Technische Universität Dresden

Fakultät Elektrotechnik und Informationstechnik

Institut für Regelungs- und Steuerungstheorie

Studienarbeit

Approximationsmethoden für dynamische Systeme auf Basis des Maschinellen Lernens

vorgelegt von: Julius Fiedler

geboren am: 13. Oktober 1996 in Dresden

Betreuer:	Dr.-Ing. C. Knoll
Verantwortlicher Hochschullehrer:	Prof. Dr.-Ing. habil. Dipl.-Math. K. Röbenack
Tag der Einreichung:	27. Oktober 2020

Bitte ersetzen Sie diese Seite vor dem Binden mit der Aufgabenstellung.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage an der Fakultät Elektrotechnik und Informationstechnik eingereichte Studienarbeit zum Thema

Approximationsmethoden für dynamische Systeme auf Basis des Maschinellen Lernens

selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften entnommen sind, wurden als solche kenntlich gemacht.

Dresden, 27. Oktober 2020 Julius Fiedler

Kurzfassung

Systemmodelle sind eine Grundvoraussetzung in der Regelungstechnik. Um analytische Zusammenhänge verstehen zu können, sind White-Box-Modelle von Nöten. *Sparse Identification of Nonlinear Dynamics* (SINDy) ist ein Verfahren, um die zugrundeliegenden Differentialgleichungen eines Systems zu bestimmen. SINDy wird an Systemen unterschiedlicher Komplexität getestet und es werden Richtlinien für eine effektive Nutzung abgeleitet. Es werden Methoden aufgezeigt, um die Genauigkeit des Algorithmus zu erhöhen und im Voraus bestehendes Wissen einfließen zu lassen.

Abstract

Having access to system models is essential in control theory. To understand the underlying analytical equations of a system, a white-box-model is needed. *Sparse Identification of Nonlinear Dynamics* (SINDy) is a method to identify the governing equations of a system from its measurement data. SINDy is tested on systems of different complexity to establish guidelines for an effective use. Furthermore, techniques are shown to enhance the accuracy of the algorithm and make use of prior knowledge of the system.

Inhaltsverzeichnis

Verzeichnis der Formelzeichen VII

Verzeichnis der Abkürzungen VIII

Abbildungsverzeichnis IX

1 Einleitung 1
- 1.1 Motivation . . . 1
- 1.2 Präzisierung der Aufgabenstellung . . . 1

2 Der SINDy-Algorithmus 2
- 2.1 Vorbetrachtungen . . . 2
- 2.2 Funktionsweise der Methode . . . 2
- 2.3 Lösen des Minimierungsproblems . . . 3
- 2.4 Rekursive Methode der kleinsten Quadrate . . . 4

3 Durchführung von Tests 6
- 3.1 Vorstellung der Implementationen . . . 6
- 3.2 Vorstellung der Testsysteme . . . 6
- 3.3 Einfluss ausgewählter Parameter . . . 8
  - 3.3.1 Anmerkungen zur Auswertung . . . 8
  - 3.3.2 Einfluss der Simulationszeit . . . 11
  - 3.3.3 Einfluss der Simulationsschrittweite . . . 11
  - 3.3.4 Einfluss der Gestaltung der Bibliothek . . . 12
  - 3.3.5 Einfluss von verrauschten Daten . . . 12
  - 3.3.6 Rechenzeit . . . 14
  - 3.3.7 Zusammenfassung der Ergebnisse . . . 14
- 3.4 Verbesserung der Ergebnisse . . . 15
  - 3.4.1 Arbeit mit mehreren Trajektorien . . . 15
  - 3.4.2 Identifikation von teilweise bekannten Systemen . . . 15

4 Anwendung auf komplexes Beispiel - Inverses Pendel 17
- 4.1 Einleitung . . . 17
- 4.2 Aufbau der Bibliothek . . . 18
- 4.3 Bewertung der Ergebnisse . . . 19

5 Zusammenfassung und Ausblick **21**
5.1 Zusammenfassung 21
5.2 Ausblick 21

Verzeichnis der Formelzeichen

$C^r(n,k)$	Kombination mit Wiederholung ε
ε_r	relativer Fehler der identifizieren Koeffizienten
$\boldsymbol{f}$	Systemdynamik
F	Kraft, die auf den Wagen wirkt
g	Erdbeschleunigung
m	Anzahl der Messungen
m_1	Masse des Wagens
m_2	Masse am Ende des Pendels
n	Anzahl der Zustände
L	Anzahl der Ansatzfunktionen in Θ
$\boldsymbol{p}$	Parametervektor
P	nominale Koeffizientenmatrix
φ	Auslenkung des Pendels aus der unteren Ruhelage
Q	identifizierte Koeffizientenmatrix
R	Hilfsmatrix zur Berechnung von ε_r
s	Länge des Pendels
t	Zeit
θ	Gruppe von Ansatzfunktionen
Θ	$\Theta(X)$
$\Theta(X)$	Bibliotheksmatrix aus Ansatzfunktionen, ausgewertet für alle von X
Θ^+	Moore-Penrose-Inverse von Θ
Θ_v	verkleinerte Bibliotheksmatrix
U	Unbekannte Funktion
x	Position des Wagens
$\boldsymbol{x}(t) \in \mathbb{R}^n$	Zustandsvektor
X	Matrix der zeitlichen Verläufe der Zustandskomponenten
$\dot{X}$	Matrix der zeitlichen Verläufe der Ableitungen der Zustandskomponenten
ξ	Spalte von Ξ
Ξ	Koeffizientenmatrix
Ξ^d	dünn besetzte Koeffizientenmatrix
$\hat{\Xi}$	verkleinerte Koeffizientenmatrix unter Nutzung von Θ_v
Ξ^n	Koeffizientenmatrix in den Dimensionen von Ξ mit den Einträgen von $\hat{\Xi}$

Verzeichnis der Abkürzungen

DGL	Differentialgleichung
MKQ	Methode der kleinsten Quadrate
SINDy	System Identification of Nonlinear Dynamics
STLSQ	Sequentially Thresholded Least Square

Abbildungsverzeichnis

1 Simulation des Lotka-Volterra-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_r = 10\%$. 8
2 Simulation des Lotka-Volterra-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_r = 1\%$. 9
3 Simulation des Lotka-Volterra-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_r = 0.1\%$. 10
4 relativer Fehler der identifizierten Parameter in Abhängigkeit der Simulationszeit. 11
5 relativer Fehler der identifizierten Parameter in Abhängigkeit der Simulationsschrittweite. 12
6 relativer Fehler der Identifizierten Parameter in Abhängigkeit der Ordnung der Polynom-Bibliothek. 13
7 relativer Fehler der identifizierten Parameter in Abhängigkeit der Rauschstärke. 13
8 durchschnittliche Rechenzeit der unterschiedlichen Algorithmen. 14
9 relativer Fehler der identifizierten Parameter in Abhängigkeit der Anzahl der gleichzeitig verwendeten Trajektorien. 16

10 Schematische Darstellung des Wagen-Pendel-Systems, mit den Massen m_1 und m_2, der Pendellänge s, der Erdbeschleunigung g, dem Krafteingang F, der Position des Wagens x und der Auslenkung des Pendels aus der unteren Ruhelage φ. 17
11 Simulation des Wagen-Pendel-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_r = 8 \cdot 10^{-5}$. 20

Kapitel 1

Einleitung

1.1 Motivation

Für viele regelungstechnische Anwendungen ist die Kenntnis von Systemmodellen erforderlich. Unter Nutzung von Methoden des maschinellen Lernens können aus Messdaten Modelle erzeugt werden, die das Verhalten des Systems widerspiegeln können. Oft handelt es sich um Black-Box-Modelle, die zwar das Eingangs-Ausgangs-Verhalten des Systems wiedergeben, jedoch keinen Aufschluss zu den zugrunde liegenden Differentialgleichungen liefern können. Diese mathematischen Zusammenhänge können jedoch essenziell im Verständnis und im Umgang mit komplexen Systemen sein. In dieser Arbeit wird die Methode *Sparse Identification of Nonlinear Dynamics* (SINDy) untersucht, die verspricht die Systemdifferentialgleichungen aus Messdaten des Systems zu rekonstruieren.

1.2 Präzisierung der Aufgabenstellung

Aus den existierenden Methoden zur Systemidentifikation sollen vielversprechende Kandidaten ausgewählt werden. Diese sollen während der Literaturrecherche auf Eignung und Implementierbarkeit geprüft werden. Anschließend sollen wichtige Ergebnisse aus der Literatur reproduziert und weitere Beispielsysteme steigender Komplexität übertragen werden. Dabei soll der Einfluss der zur Verfügung stehenden Freiheitsgrade herausgearbeitet werden.

Kapitel 2

Der SINDy-Algorithmus

2.1 Vorbetrachtungen

Sparse Identification of Nonlinear Dynamics (SINDy) ist eine Methode, um aus den Messdaten eines Systems auf dessen Systemdifferentialgleichungen zu schließen. Die zugrundeliegende Annahme ist, dass die zu identifizierende Funktion in einem geeigneten Raum an Basisfunktionen *dünnbesetzt* ist. Betrachtet man beispielsweise die Funktion

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} 1 + 2x_1 + x_1 x_2^2 \\ x_1^3 - 3x_2 \end{bmatrix}, \tag{2.1}$$

so ist leicht zu erkennen, dass $\boldsymbol{f}$ in Bezug auf die Basis von Polynomen aus zwei Variablen (z.B. $f_1(\boldsymbol{x}) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{ij} x_1^i x_2^j$) dünnbesetzt ist. Nur eine sehr geringe Zahl der Koeffizienten a_{ij} ist ungleich null. Die dem Algorithmus zur Verfügung gestellten Basisfunktionen werden zu einer Bibliothek zusammengefasst. Hieraus werden mittels Regression diejenigen Ansatzfunktionen ausgewählt, deren Linearkombination die Funktion $\boldsymbol{f}$ am besten repräsentiert. Die Auswahl der Bibliotheksfunktionen spielt dabei eine entscheidende Rolle für den Erfolg der Methode. Daher ist es günstig, wenn man bereits im Voraus gewisse Kenntnisse zu den zu identifizierenden Funktionsklassen besitzt, um die Bibliothek geeignet auslegen zu können. Wie der Methodenname suggeriert, können mit SINDy auch nichtlineare Funktionen identifiziert werden.

2.2 Funktionsweise der Methode

Der folgende Abschnitt orientiert sich an der Beschreibung des SINDy-Algorithmus durch Brunton et al.[1].

Sei $\boldsymbol{f}\left(\boldsymbol{x}(t)\right) = \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}$ das zu identifizierende Differentialgleichungssystem mit dem Zustandsvektor $\boldsymbol{x}(t) = [x_1(t), x_2(t), ..., x_n(t)]^T \in \mathbb{R}^n$. Für die Anwendung von SINDy benötigt man die Messdaten aller Zustandsgrößen zu den Zeitpunkten $t_1, t_2, ..., t_m$. Zusätzlich müssen die zeitlichen Ableitungen der Zustände an den gegebenen Zeitpunkten

gegeben sein, entweder durch direkte Messung oder durch numerische Approximation. Die Daten werden wie folgt angeordnet:

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix} \in \mathbb{R}^{m \times n}, \tag{2.2}$$

$$\dot{X} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \dots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \dots & \dot{x}_n(t_2) \\ \vdots & \vdots & & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \dots & \dot{x}_n(t_m) \end{bmatrix} \in \mathbb{R}^{m \times n}. \tag{2.3}$$

Nun muss man die Bibliothek Θ an Funktionen konstruieren, durch welche $\boldsymbol{f}$ dargestellt werden soll. Die Spalten der Bibliotheksmatrix repräsentieren die gewählten Ansatzfunktionen, angewendet auf die Datenmatrix X

$$\Theta(X) = \begin{bmatrix} | & | & & | \\ \theta_1(X) & \theta_2(X) & \dots & \theta_\ell(X) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{m \times L}. \tag{2.4}$$

Dabei steht ℓ für die Anzahl von verschiedenen Typen von Ansatzfunktionen und L für die Gesamtzahl der Spalten von Θ, welche sich aus der Wahl der Ansatzfunktionen ergibt. Wählt man beispielsweise für θ_1 die Sinusfunktion und für θ_2 Monome zweiten Grades so ergeben sich

$$\theta_1(X) = \begin{bmatrix} | & | & & | \\ \sin(x_1(t)) & \sin(x_2(t)) & \dots & \sin(x_n(t)) \\ | & | & & | \end{bmatrix}, \tag{2.5}$$

$$\theta_2(X) = \begin{bmatrix} | & | & | & | & | \\ x_1(t)^2 & x_1(t)x_2(t) & \dots & x_2(t)^2 & x_2(t)x_3(t) & \dots & x_n^2(t) \\ | & | & | & | & | \end{bmatrix}. \tag{2.6}$$

Gesucht sind nun die Koeffizienten $\boldsymbol{\xi}_i$ der Linearkombinationen von Bibliotheksfunktionen, sodass gilt

$$f_i(x) = \Theta(\boldsymbol{x}^T)\,\boldsymbol{\xi}_i. \tag{2.7}$$

Fasst man alle $\boldsymbol{\xi}_i$ in eine Koeffizientenmatrix $\Xi \in \mathbb{R}^{L \times n}$ zusammen, so ergibt sich das zu lösende Minimierungsproblem zu

$$\dot{X} \approx \Theta(X)\Xi. \tag{2.8}$$

2.3 Lösen des Minimierungsproblems

Das Lösen des Gleichungssystems (2.8) ist die Kernaufgabe des Algorithmus. Das Gleichungssystem besitzt $m \cdot n$ Gleichungen ($m \mathrel{\hat{=}}$ Anzahl Messungen) und $L \cdot n$ Unbekannte

($L \mathrel{\hat{=}}$ Anzahl Ansatzfunktionen). Unter der Annahme, dass die Anzahl der Messungen ausreichend groß ist, handelt es sich um ein überbestimmtes Gleichungssystem. Brunton et. al. [1] schlägt einen sequentiellen Algorithmus (*Sequentially Thresholded Least Squares algorithm* STLSQ) vor. Eine Abwandlung davon wurde im Verlauf dieser Arbeit in *Python* entwickelt und wird im Folgenden vorgestellt.

2.4 Rekursive Methode der kleinsten Quadrate

Eine weit verbreitete Strategie zur Lösung überbestimmter Gleichungssysteme der Form

$$\dot{X} = \Theta\Xi \tag{2.9}$$

ist die Methode der kleinsten Quadrate. Dabei wird die Moore-Penrose-Inverse [2] von Θ berechnet

$$\Theta^{+} := (\Theta^{T}\Theta)^{-1}\Theta^{T}. \tag{2.10}$$

Die Moore-Penrose-Lösung

$$\Xi = \Theta^{+}\dot{X} \tag{2.11}$$

minimiert dabei den Fehler

$$\varepsilon := \left\|\Theta\Xi - \dot{X}\right\|_2. \tag{2.12}$$

Der Algorithmus beginnt mit dieser Näherungslösung für die Koeffizientenmatrix Ξ. Um die Forderung nach einer dünnbesetzten Matrix umzusetzen, werden anschließend alle Koeffizienten, deren Betrag unter einem festgelegten Grenzwert λ liegt, zu Null gesetzt.

$$\Xi^{\mathrm{d}}_{ij} := \begin{cases} 0, & |\Xi_{ij}| < \lambda \\ \Xi_{ij}, & \text{sonst} \end{cases}, \quad 1 \leq i \leq L, \quad 1 \leq j \leq n \tag{2.13}$$

Jeder von Null verschiedene Koeffizient Ξ^{d}_{ij} repräsentiert eine im Differentialgleichungssystem vorkommende Ansatzfunktion. Allerdings sind die Koeffizienten in Ξ^{d} noch ungenau, da sie unter Berücksichtigung aller Ansatzfunktionen der Bibliothek berechnet wurden, also auch derer, die nicht in der DGL vorkommen. Daher wird die Koeffizientenmatrix Ξ neu berechnet, unter Nutzung des Wissens darüber, welche Ansatzfunktionen nicht in der Differentialgleichung vorkommen.

Jeder Null-Koeffizient einer Spalte k der Koeffizientenmatrix Ξ^{d} repräsentiert eine Ansatzfunktion, die in der k-ten Zeile des DGL-Systems nicht vorkommt. Die neue Koeffizientenmatrix kann spaltenweise berechnet werden, indem man die Bibliothek um die für diese Zeile nicht relevanten Ansatzfunktionen verkleinert. Damit wird statt der Originalbibliothek $\Theta \in \mathbb{R}^{m\times L}$ die verkleinerte Bibliothek $\Theta_{\mathrm{v}} \in \mathbb{R}^{m\times h_k}$, $h_k \leq L$ verwendet. (Der Fall $h = L$ kann auftreten, wenn in einer Spalte jeder Koeffizient von Null verschieden ist, in einer anderen Spalte aber noch mindestens eine Null vorkommt.) Die neue Koeffizientenmatrix wird wie folgt berechnet (die Indizes der θ sind willkürlich

gewählt und repräsentieren diejenigen Ansatzfunktionen, deren Koeffizienten nicht Null sind):

$$\dot{X}_k = \begin{pmatrix} | \\ \dot{\boldsymbol{x}}_k(t) \\ | \end{pmatrix} \in \mathbb{R}^m \tag{2.14a}$$

$$\Theta_{\mathrm{v}}(X) := \begin{bmatrix} | & | & | & \\ \theta_2(X) & \theta_5(X) & \theta_7(X) & \dots \\ | & | & | & \end{bmatrix} \in \mathbb{R}^{m \times h_k}. \tag{2.14b}$$

$$\hat{\Xi}_k := \Theta_{\mathrm{v}}^{+} \dot{X}_k \in \mathbb{R}^{h_k} \tag{2.14c}$$

Um die neue Koeffizientenmatrix Ξ^{n} erzeugen zu können, müssen die Spalten $\hat{\Xi}_k \in \mathbb{R}^{h_k}$ erst auf die einheitliche Größe $\Xi_k^{\mathrm{n}} \in \mathbb{R}^L$ gebracht werden:

$$\Xi_{k,\,\mathrm{i}}^{\mathrm{n}} = \begin{cases} 0, & \Xi_{ik}^{\mathrm{d}} = 0 \\ \hat{\Xi}_{k,\,\mathrm{g}}, & \text{sonst} \end{cases} \quad \text{mit } 1 \leq g \leq h_k. \tag{2.15}$$

Die Spalten Ξ_k^{n} können nun durch Hintereinanderreihung zur Matrix Ξ^{n} zusammengesetzt werden.

Allerdings besteht die Möglichkeit, dass Ξ^{n} durch die erneute Berechnung Einträge besitzt, deren Beträge unterhalb des Grenzwertes λ liegen. Daher werden die Schritte (2.13), (2.14c) und (2.15) mit Ξ^{n} anstelle von Ξ solange wiederholt, bis in (2.13) $\Xi^{\mathrm{d}} = \Xi^{\mathrm{n}}$ gilt, also bis der Algorithmus die Koeffizientenmatrix nicht mehr verändert. Es werden somit iterativ immer mehr Ansatzfunktionen ausgeschlossen. Damit gilt am Ende

$$\dot{X} \approx \Theta(X)\Xi \tag{2.16a}$$

und somit

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \approx \Xi^T \left(\Theta(\boldsymbol{x}^T)\right)^T, \tag{2.16b}$$

wobei Ξ dünnbesetzt ist[1].

[1] *Xi* ist nur dann nicht dünnbesetzt, wenn die Bibliothek sehr wenige Funktionen enthält (was in der Regel jedoch nicht gegen die Güte des Identifikationsergebnisses spricht), oder wenn der Grenzwert λ zu klein gewählt wird und Funktionen mit sehr kleinen Koeffizienten nicht aus der DGL entfernt werden, was das Identifikationsergebnis verschlechtert.

Kapitel 3

Durchführung von Tests

3.1 Vorstellung der Implementationen

Neben dem in Abschnitt 2.4 vorgestellten vereinfachten Algorithmus werden zwei weitere Implementationen der SINDy-Methode untersucht. De Silva et al. [7] haben SINDy in *Python* (PySINDy) implementiert, während Rackauckas et al.[6] die SINDy-Implementation der DataDrivenDiffEq-Bibliothek in *Julia* nutzen. Die drei Algorithmen sind sich im Aufbau sehr ähnlich und folgen dem in Kapitel 2 beschriebenen Ablauf. Jedoch unterscheiden sich die Algorithmen in der Lösungsstrategie des überbestimmten Gleichungssystems (2.9). PySINDy verwendet das Cholesky-Verfahren [4], während die Julia-Implementation die QR-Faktorisierung [5] anwendet.

3.2 Vorstellung der Testsysteme

Für den Vergleich der Implementationen werden drei verschiedene Differentialgleichungssysteme betrachtet, deren Parameter es zu identifizieren gilt.

Das *Lotka-Volterra*-System (im Folgenden auch kurz: Volterra-System) ist definiert durch

$$\begin{aligned} \dot{x} &= \alpha x + \beta xy \\ \dot{y} &= \gamma y + \delta xy \end{aligned} \tag{3.1}$$

mit den Nominalparametern

$$\boldsymbol{p}_{\mathrm{V}} = \begin{pmatrix} \alpha & \beta & \gamma & \delta \end{pmatrix} = \begin{pmatrix} 1.3 & -0.9 & -1.8 & 0.8 \end{pmatrix} \tag{3.2}$$

und den Anfangswerten

$$\begin{pmatrix} x_0 & y_0 \end{pmatrix} = \begin{pmatrix} 0.442 & 4.62 \end{pmatrix}. \tag{3.3}$$

Das *Lorenz*-System ist definiert durch

$$\begin{aligned} \dot{x} &= \alpha(y - x) \\ \dot{y} &= x(\beta - z) - y \\ \dot{z} &= xy - \gamma z \end{aligned} \tag{3.4}$$

mit den Nominalparametern

$$\boldsymbol{p}_{\mathrm{L}} = \begin{pmatrix} \alpha & \beta & \gamma \end{pmatrix} = \begin{pmatrix} 10 & 28 & \frac{8}{3} \end{pmatrix} \tag{3.5}$$

und den Anfangswerten

$$\begin{pmatrix} x_0 & y_0 & z_0 \end{pmatrix} = \begin{pmatrix} -8 & 8 & 27 \end{pmatrix}. \tag{3.6}$$

Das *Rössler*-System ist definiert durch

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + \alpha y \\ \dot{z} &= \beta + (x - c)z \end{aligned} \tag{3.7}$$

mit den Nominalparametern

$$\boldsymbol{p}_{\mathrm{R}} = \begin{pmatrix} \alpha & \beta & \gamma \end{pmatrix} = \begin{pmatrix} 0.2 & 0.1 & 5.3 \end{pmatrix} \tag{3.8}$$

und den Anfangswerten

$$\begin{pmatrix} x_0 & y_0 & z_0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 \end{pmatrix}. \tag{3.9}$$

Ausschlaggebend für die Güte des jeweiligen Algorithmus ist an erster Stelle die Genauigkeit des Ergebnisses, an zweiter Stelle die Rechenzeit. Um die Genauigkeit des Algorithmus vergleichen zu können, wird ein Fehler ε_{r} definiert. Dieser ist an das quadratische Mittel des relativen Fehlers angelehnt. $P \in \mathbb{R}^{L \times n}$ bezeichnet die nominale und $Q \in \mathbb{R}^{L \times n}$ die identifizierte Koeffizientenmatrix eines Systems. Dann sei

$$R \in \mathbb{R}^{L \times n}, \quad R_{ij} = \begin{cases} |\frac{P_{ij} - Q_{ij}}{P_{ij}}|, & P_{ij} \neq 0 \\ |Q_{ij}|, & P_{ij} = 0 \wedge |Q_{ij}| < 1 \\ 1, & P_{ij} = 0 \wedge |Q_{ij}| \geq 1 \\ 0, & \text{sonst} \end{cases}, \quad 1 \leq i \leq L, \quad 1 \leq j \leq n \tag{3.10}$$

$$\varepsilon_{\mathrm{r}} = \frac{1}{\sqrt{k}} \|R\|_2, \tag{3.11}$$

wobei k die Anzahl der von Null verschiedenen Elemente von P ist. Für richtig identifizierte Ansatzfunktionen wird somit der relative Fehler zwischen Nominalparameter und identifiziertem Parameter berechnet, während bei falsch identifizierten Ansatzfunktionen der Fehler, bis hin zu einer Grenze, vom identifizierten Parameter abhängt. Dadurch

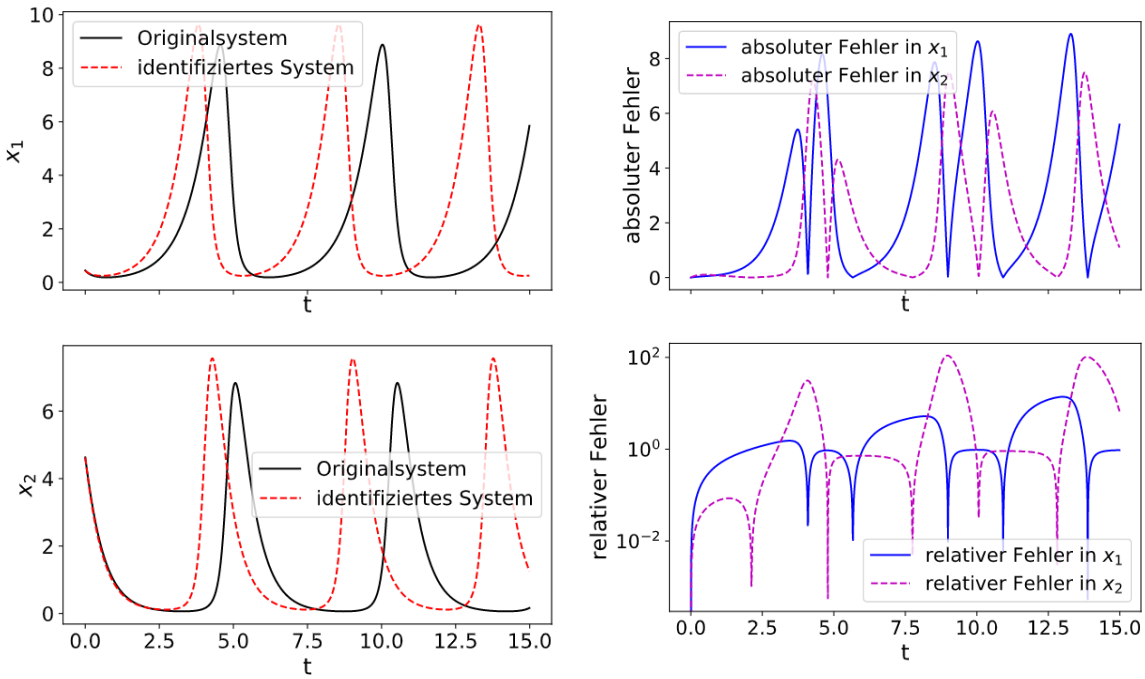
werden kleine Koeffizienten vor falschen Funktionen weniger stark bestraft, da diese einen geringeren Einfluss auf das Ergebnis haben.

Die Systeme werden durch einen Differentialgleichungslöser simuliert und die zeitlichen Verläufe der Zustandskomponenten numerisch ermittelt. Die Ableitungsverläufe werden sowohl exakt vorgegeben, als auch über die Zentraldifferenz angenähert. Die exakten Ableitungsverläufe dienen hier nur der Überprüfung, ob die Methode theoretisch funktioniert und sind im Praxisfall aufgrund von Messfehlern oft nicht gegeben. Das Hauptaugenmerk wird auf der Identifikation unter Nutzung der Zentraldifferenz liegen, da diese den Praxisfall besser abbildet.

3.3 Einfluss ausgewählter Parameter

3.3.1 Anmerkungen zur Auswertung

Im Folgenden wird der Einfluss jedes Parameters exemplarisch für eine Auswahl an Systemen gezeigt, da der zu beobachtende Trend für alle drei Systeme ähnlich ausfällt.

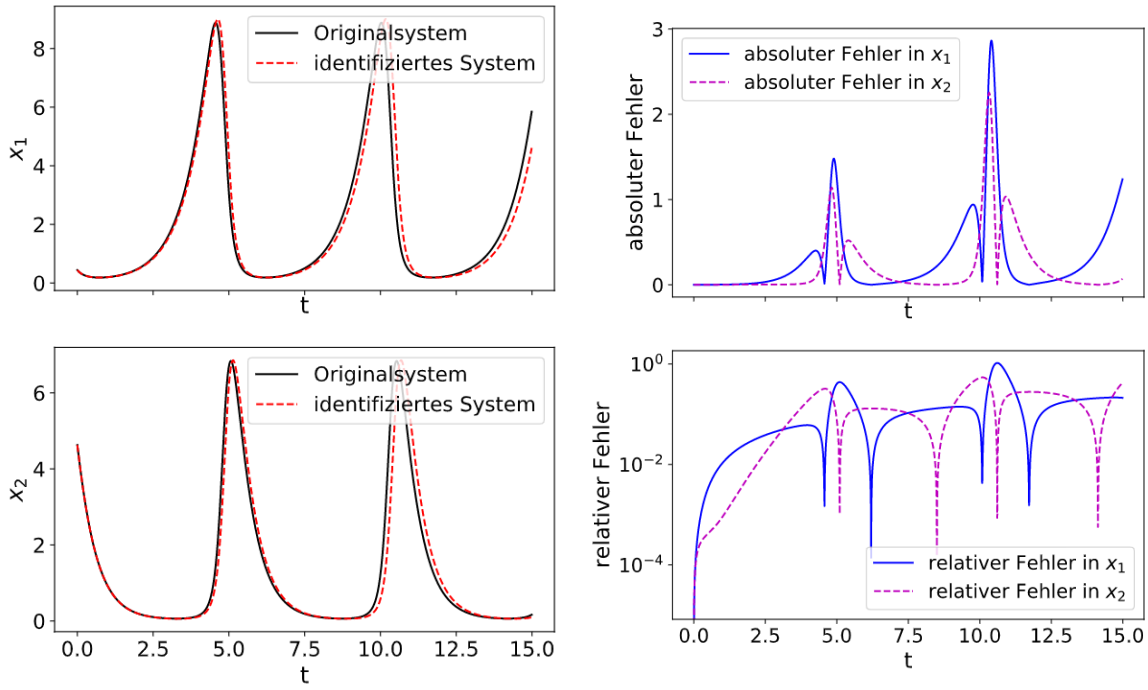


a) Simulationsverlauf

b) Fehlerverlauf

Abbildung 1 – Simulation des Lotka-Volterra-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_\mathrm{r} = 10\%$.

Um die nachfolgend gezeigten Fehler in den identifizierten Koeffizienten in Relation



a) Simulationsverlauf

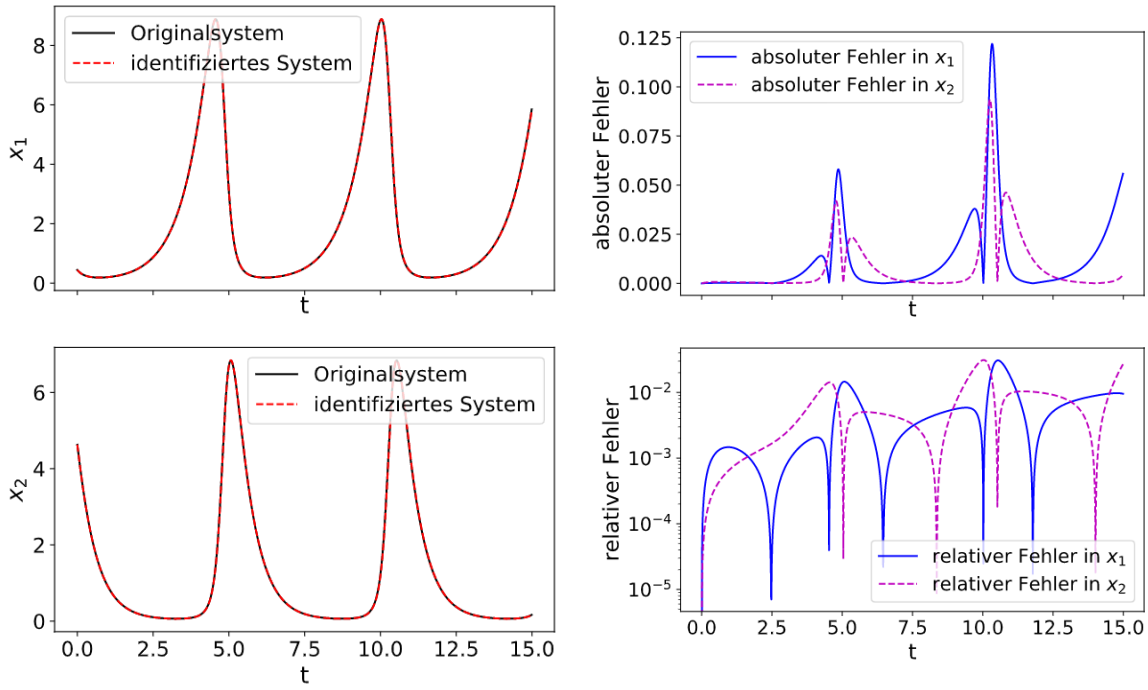
b) Fehlerverlauf

Abbildung 2 – Simulation des Lotka-Volterra-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_\mathrm{r} = 1\%$.

setzen zu können, ist es sinnvoll, einen Blick auf Simulationen der identifizierten Systeme zu werfen. Abb. 1, 2 und 3 zeigen die Simulation des Lotka-Volterra-Systems mit unterschiedlich gut identifizierten Koeffizienten. Je kleiner der relative Fehler ε_r der Koeffizienten ist, desto länger kann das identifizierte System als gute Näherung für das Originalsystem genutzt werden. Wie lange dies der Fall sein soll ist im Einzelfall von der gegebenen Anwendung abhängig und gibt damit auch eine Vorgabe für den maximal erlaubten Fehler der identifizierten Koeffizienten. Im Folgenden wird ein Identifikationsfehler ε_r als klein und damit die Identifikation als erfolgreich befunden, wenn $\varepsilon_\mathrm{r} < 1\%$.

Der relative Fehler der Nominalableitungen zeigt den geringstmöglichen Fehler, den die Methode theoretisch erzielen könnte, wenn die Messungen der Zustandsableitungen exakt wären. Ist dieser Fehler null, so ist unter den gegebenen Parametern eine Systemidentifikation prinzipiell möglich. Im Praxisfall ist der Erfolg der Identifikation jedoch weiterhin von einem geeigneten Zusammenspiel zwischen Algorithmus-Parametern und den Messfehlern in den Datenreihen abhängig. Ist der Fehler bereits im Nominalfall nicht klein genug, so ist eine Identifikation bei Verwendung der Zentraldifferenz nicht möglich. Das deutet darauf hin, dass die Algorithmus-Parameter ungünstig gewählt sind.

Es wird der Einfluss der nachfolgenden Parameter auf das Identifikationsergebnis



a) Simulationsverlauf

b) Fehlerverlauf

Abbildung 3 – Simulation des Lotka-Volterra-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_\mathrm{r} = 0.1\%$.

untersucht. Für die Untersuchung werden Standardwerte festgelegt, die konstant bleiben, während der zu untersuchende Parameter verändert wird. Diese werden in eckigen Klammer angegeben.

- die Simulationszeit t des DGL-Lösers, [3s],
- die Schrittweite dt des DGL-Lösers, [0.01s],
- die Gestaltung der Bibliothek von Ansatzfunktionen, [Polynome maximal zweiter Ordnung],
- die Stärke des Rauschens in den Daten, [kein Rauschen].

Die Standardparameter sind so gewählt, dass der resultierende Fehler für alle Systeme bereits klein ist. Die Veränderung eines Parameters kann dann den Bereich zeigen, in dem dieser noch sinnvoll verändert werden kann.

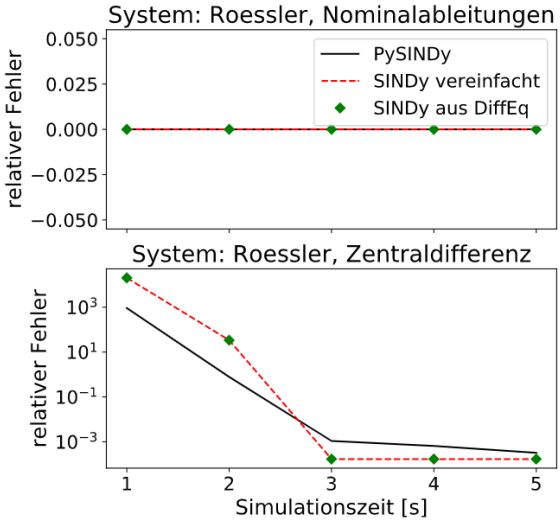


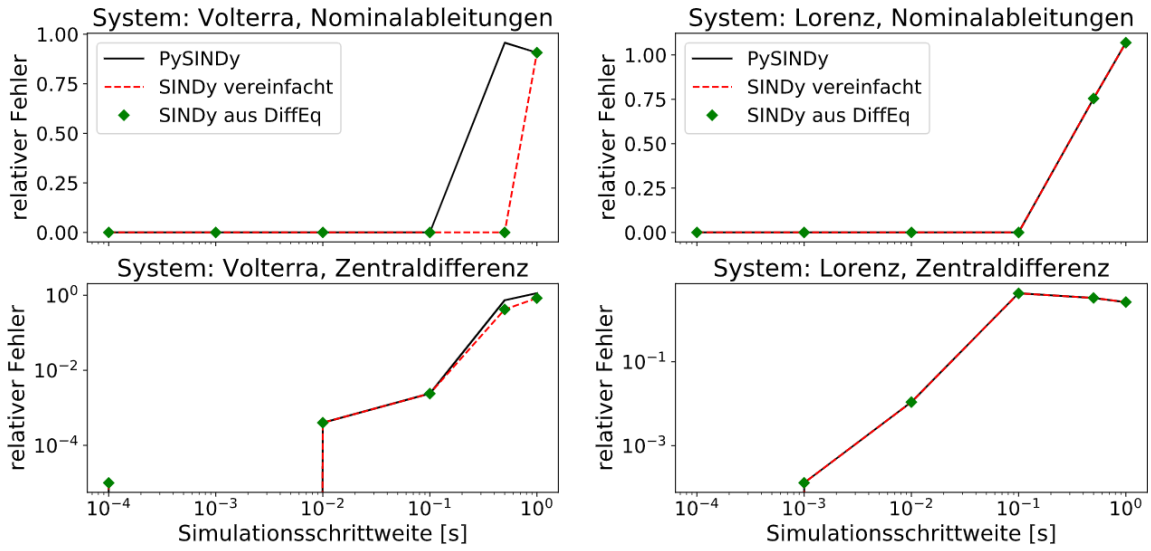
Abbildung 4 – relativer Fehler der identifizierten Parameter in Abhängigkeit der Simulationszeit.

3.3.2 Einfluss der Simulationszeit

Wie aus Abb. 4 hervorgeht, sinkt der relative Fehler der identifizierten Parameter bei Verwendung der Zentraldifferenz mit steigender Simulationszeit. Dabei ist auffällig, dass für zu geringe Simulationszeiten keine Systemidentifikation möglich ist, während sich der Fehler ab einer gewissen Mindestsimulationszeit kaum noch verändert. Diese Mindestzeit ist notwendig, damit charakteristische Zustandsverläufe sichtbar werden, ähnlich wie die Sinusfunktion mit der Winkelhalbierenden verwechselt werden kann, wenn das Betrachtungsfenter ungünstig gewählt wird. Allerdings sinkt der Fehler für große Simulationszeiten nicht mehr signifikant. Damit existiert für jedes System eine optimale Simulationsdauer.

3.3.3 Einfluss der Simulationsschrittweite

Um Anforderungen an die Messung der Zustandskomponenten zu minimieren, ist es sinnvoll zu fordern, dass die Zeit zwischen zwei Messungen (hier durch die Simulationsschrittweite repräsentiert) so groß wie möglich gewählt wird. In Abb. 5 zeigt sich, dass der Fehler mit zunehmender Schrittweite steigt. Jedes System besitzt eine spezifische Schrittweite, ab welcher keine Identifikation mehr möglich ist. Anders als bei der Simulationszeit sinkt der Fehler hier für kleine Schrittweiten immer weiter, bis hin zum Extremfall, dass die Schrittweite infinitesimal klein ist und der Fehler null wird (siehe Nominalfall). Damit muss man bei der Auswahl der Schrittweite zwischen Genauigkeit und Messaufwand abwägen.



a) Lotka-Volterra-System b) Lorenz-System

Abbildung 5 – relativer Fehler der identifizierten Parameter in Abhängigkeit der Simulationsschrittweite.

3.3.4 Einfluss der Gestaltung der Bibliothek

Wie bereits angedeutet ist die Auslegung der Bibliothek entscheidend für eine erfolgreiche Identifikation. Je weniger Informationen dem Algorithmus im Vorhinein zur Verfügung gestellt werden müssen, desto mächtiger ist er. Daher ist es wünschenswert, die Bibliothek so groß wie möglich ansetzen zu können, um auch Funktionsklassen zu beinhalten, deren Vorkommen nicht gesichert ist. Für die drei zu untersuchenden Systeme wurde eine polynomiale Bibliothek angesetzt, die schrittweise durch Monome höherer Ordnung ergänzt wurde. Abb. 6 zeigt, dass die Bibliothek mit deutlich mehr Funktionen besetzt werden kann als nötig und eine Identifikation trotzdem noch möglich ist. Allerdings gibt es auch hier einen Punkt, ab dem die Identifikation scheitert. Mit zunehmender Anzahl an Funktionen wird es wahrscheinlicher, dass eine Linearkombination dieser Funktionen existiert, die eine in der DGL vorkommende Funktion hinreichend genau beschreibt und ein Teil des Ergebnisses wird. Die so entstandene DGL kann das System in der Regel ausreichend gut numerisch modellieren, ist aber auf Grund der falsch identifizierten Ansatzfunktionen nur als Black-Box-Modell zu gebrauchen und für die Identifikation von analytischen Zusammenhängen unbrauchbar.

3.3.5 Einfluss von verrauschten Daten

Für die folgende Untersuchung (Abb. 7) wurden die Verläufe der Zustandskomponenten mit additivem weißen Rauschen überlagert. Die Stärke des Rauschsignals wurde über

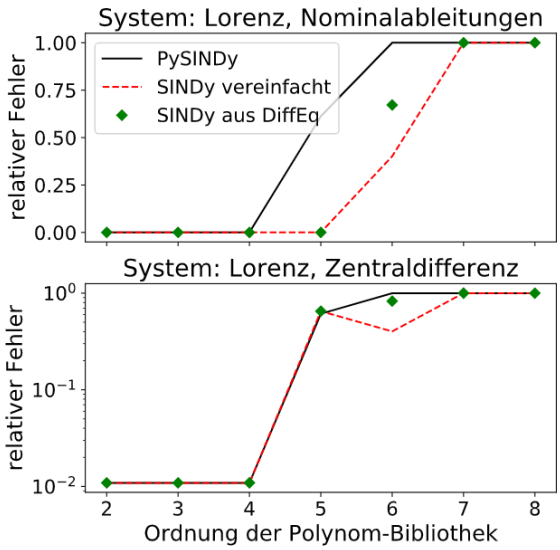
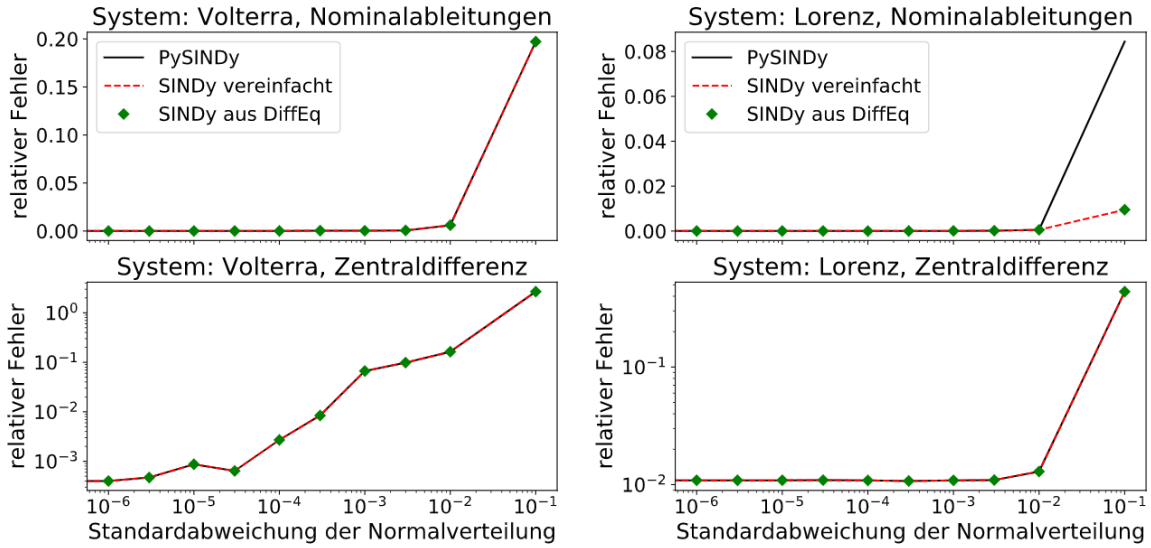


Abbildung 6 – relativer Fehler der Identifizierten Parameter in Abhängigkeit der Ordnung der Polynom-Bibliothek.

die Standardabweichung der verwendeten Normalverteilung eingestellt. Je stärker das Rauschen, desto ungenauer werden die identifizierten Koeffizienten. Wie stark verrauscht die Daten sein können, sodass dennoch eine Identifikation möglich ist, hängt vom System ab.



a) Lotka-Volterra-System b) Lorenz-System

Abbildung 7 – relativer Fehler der identifizierten Parameter in Abhängigkeit der Rauschstärke.

3.3.6 Rechenzeit

Abb. 8 zeigt exemplarisch das qualitative Verhältnis der Rechenzeiten der Implementationen. Beide *Python*-Implementationen sind deutlich schneller als die *Julia*-Version. Dennoch ist die Rechenzeit in allen Fällen recht klein. Der vereinfachte Algorithmus schneidet am besten ab, was damit zu erklären ist, dass viele periphere Funktionen, wie die Validierung der Eingabedaten, nicht implementiert sind.

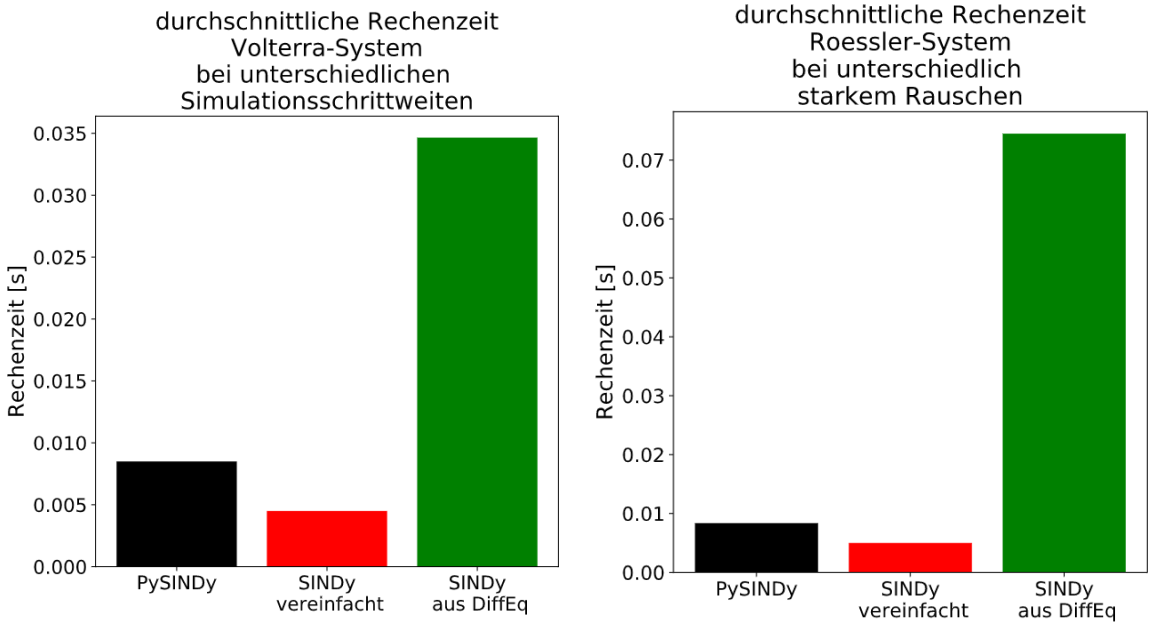


Abbildung 8 – durchschnittliche Rechenzeit der unterschiedlichen Algorithmen.

3.3.7 Zusammenfassung der Ergebnisse

Anders als zu Beginn der Arbeit vermutet, lassen sich keine Unterschiede in der Genauigkeit der SINDy-Implementationen feststellen. Allerdings ist die richtige Wahl der Parameter von großer Bedeutung für die Qualität der Identifikation. Die identifizierten Parameter werden genauer, wenn mehr Messdaten zur Verfügung stehen, also die Anzahl der Messungen steigt und der zeitliche Abstand zwischen ihnen verringert wird. Der Algorithmus kann mit verrauschten Messdaten arbeiten, genauere Messungen liefern bessere Ergebnisse. Die Bibliothek aus Ansatzfunktionen muss groß genug sein, um alle in den Systemgleichungen vorkommenden Ansatzfunktionen zu beinhalten und gleichzeitig klein genug, damit die Richtigen ausgewählt werden. Leider sind die optimalen Parameter von System zu System unterschiedlich, sodass hier keine allgemeingültigen Aussagen zur Größe von Parametern getroffen werden können.

3.4 Verbesserung der Ergebnisse

3.4.1 Arbeit mit mehreren Trajektorien

Wenn die Identifikation in den bisher betrachteten Beispielen scheiterte, dann wurde statt der korrekten rechten Seite einer Differentialgleichung eine andere Linearkombination von Ansatzfunktionen identifiziert, die einen kleineren Fehler (2.12) hinterlässt und das System somit "besser" modelliert. Anschaulich bedeutet das, dass aus den genutzten Daten die Dynamik des Systems nicht eindeutig herauszulesen war. Dabei spielen die Anfangswerte der verwendeten Trajektorie eine entscheidende Rolle. Besitzt das System beispielsweise eine Ruhelage und wird diese als Anfangswert dem Differentialgleichungslöser vorgegeben, so ist aus der resultierenden Trajektorie keine Systemidentifikation möglich. Daher kann es nützlich sein, mehrere Trajektorien mit verschiedenen Anfangswerten zu verwenden. Umgesetzt wird dies, indem die Matrizen der Zustands- und Ableitungsverläufe untereinander angeordnet werden

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_t \end{bmatrix} \in \mathbb{R}^{m \cdot t \times n}, \quad \dot{X} = \begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \vdots \\ \dot{X}_t \end{bmatrix} \in \mathbb{R}^{m \cdot t \times n}. \tag{3.12}$$

In Abb. 9 wurde jeder Parameter so ungünstig gewählt, dass allein dieser die Identifikation schwierig machen würde (Simulationszeit von 1s, Schrittweite von 0.1s, Bibliothek aus Polynomen bis zur Ordnung 5, Weißes Rauschen mit Standardabweichung von 10^{-4}). Durch die Verwendung von ausreichend vielen Messreihen kann die Identifikation dennoch gelingen.

3.4.2 Identifikation von teilweise bekannten Systemen

Rackauckas et al. [6] zeigen, wie man Wissen über bereits bekannte Teilsysteme in die Identifikation integrieren kann. Angenommen man möchte das Lotka-Volterra-System (3.1) identifizieren und kennt bereits die linearen Terme und ihre Koeffizienten. Dann lässt sich das System darstellen als

$$\begin{aligned} \dot{x} &= \alpha x + U_1(x, y) \\ \dot{y} &= \gamma y + U_2(x, y), \end{aligned} \tag{3.13}$$

mit den unbekannten nichtlinearen Termen $U_1(x, y)$ und $U_2(x, y)$. Da die Zustandsableitungen als bekannt angenommen werden (durch Messung oder Zentraldifferenz aus den Zuständen ermittelt), können die unbekannten Terme auf der rechten Seite isoliert werden:

$$\begin{aligned} \dot{x} - \alpha x &= U_1(x, y) \\ \dot{y} - \gamma y &= U_2(x, y). \end{aligned} \tag{3.14}$$

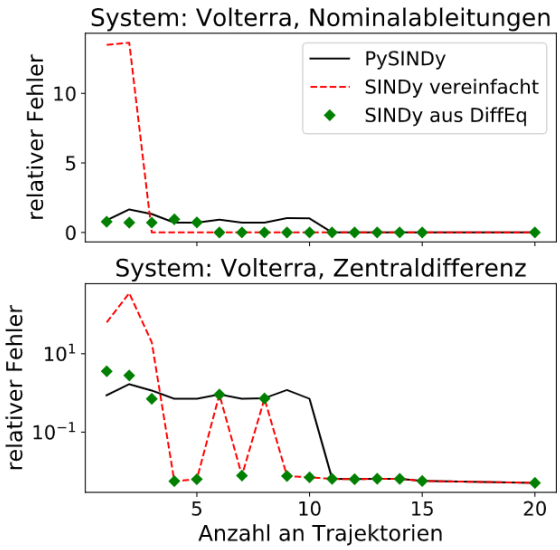


Abbildung 9 – relativer Fehler der identifizierten Parameter in Abhängigkeit der Anzahl der gleichzeitig verwendeten Trajektorien.

Somit kann die angepasste linke Seite der Differentialgleichung folgendermaßen konstruiert werden:

$$\dot{X} = \begin{bmatrix} \dot{x}(t_1) - \alpha x(t_1) & \dot{y}(t_1) - \gamma y(t_1) \\ \dot{x}(t_2) - \alpha x(t_2) & \dot{y}(t_2) - \gamma y(t_2) \\ \vdots & \vdots \\ \dot{x}(t_m) - \alpha x(t_m) & \dot{y}(t_m) - \gamma y(t_m) \end{bmatrix} \in \mathbb{R}^{m \times 2}. \tag{3.15}$$

Von hier aus kann der Algorithmus wie bekannt weiterlaufen.

Eine Voraussetzung für diese Methode ist, dass sich bekannte und unbekannte Teilsysteme voneinander isolieren lassen. Dies ist der Fall, wenn eine bekannte Funktion $K(\boldsymbol{x}) \in \mathbb{R}^n$ und eine unbekannte Funktion $U(\boldsymbol{x}) \in \mathbb{R}^n$ existieren, sodass die Umformung

$$K(\boldsymbol{x}) \circ \dot{\boldsymbol{x}} = U(\boldsymbol{x}) \tag{3.16}$$

möglich ist. Ist dies nicht der Fall, z.B. für

$$\dot{\boldsymbol{x}} = K_1(\boldsymbol{x}) \cdot U_1(\boldsymbol{x}) + K_2(\boldsymbol{x}) \cdot U_2(\boldsymbol{x}), \tag{3.17}$$

dann kann das Wissen über das bekannte Teilsystem nur in die Gestaltung der Bibliothek einfließen, nicht jedoch die Identifikation wie beschrieben von Anfang an vereinfachen. Dieses Problem wird in Abschnitt 4.2 noch einmal aufgegriffen.

Kapitel 4

Anwendung auf komplexes Beispiel - Inverses Pendel

4.1 Einleitung

Ein beliebtes regelungstechnisches Problem ist die Regelung eines inversen Pendels an einem Wagen (Abb. 10). Die Voraussetzung dafür ist das Vorhandensein eines guten Systemmodells. Während solche Modelle berechnet werden können [3, Softwarebeispiel 3], soll hier untersucht werden, ob ähnliche Ergebnisse mit SINDy reproduziert werden können, oder ob SINDy zur Parameterschätzung genutzt werden kann. Das Wagen-

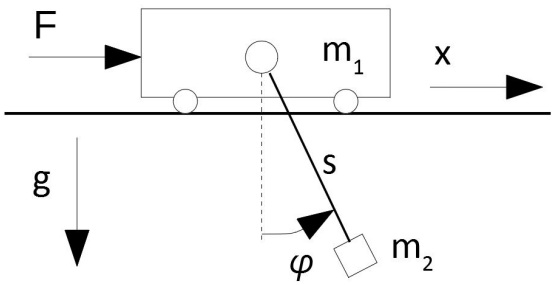


Abbildung 10 – Schematische Darstellung des Wagen-Pendel-Systems, mit den Massen m_1 und m_2, der Pendellänge s, der Erdbeschleunigung g, dem Krafteingang F, der Position des Wagens x und der Auslenkung des Pendels aus der unteren Ruhelage φ.

Pendel-System wird durch folgende DGL beschrieben:

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} \dot{\varphi} \\ \dot{x} \\ \ddot{\varphi} \\ \ddot{x} \end{pmatrix} = f(\boldsymbol{x}) + g(\boldsymbol{x})u \tag{4.1}$$

$$= \begin{pmatrix} \dot{\varphi} \\ \dot{x} \\ -\frac{m_2 s\dot{\varphi}^2 \sin\varphi\cos\varphi + (m_1+m_2)g\sin\varphi}{s(m_1+m_2\sin^2\varphi)} \\ \frac{g m_2 \sin\varphi\cos\varphi + m_2 s\dot{\varphi}^2\sin\varphi}{(m_1+m_2\sin^2\varphi)} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -\frac{\cos\varphi}{s(m_1+m_2\sin^2\varphi)} \\ \frac{1}{m_1+m_2\sin^2\varphi} \end{pmatrix} F. \tag{4.2}$$

Der Einfachheit halber wird nur das autonome System betrachtet, daher gilt $F \equiv 0$. Ausgehend von den zeitlichen Verläufen der Zustandskomponenten und deren Ableitungen sollen die Struktur und die Parameter $\boldsymbol{p} = \begin{pmatrix} m_1 & m_2 & s & g \end{pmatrix}$ des Systems bestimmt werden.

4.2 Aufbau der Bibliothek

Das Wagen-Pendel-System wird im Gegensatz zu den bisher betrachteten Systemen nicht ausschließlich durch Polynome beschrieben. Das macht die Konstruktion der Bibliothek deutlich schwieriger. Der Algorithmus berechnet eine Linearkombination von Ansatzfunktionen, die das System modellieren sollen. Schreibt man die letzten beiden Zeilen von (4.2) als Summe

$$\begin{pmatrix} \dot{\varphi} \\ \dot{x} \end{pmatrix} = \begin{pmatrix} -\frac{m_2 s\dot{\varphi}^2 \sin\varphi\cos\varphi}{sm_1+sm_2\sin^2\varphi} - \frac{(m_1+m_2)g\sin\varphi}{sm_1+sm_2\sin^2\varphi} \\ \frac{g m_2 \sin\varphi\cos\varphi}{m_1+m_2\sin^2\varphi} + \frac{m_2 s\dot{\varphi}^2\sin\varphi}{m_1+m_2\sin^2\varphi}, \end{pmatrix} \tag{4.3}$$

so fällt auf, dass in jeder Zeile nur zwei verschiedene Ansatzfunktionen vorkommen. Diese müssen dem Algorithmus in der Bibliothek vorgegeben werden. Es lohnt sich darauf hinzuweisen, dass es nicht ausreicht, eine Bibliothek aus den vorkommenden Elementarfunktionen ($\sin, \cos$, Zustandskomponenten, Identität) anzulegen in der Hoffnung, dass SINDy diese geeignet verknüpft. SINDy kann die vorgegeben Funktionen nur additiv durch die angesprochene Linearkombination verknüpfen. Damit eine Identifikation möglich ist, muss sich jede Zeile der DGL wie folgt aus Funktionen u_i und v_i darstellen lassen:

$$x_i = \sum_i u_i(\boldsymbol{p}) v_i(\boldsymbol{x}), \tag{4.4}$$

wobei u_i nicht von $\boldsymbol{x}$ und v_i nicht von $\boldsymbol{p}$ abhängig sein darf. Dies ist auf Grund der Struktur der Nenner hier nicht möglich. Daher müssen die Nenner in diesem Beispiel als bekannt vorausgesetzt werden. Die Zähler haben eine für die Identifikation günstige Struktur, da sie Bedingung (4.4) erfüllen. Dabei werden maximal vier Elementarfunktionen miteinander multipliziert. Um die Bibliothek systematisch aufzubauen, müssten für

die Zähler alle Kombinationen mit Wiederholung aus maximal vier Elementarfunktionen berücksichtigt werden. Damit ergeben sich $C^r(13, 4) = 1820$[1] mögliche Zähler, wodurch die Bibliothek viel zu groß ist. Um das zu verhindern müssen einschränkende Annahmen getroffen werden:

- Nur die Auslenkung des Pendels aus der unteren Ruhelage φ darf als Argument der Winkelfunktionen verwendet werden.
- Nur $\dot{\varphi}$ kann außerhalb der Winkelfunktionen vorkommen.

Damit verringert sich die Anzahl an möglichen Zählern auf $C^r(4, 4) = 35$. Aus diesen Zählern und den bekannten Nennern kann der Großteil der Ansatzfunktionen für die Bibliothek erzeugt werden. Es fehlen nur noch Monome ersten Grades, um die ersten beiden Zeilen der DGL abbilden zu können. Damit ergibt sich eine Bibliothek aus 39 Ansatzfunktionen. Jedoch liefert diese noch nicht das gewünschte Ergebnis. Problematisch ist, dass es in der Bibliothek linear abhängige Spalten gibt, da gilt:

$$\cos\varphi = \cos^3\varphi + \sin^2\varphi\cos\varphi. \tag{4.5}$$

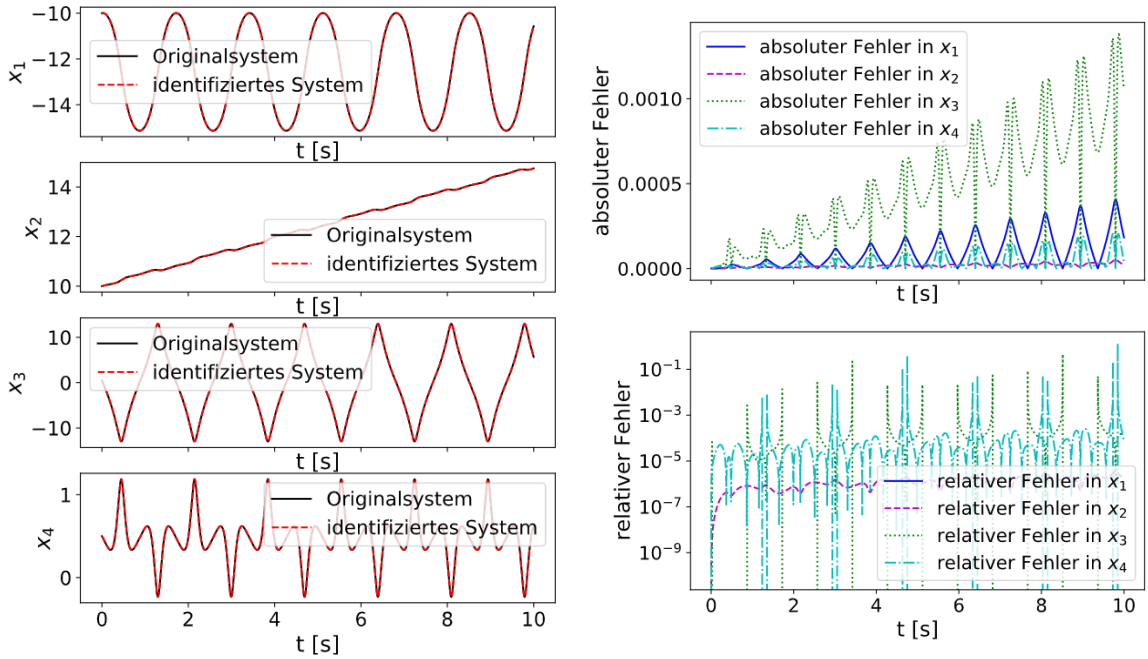
Dadurch kann, statt der vollständig vereinfachten Gleichung, eine zwar mathematisch richtige, jedoch wenig verwertbare Gleichung geliefert werden, da diese erst vereinfacht und die Koeffizienten neu berechnet werden müssten. Stattdessen kann man die Bibliothek noch einmal verkleinern, indem man die linear abhängigen Spalten entfernt. Diese kann man finden, indem man den Spaltenrang der ersten i Spalten mit dem Spaltenrang der ersten $i + 1$ Spalten der Bibliothek vergleicht. Sind die Ränge gleich, ist die $i + 1$-te Spalte von den ersten i Spalten linear abhängig und kann entfernt werden. Führt man diesen Prozess für die gesamte Bibliothek aus, so bleiben am Ende noch 28 Ansatzfunktionen übrig[2]. Die Bibliotheksmatrix besitzt vollen Rang.

4.3 Bewertung der Ergebnisse

Unter Nutzung des Wissens über die geeignete Wahl der Identifikationsparameter und Verwendung mehrerer Trajektorien ist die Identifikation, wie in Abb. 11 ersichtlich, erfolgreich. Auf Grund der komplexen Struktur des Wagen-Pendel-Systems ist der SINDy-Algorithmus ohne beträchtliches Vorwissen über das System und einige Vereinfachungen nicht geeignet. Lassen sich die Differentialgleichungen nicht als Linearkombination relativ einfacher Funktionen darstellen, stößt die Methode schnell an ihre Grenzen. Allerdings zeigt das Beispiel, dass SINDy sehr gute Ergebnisse liefern kann, wenn die Struktur des Systems bereits bekannt ist und nur Parameter geschätzt werden sollen.

[1] Es gibt 13 mögliche Faktoren: sin und cos von jeder Zustandskomponente, vier Zustandskomponenten, Identität.

[2] Die Auswahl der 28 Spalten ist nicht eindeutig, aber mathematisch äquivalent, da die resultierenden Gleichungen gegebenenfalls vereinfacht werden können.



a) Simulationsverlauf

b) Fehlerverlauf

Abbildung 11 – Simulation des Wagen-Pendel-Systems, relativer Fehler der identifizierten Koeffizienten $\varepsilon_\mathrm{r} = 8 \cdot 10^{-5}$.

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

In dieser Arbeit wurde der SINDy-Algorithmus als Methode zur Gewinnung von Systemdifferentialgleichungen aus Messdaten untersucht. Es wurde der Einfluss verschiedener Parameter auf das Identifikationsergebnis vorgestellt und Richtlinien zu deren Einstellung abgeleitet. SINDy ist besonders für die Identifikation parameterlinearer Systeme geeignet, darüber hinaus muss man viel Vorwissen beisteuern, um brauchbare Ergebnisse zu erzielen. Allgemein ist das benötigte Vorwissen über das zu untersuchende System ein entscheidender Nachteil der Methode. Ist dieses jedoch vorhanden, so kann SINDy sehr genaue Ergebnisse liefern, was z.B. die Schätzung von Systemparametern angeht.

5.2 Ausblick

Potential für weitere Untersuchungen bietet der SINDy-Algorithmus bei der Identifikation teilweise bekannter Systeme. Oft sind die physikalischen Zusammenhänge eines Systems theoretisch bekannt. SINDy kann dann dazu genutzt werden, nur parasitäre Einflüsse wie Reibungen zu modellieren. Eine weitere vorstellbare Anwendung könnte die Echtzeitschätzung von Parametern sein. Die Messdaten der Zustandskomponenten werden zur Laufzeit, z.B. durch einen Beobachter, gesammelt und mittels SINDy werden die Systemparameter aktualisiert.

Literatur

[1] Steven L. Brunton, Joshua L. Proctor und J. Nathan Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 113.15 (März 2016), S. 3932–3937. DOI: 10.1073/pnas.1517384113.

[2] Pierre Courrieu. “Fast Computation of Moore-Penrose Inverse Matrices”. In: *Neural Information Processing - Letters and Reviews 8, 2 (2005) 25-29* (30. Apr. 2008). arXiv: 0804.4809 [cs.NE].

[3] Carsten Knoll. *Regelungstheoretische Analyse- und Entwurfsansätze für unteraktuierte mechanische Systeme*. Dresden: Carsten Knoll, 2016. ISBN: 9783741858994.

[4] Aravindh Krishnamoorthy und Deepak Menon. “Matrix Inversion Using Cholesky Decomposition”. In: (17. Nov. 2011). arXiv: 1111.4144 [cs.MS].

[5] Per-Gunnar Martinsson u. a. “Householder QR Factorization with Randomization for Column Pivoting (HQRRP). FLAME Working Note 78”. In: (8. Dez. 2015). arXiv: 1512.02671 [math.NA].

[6] Christopher Rackauckas u. a. “Universal Differential Equations for Scientific Machine Learning”. In: (13. Jan. 2020). arXiv: 2001.04385 [cs.LG].

[7] Brian de Silva u. a. “PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data”. In: *Journal of Open Source Software* 5.49 (2020), S. 2104. DOI: 10.21105/joss.02104. URL: https://doi.org/10.21105/joss.02104.