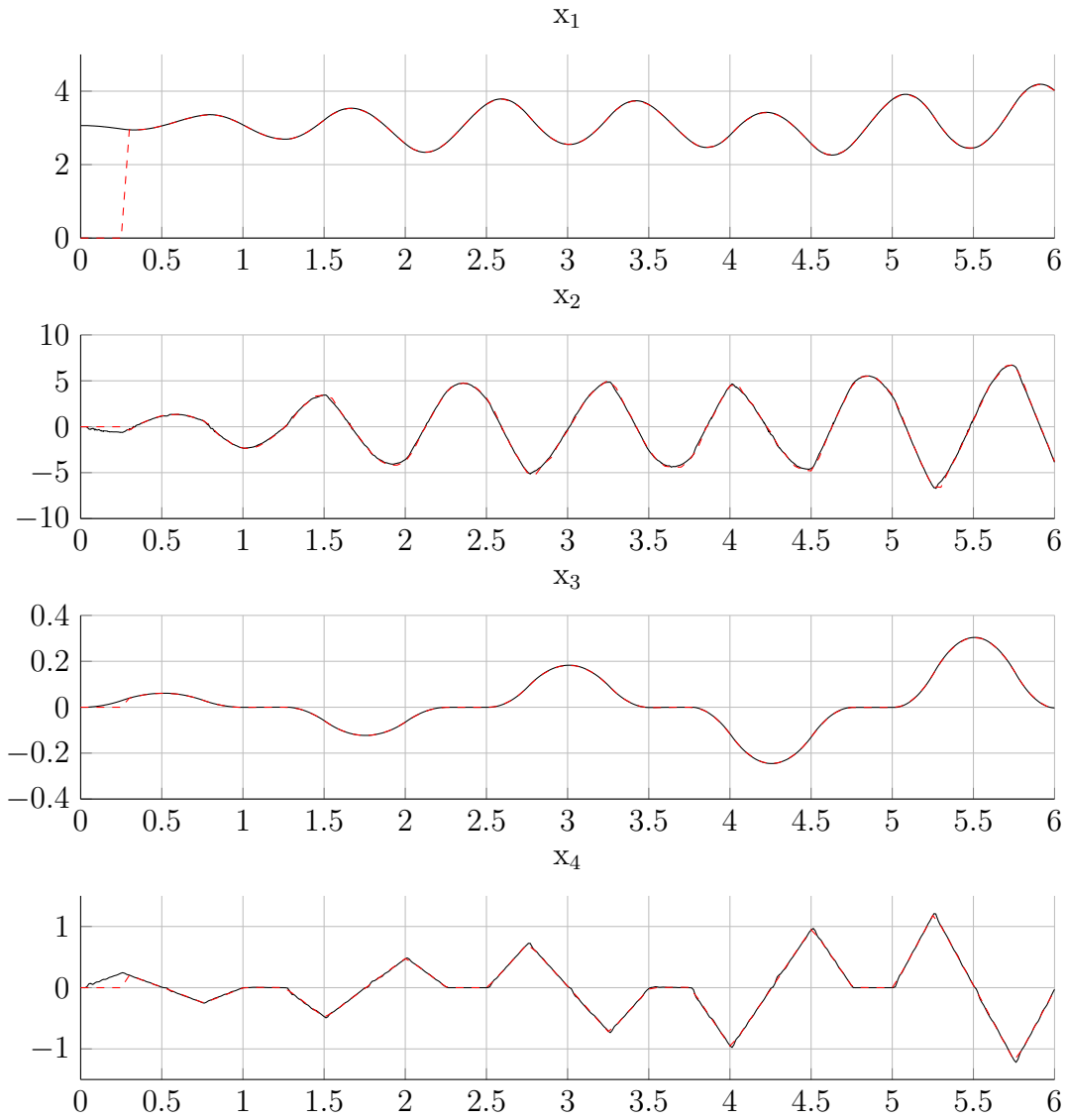

Abbildung 3.19: Das Ergebnis beim praktischen Einsatz des Beobachters mit Horizontlänge $N = 6$ und Basisgröße $n_b = 12$. Der Zustand wird offenbar gut approximiert. Nur an den Extrema der 2. Komponente sind leichte Abweichungen zu erkennen.

Wie man sieht, werden x_1 und x_3 sehr genau wiedergegeben. Auch die Geschwindigkeit des Wagens wird präzise geschätzt. Bei der Schätzung der Winkelgeschwindigkeit sind vereinzelt noch Unterschiede zu erkennen. Vor allem an den lokalen Extrema scheint die Zustandsschätzung etwas abzuweichen. Möglicherweise liegt das auch nur an der ungenauen Approximation der Ableitung aus den Messwerten.

Insgesamt liefert der Beobachter eine durchaus brauchbare Schätzung. Ob die erreichte Genauigkeit ausreichend für die Anwendung in Kombination mit Modellprädiktiver Regelung ist, muss sich allerdings noch zeigen. Man kann gegebenenfalls analog zu den vorherigen Abschnitten eine Verlängerung des Horizonts und die Verwendung einer größeren Basis in Betracht ziehen.

Eine weitere Verringerung des Fehlers könnte man vermutlich auch erreichen, indem man die Messwerte durch geeignete Filter manuell aufbessert und damit die zufälligen Messfehler etwas abschwächt.

4 Diskussion der Ergebnisse

Bevor wir zum Fazit kommen, sollen nun die Stärken und Schwächen des betrachteten Beobachters anhand der bisherigen Ergebnisse diskutiert werden.

4.1 Vorteile des Beobachters

Der wohl größte Vorteil des Beobachters auf beweglichem Horizont ist die schnelle Berechnung des geschätzten Zustandes. Der geringe Overhead rührt daher, dass die Identifikation des Beobachters offline stattfindet. Die Optimierung wird also bereits im Vorfeld erledigt und schlägt sich nicht auf die spätere Anwendung nieder. Durch geeignete Annahmen wurde außerdem erreicht, dass das resultierende Optimierungsproblem konvex und damit einfach zu lösen ist.

Für den Einsatz des Beobachters muss lediglich ein eindimensionales Nullstellenproblem gelöst werden, was dank des Newton-Verfahrens sehr effizient möglich ist. Auch der Speicheraufwand ist relativ gering. Benötigt werden nur der Horizont aus den gemessenen Ausgängen und Kontrollen, sowie für jede beobachtete Komponente die Matrizen L und μ, die den Beobachter definieren. Dadurch sollte es möglich sein, den Beobachter auch in Umgebungen mit sehr begrenzten Ressourcen einzusetzen.

Andere Beobachter verlangen dagegen meist nach erheblich aufwendigeren Techniken für die Zustandsrekonstruktion. Bei der in der Einleitung erwähnten Moving Horizon Estimation muss in jedem Zeitschritt ein Optimierungsproblem gelöst werden.[7] Der erweiterte Kalman Filter setzt eine Linearisierung voraus und erfordert fortwährend die Lösung einer algebraischen Riccati-Gleichung.[1]

Ein weiterer Vorteil des vorgestellten Beobachters liegt darin, dass es denkbar einfach ist einen reduzierten Beobachter umzusetzen, der nur die nicht bereits gemessenen Zustandsgrößen rekonstruiert. Da man ohnehin für jede Komponente gesondert den Zusammenhang zwischen Horizont und tatsächlichem Zustand identifiziert, kann man sich direkt darauf beschränken, das nur für die unbekannten Zustände zu tun. Im Fall des inversen Pendels aus dem vorherigen Kapitel hätte man beispielsweise darauf verzichten können, die gemessenen Komponenten x_1 und x_3 zu beobachten.

Weil der Beobachter die einzelnen Komponenten des Zustandsvektors unabhängig voneinander berechnet, kann man sogar noch einen Schritt weiter gehen und für jede Komponente einen eigenen Beobachter verwenden. Beim Pendel hat der Beobachter beim Schätzen der x_2-Komponente in der Regel die größten Probleme, wodurch eine große Funktionenbasis und ein langer Horizont nötig werden. Die anderen Komponenten werden aber bereits mit deutlich weniger Aufwand präzise approximiert. Es würde also Sinn machen, mehrere Teilbeobachter einzusetzen, die genau auf die jeweils beobachtete

Komponente zugeschnitten sind. So erhält man auch in schwierigen Fällen eine gute Schätzung, während man anderswo keinen unnötigen Aufwand betreibt.

Zusätzlich kann man die Teilbeobachter parallel implementieren, wodurch sich ein weiterer Geschwindigkeitsvorteil ergibt.

4.2 Probleme

Es gibt allerdings auch eine Reihe von Problemen, die den allgemeinen Einsatz des Beobachters erschweren können.

An erster Stelle steht hier die Notwendigkeit eines exakten Systemmodells. Ohne das wäre es nicht möglich, die Lerndaten durch Simulationen zu generieren. Im Prinzip könnte man die Daten zwar auch aus vielen realen Messungen erhalten, das wäre aber mit erheblich mehr Aufwand verbunden und ist auch nicht immer möglich. Der Nachteil relativiert sich insofern, als dass auch andere Arten von Beobachtern ein Systemmodell voraussetzen.

Wenn ein ausreichend genaues Modell vorliegt, muss das Kontrollsystem für eine Vielzahl von Startwerten und Kontrollen simuliert werden. Dieser Schritt ist der mit Abstand zeitintensivste im MATLAB-Programm. Selbst für eine niedrige Anzahl von 500 Startwerten und einer maximalen Horizontlänge von 30 kann die Simulation beim inversen Pendel mehrere Minuten in Anspruch nehmen. Prinzipiell muss das aber nur ein einziges Mal gemacht werden. Anschließend kann man die Daten nutzen, um daraus Lerndaten für unterschiedliche Horizontlängen zu extrahieren.

Nachteilig sind außerdem die schlechteren Ergebnisse des Beobachters bei gestörten Systemen. Man kann dieser Schwierigkeit begegnen, indem man die Störung schon in die Simulation mit einbezieht. Ein optimales Ergebnis setzt aber möglichst genaue Kenntnis über Art und Ausmaß der Störung des Systemausgangs voraus, Informationen, die nicht immer zur Verfügung stehen. Im Beispiel des Pendels haben wir auch mit Schätzwerten der Streuung der Messwerte gute Resultate erzielen können.

Ein weiteres Problem ist, dass der Beobachter nur lokal einsetzbar ist. Dies ist der Tatsache geschuldet, dass man sich bei der Simulation auf einen bestimmten Bereich des gesamten Zustandsraums einschränken muss. Außerhalb wird der Zustand nicht korrekt rekonstruiert. Im Allgemeinen wird eine Transformation des Horizonts wie beim inversen Pendel nicht möglich sein. Einzige Möglichkeit bleibt somit, das betrachtete Gebiet entsprechend zu vergrößern, um den Nachteil etwas abzumildern. Es werden dadurch aber auch mehr Trainingsdaten nötig, um den größeren Raum ausreichend abzudecken. Das wiederum verringert die Genauigkeit der Schätzung, da die Lösung des QP-Problems ungenauer wird. Andere Beobachter, wie beispielsweise der Luenberger Beobachter, sind zumindest im Fall linearer Kontrollsysteme global einsetzbar, beim vorliegenden Beobachter tritt selbst hier das Problem auf.

Der gravierenste Nachteil ist wohl die Vielzahl von Parametern, die mitunter entscheiden-

den Einfluss auf die Präzision der Zustandsschätzung haben. Das beginnt bei der Frage nach der Anzahl der verwendeten Lerndaten, über die Horizontlänge und Eigenschaften der Funktionenbasis bis zu Parametern im QP-Problem, wie die Feinheit des Gitters für die Ungleichheits-Nebenbedingung oder den verwendeten Lösungsalgorithmus. Für sich genommen sind viele Parameter natürlich nichts Schlechtes, denn sie können dazu dienen den Beobachter genau für das vorliegende Problem maßzuschneidern. Jedoch fehlen für den vorliegenden Beobachter Anhaltspunkte dafür, welche Parameterwahl zu einem möglichst guten Ergebnis führt.

Erschwerend kommt hinzu, dass man selbst dann nicht unbedingt eine brauchbare Schätzung erhält, wenn man in der Lage ist, den Beobachter für eine große Zahl von Parametern zu testen. Durch die Annahme (2.4) wird die universelle Verwendbarkeit des Beobachters eingeschränkt. Falls sich der Zustand nicht in geeigneter Weise aus dem Horizont rekonstruieren lässt, ist der Beobachter nicht einsetzbar.

Beim Pendel konnten wir beobachten, dass ein längerer Horizont und eine größere Basis das Ergebnis in der Regel verbessern. Dabei handelt es sich aber um eine Heuristik, die sich durch Ausprobieren verschiedenster Parameterkombinationen abzeichnete. Es fehlt ein allgemeines Kriterium, das beschreibt, unter welchen Voraussetzungen man eine sinnvolle Zustandsbestimmung erwarten kann. Im Moment ist also noch viel manuelles Feintuning des Beobachters notwendig.

5 Fazit und weitere Ideen

Ziel dieser Arbeit war es, einen Ansatz für einen Beobachter auf beweglichem Horizont vorzustellen, der auch für den Einsatz bei beschränkten Rechenkapazitäten geeignet ist. Der Grundgedanke war, eine Alternative zu einem MHE-Beobachter zu entwickeln, bei der sämtliche Optimierung offline stattfindet.

Es wurden dazu zunächst die theoretischen Grundlagen für den Beobachter erarbeitet. Wir sahen, dass der Beobachter mathematisch durch eine nichtlineare Funktion beschrieben wird, die vom Horizont auf den Zustand abbildet. Für die numerische Approximation dieser Funktion wurde auf eine Darstellung durch geeignete Basisfunktionen zurückgegriffen. Die Identifikation erfolgte unter Verwendung von Referenzdaten, die durch Simulationen gewonnen wurden. Formal führte das auf ein kleinste-Quadrate-Problem, das durch leichte Modifikationen in ein einfach zu lösendes QP-Problem überführt werden konnte. Außerdem wurde geschildert, dass zur Anwendung des Beobachters die Lösung eines Nullstellenproblems erforderlich ist.

Im zweiten Teil der Arbeit haben wir den Beobachter in der Anwendung für das inverse Pendel auf dem Wagen untersucht. Die Grundlage bildete hier ein detailliertes Modell des Systems aus einer Diplomarbeit. Anhand von numerischen Simulationen wurde vorgeführt, wie die Zustandsschätzung auf unterschiedliche Parameter in der Implementierung reagiert und welche Probleme dabei auftreten können. Wir konnten sehen, dass der Beobachter eine hohe Genauigkeit aufweist, die durch gestörte Systeme aber verschlechtert wird. Anschließend wurde der Beobachter auch am realen Pendel eingesetzt und die Genauigkeit der Schätzung durch den Vergleich von Messwerten und berechneten Zuständen überprüft.

Der letzte Teil der Arbeit diskutierte die erzielten Resultate und ging auf Vorteile und Probleme ein. Es wurde ausgeführt, dass beim betrachteten Beobachter der Schwerpunkt auf guter Performance und ressourcenschonender Anwendung liegt. Bis man allerdings ein zufriedenstellendes Ergebnis erhält, kann unter Umständen einiger Aufwand zum Finden geeigneter Parameter nötig sein.

Zusammenfassend kann man sagen, dass der Beobachter tätsächlich die erwarteten Geschwindigkeitsvorteile bringen kann. Da ein rigoroser mathematischer Konvergenzbeweis noch aussteht, ist man im Moment aber auf eine Reihe von Heuristiken und Intuition bei der Parameterwahl angewiesen. Insbesondere stellt der Beobachter keinen bedingungslosen Ersatz für andere Techniken wie EKF oder MHE dar. Stattdessen bietet sich der Einsatz gerade dort an, wo andere Ansätze aufgrund technischer Limitierungen nicht anwendbar sind.

Zum Schluss werden noch einige weiterführende Ideen kurz geschildert, die zum Ziel haben, die Schwächen des Beobachters etwas abzumildern oder an seine Stärken anzuknüpfen.

Unabhängig davon wäre es zudem interessant, den erstellten Beobachter für das inverse Pendel zur Stabilisierung mittels MPC zu nutzen.

Wie erwähnt leidet der Beobachter unter dem Problem, dass es keine klaren Vorgaben für die Wahl der Parameter gibt. Passende Parameter für das Pendel wurden in der Arbeit stets durch Ausprobieren gefunden.

Eine erste Verbesserung wäre es, dieses Ausprobieren zu automatisieren, z. B. indem man über verschiedene Horizontlängen, Basisgrößen und Schrittweiten iteriert und die Resultate vergleicht. Die Einfachheit des QP-Problems sollte auch ein effizientes systematisches Vorgehen ermöglichen, bei dem die Beobachterparameter durch Optimierung bestimmt werden. Die Zielfunktion wäre dabei ein Maß für den Schätzfehler des Beobachters, der minimiert werden soll. Durch geeignete Nebenbedingungen ließe sich außerdem erreichen, dass das Ergebnis bestimmte Vorgaben erfüllt. Beispielsweise könnte man die Basisgröße beschränken, um praktische Verwendbarkeit zu gewährleisten.

In der bisherigen Form wird der Beobachter vor der ersten Anwendung definiert und bleibt dann für alle Zeit unveränderlich bestehen. Dies hat den Nachteil, dass nicht auf Veränderungen des Systems reagiert werden kann, wenn sich z. B. durch Verschleiß die Reibung ändert.

Denkbar wäre es nun, den Beobachter so zu modifizieren, dass er sich im laufenden Betrieb selbstständig an das neue Modell anpasst. Natürlich müsste der Beobachter dazu in der Lage sein, die Abweichung der Schätzung zu erkennen, was ein erstes Hindernis darstellt. Dies könnte man dadurch lösen, dass man gelegentlich den Systemzustand auf andere Weise rekonstruiert. Beim Pendel wäre es beispielsweise möglich den Zustand mit der in Abschnitt 3.4 beschriebenen Methode des Polynomfittings zu bestimmen. So würde man auch online neue Trainingsdaten für den Beobachter erhalten. Die neuen Daten könnten dann zu den bisherigen hinzugenommen und das QP-Problem erneut gelöst werden.

Man könnte auch erwägen, Trainingsdaten veralten zu lassen, also für die Matrix Q aus (2.13) nicht mehr zu berücksichtigen, da diese ja nicht mehr dem aktuellen Systemmodell entsprechen. Da das Update nur verhältnismäßig kleine Änderungen an Q bewirkt, müsste das QP-Problem zudem eventuell nicht jedes Mal von Grund auf neu gelöst werden. Stattdessen könnte ausgehend von der alten Lösung schnell eine neue gefunden werden.

Schließlich bestünde noch die Möglichkeit, eine Verschmelzung von Beobachter und Feedback zu versuchen. Die Idee dahinter wäre, nicht den aktuellen Zustand zu schätzen, von dem aus dann eine Kontrolle berechnet wird. Stattdessen würde der neue „Beobachter“ so trainiert, dass er direkt eine stabilisierende Kontrolle für das System im nächsten Zeitschritt liefert. Es würde also direkt die Verknüpfung $\gamma \circ \mathcal{F}$ der Funktionen $\mathcal{F}$, die den Horizont auf den aktuellen Zustand abbildet, und des Feedbacks γ identifiziert.

Ob und wann dieser Ansatz zum Erfolg führt, ist noch unklar und bedarf weiterer Untersuchungen. Eine Herausforderung würde dabei auch die Generierung der Referenzdaten darstellen, die zum Training verwendet werden. Schon jetzt war dieser Schritt sehr zeitintensiv, nun käme als zusätzlicher Aufwand hinzu, für jeden erzeugten Horizont eine

Kontrolle zu berechnen.

Sollte es gelingen, das in der Arbeit vorgestellte Konzept geeignet zu erweitern und die besprochenen Nachteile zu vermeiden, könnte man eine sehr effiziente Regelungsmethode erwarten.

A Anhang

Inhalt der beiliegenden CD

Die CD enthält neben einer PDF-Version dieser Arbeit eine MATLAB-Implementierung des vorgestellten Beobachters, sowie eine C++-Klasse, die als Grundlage für den Einsatz des Beobachters in praktischen Anwendungen dienen kann. Die einzelnen Funktionen sind ausführlich dokumentiert. Außerdem enthalten ist eine Reihe von Beispielen, die die Verwendung des Beobachters demonstrieren.

Im Detail sind folgende Dateien und Ordner vorhanden:

`/BA_Simon_Pirkelmann.pdf`	PDF-Version dieser Arbeit
/CPP	Ordner mit C++-Quellcodedateien
`/CMHO.cpp`	Datei mit der Implementierung der Methoden
`/CMHO.h`	Headerfile einer Klasse zur Rekonstruktion des Zustands gemäß Abschnitt 2.2.3
/MATLAB	Ordner mit MATLAB-Routinen
`/inv_pendel_pt1_ode.m`	Differentialgleichungssystem des inversen Pendels
`/inv_pendel_pt1_output.m`	Nachbildung des Ausgangs für das inverse Pendel (ohne Störung)
`/inv_pendel_pt1_output_noise.m`	Nachbildung des Ausgangs für das inverse Pendel (mit Störung)
`/num_derivate.m`	Numerische Berechnung der Ableitungen zur Bestimmung der Geschwindigkeiten (vgl. Abschnitt 3.4)
`/observer_pend_app.mat`	Beobachter für die Anwendung mit realen Messwerten (Sicherung des MATLAB-Workspace)
`/test_obs.m`	Funktion zum grafischen Vergleich des vom Beobachter geschätzten Zustandes mit dem tatsächlichen Zustand bzw. den Messungen
`/test_obs_app_pendel.m`	Test des Beobachters in der Anwendung mit realen Messwerten

`/test_pendel_pt1.m`	Beispiel für den Beobachter beim Pendel ohne Störungen des Systemausgangs
`/test_pendel_pt1_noise.m`	Beispiel für den Beobachter beim Pendel mit Störungen des Systemausgangs
/MATLAB/data	Unterordner mit Beschleunigungsprofilen und zugehören Messwerten zu Winkel und Wagenposition vom Pendel des Lehrstuhls
/MATLAB/+mho	Unterordner mit der Implementierung des Beobachters
`/calc_state_component.m`	Rekonstruktion einer Zustandskomponente gemäß Abschnitt 2.2.3
`/simulate_control_system.m`	Simulation des Kontrollsystems
`/train_mho.m`	Gewinnung von Lerndaten aus den Simulationsdaten und Identifikation des Beobachters durch Lösung des QP-Problems

/MATLAB/+mho/+basis_functions

`/basis.m`	Implementierung der Funktionenbasis
`/diff_basis.m`	Ableitungen der Basisfunktionen
`/int_basis.m`	Integrale der Basisfunktionen

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] ADAMY, J.: *Nichtlineare Regelungen.* Springer-Verlag, Berlin, Heidelberg, 2009

[2] ALAMIR, M.: *A New Identification Framework for Off-Line Computation of Moving-Horizon Observers.* Automatic Control, IEEE Transactions on , vol.58, no.7, pp.1877,1882, July 2013

[3] BAUMANN, C.: *Analyse, Weiterentwicklung und Optimierung eines Simulationsmodells für ein inverses Pendel.* Diplomarbeit, Universität Bayreuth, 2012

[4] GRÜNE, L.: *Einführung in die Numerische Mathematik.* Vorlesungsskript, Universität Bayreuth, 2014

[5] GRÜNE, L.: *Mathematische Kontrolltheorie.* Vorlesungsskript, Universität Bayreuth, 2014

[6] NOCEDAL, J. ; WRIGHT, S.: *Numerical Optimization.* Springer-Verlag, New York u.a., 1999

[7] RAWLINGS, J. ; MAYNE, D.: *Model Predictive Control: Theory and Design.* Nob Hill Publishing, Madison, 2013

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Bayreuth, 20. April 2014

_____

(Simon Pirkelmann)