Technische Universität Dresden

Fakultät Elektrotechnik und Informationstechnik

Institut für Regelungs- und Steuerungstheorie

Diplomarbeit

Trajektorienplanung und Regelung für unteraktuierte mechanische Systeme

vorgelegt von: Yifan Xue

geboren am: 25.03.1992 in Xi'an, China

zum Erlangen des akademischen Grades

Diplomingenieur

(Dipl.-Ing.)

Betreuer:	Dr.-Ing. C. Knoll
Verantwortlicher Hochschullehrer:	Prof. Dr.-Ing. habil. Dipl.-Math. K. Röbenack
Tag der Einreichung:	24.10.2017

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

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tage dem Prüfungsausschuss der Fakultät Elektrotechnik und Informationstechnik eingereichte Diplomarbeit zum Thema

Trajektorienplanung und Regelung für unteraktuierte mechanische Systeme

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, 24. Oktober 2017 Yifan Xue

Kurzfassung

Unter unteraktuierte Systeme versteht man mechanische Systeme mit weniger Aktoren als mechanische Freiheitsgrade. Einige davon erfüllen die Brockett-Bedingung zur Überprüfung der asymptotischen Stabilität einer gewünschten Ruhelage mittels einer stetig differenzierbaren Rückführung nicht. In der vorliegenden Arbeit werden einige Modelle der unteraktuierten Systeme untersucht. Genauer gesagt wird das mit dem am Institut RST entworfenen Python-Paket *PyTrajectory* zur Trajektorienplanung auf diese Modelle angewendet. Das Paket wird dazu um die Möglichkeit erweitert, neben dem Eingangsverlauf auch eine geeignete Überführungszeit zu bestimmen. Gemäß des Ergebnises von *PyTrajectory* wird ein Steuergesetz für Brocketts nicht-holonomes Doppelintegrator-System mit Anfangswerten der Zustandsvariablen innerhalb einer kleinen Umgebung der Ruhelage im Ursprung entworfen. Einige Regelgesetze für die Überführungstrajektorien in einem größeren Bereich werden ebenfalls aufgestellt.

Abstract

Mechanical system with less actuators than degree of freedom is called unactuated system. Some of these violate the Brockett's stabilization condition thus cannot be stabilized to equilibrium point by a continuously differentiable control law. In this thesis, the author develops the function of the python package *PyTrajectory* which was released by the institute RST so that an execution time of trajectories can be calculated instead of given by the users. According to the simulation results from *PyTrajectory* it is possible to design a control law for the brockett's nonholonomic integrator that leads the states of the system moving from any start point in a limited neighborhood of the origin to the equilibrium point. The author also introduces some other control laws to solve this problem in a wider region of start points.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Symbol	Bedeutung	Bezug
$a_0, \ldots, b_2$	Koeffiziente der Polynom-förmigen Systemeingängen	Gl. 4.4
$\boldsymbol{c}$	Vektor der freien Koeffizienten	Gl. 3.2
$\mathbf{C}_a$	Zentrifugal- bzw. Corioliskraft des akutierten Teils	Gl. 4.4
$\mathbf{C}_u$	Zentrifugal- bzw. Corioliskraft des akutierten Teils	Gl. 4.4
$\boldsymbol{f}$	Systemzustandsfunktion	Gl. 2.1
$\mathbf{G}_a$	Konsevative Kraft des akutierten Teils	Gl. 4.4
$\mathbf{G}_u$	Konsevative Kraft des akutierten Teils	Gl. 4.4
$\boldsymbol{h}_*$	Lokaler Minimizer der Schrittweite für Iterationsverfahren	Gl. 3.10
$\mathbf{J}$	Jacobimatrix	Gl. 3.6
k	Zusätzliche freie Parameter für die Bestimmung der optimalen Überführungszeit	Gl. 3.14
l_p	Abstand zwischen dem Stützpunkt und dem Pendelschwerpunkt in dem inversen Pendel System	Gl. 3.16
L_{Tra}	gesamte Länge der Trajektorien	Gl. 4.17
m	Anzahl der Systemeingänge	Gl. 3.2
$\mathbf{M}_a$	Massenmatrix des akutierten Teils	Gl. 4.4
m_p	Pendelmasse	Gl. 3.16
$\mathbf{M}_u$	Massenmatrix des unterakuierten Teils	Gl. 4.4
m_w	Wagenmasse	Gl. 3.16
n	Anzahl der Systemzustände	Gl. 3.2
N_x	Anzahl der Spline-Abschnitte für den Zustand	Gl. 3.2
N_u	Anzahl der Spline-Abschnitte für den Eingang	Gl. 3.2
$Pe(k)$	Straffunktion von k	Gl. 3.20
$\boldsymbol{q}$	Minimalkoordinate des mechanischen Systems	Gl. 4.1
$\boldsymbol{S}_{x,i}$	Ansatzfunktion für den i-ten Zustand in Kollokationsverfahren	Gl. 3.2
$\boldsymbol{S}_{u,i}$	Ansatzfunktion für den i-ten Eingang in Kollokationsverfahren	Gl. 3.2
T^*	Kinetische Koenergie des mechanischen Systems	Gl. 4.1
$\boldsymbol{u}$	Systemeingangsvektor im Allgemein	Gl. 2.1

$\boldsymbol{u}_\xi$	Regelgesetz über Zeit für den Zustandsvektor $\boldsymbol{x} = \xi$ in der Umgebung von der Ruhelage	Theorem 2.1
V	Potentielle Energie des mechanischen Systems	Gl. 4.1
$\boldsymbol{x}$	Systemzustandsvektor	Gl. 2.1
$\dot{\boldsymbol{x}}$	Ableitung vom Zustandsvektor über die Zeit	Gl. 2.1
$\boldsymbol{x}_0$	Initialwert des Zustandsvektors	Gl. 2.1
$\boldsymbol{x}_e$	Ruhelage des Zustandsvektors	Abschnitt 2.2
$\boldsymbol{x}_T$	Endwert des Zustandsvektors	Gl. 3.1
$\boldsymbol{x}_*$	stationärer Punkt von Quadratmittelproblem	Gl. 3.10
$\dot{\boldsymbol{x}}_{ori}$	Originale Systemzustandsfunktion	Gl. 3.14
$\dot{\boldsymbol{x}}_{new}$	Systemzustandsfunktion nach der Koordinatentransformation von Zeit	Gl. 3.14
y	Systemausgang des inversen-Pendel-Systems	Beispiel 3.3
μ	Dämpfungsparameter für Levenberg-Marquardt-Algorithmus	Abschnitt 3.1.2
$\psi(\kappa, k)$	Sättigungsfunktion zum Beschränken von k mit κ dem begrenzten k in der neuen Koordinate	Gl. 3.18

Kapitel 1

Einführung

Im Bereich von der Engineering-Anwendung werden viele mechanischen Systeme in den letzten Jahren umfassend entwickelt. Darin existiert eine spezielle Klasse von Systemen, deren Anzahl der verallgemeinerten Koordinaten größer als die Zahl der unabhängigen Steuereingänge ist. Ein solches System wird als "unteraktuiertes mechanisches System" genannt. Ein interessantes Regelproblem liegt in der Bestimmung eines stetigen differenzierbaren Regelgesetz, mit dem eine Ruhelage dieses Systems asymptotisch stabil ist.

Zur Erforschung dieses Problems stellte der amerikanische Regeltheoretiker Roger W. Brockett das berühmte Theorem [2] auf, das eine notwendige Bedingung für die Existenz eines oben genannten Regelgesetz durch die Beurteilung der Surjektivität des Wertbereichs von Systemzustandsfunktionen liefert. Manche Systeme, beispielsweise das inverse-Pendel-System, erfüllen die Brockett-Bedingung während andere wie der nicht-holonome Doppelintegrator von Brockett die Bedingung nicht. Es ist interessant zu untersuchen, wie sich die Trajektorien dieser zwei Systeme aus der Umgebung einer Ruhelage in diese aussehen. Die Modellierung und Simulation werden in *Python* mittels eines Python-Pakets durchgeführt.

Das Python Softwarepaket *PyTrajectory* ist vom Institut von RST entwickelt, um die Trajektorien mit gegebenen Randbedingungen (Anfangs- und Endwerte der Systemzustände sowie (optional) Eingänge) für nichtlineare Systeme zu entwerfen. Mit *PyTrajectory* wird eine Trajektorie für ein voraus festgelegtes Zeitintervall geplant, die aber wegen der festen Endzeit im Allgemeinen nicht optimal ist. Deswegen ist eine Erweiterung dieses Pakets notwendig, mit dem die Überführungszeit als Teil des Optimierungsergebnisses ausgerechnet werden kann.

Falls die Trajektorien aus *PyTrajectory* von Anfangspunkten in einer Umgebung der Ruhelage in diese Ruhelage (Endzustand) eine "Regelmäßigkeit" besitzen, ist es möglich, mit Interpolation ein Steuergesetz zu entwerfen. Das wird in Kapitel 4 für Brocketts nicht-holonomen Doppelintegrator vollbracht. Dieses Modell wird ebenfalls mit einem Regelgesetz asymptotisch stabilisiert, welches auf einer Schraubenlinie mit möglichst geringer Bogenlänge basiert.

Die Arbeit umfasst fünf Kapitel, welche wie folgt gegliedert sind:

Kapitel 2 gibt die Vorstellung der Brockett-Bedingung einschließlich dem Beweis. Ein Bespiel zur Anwendung der Bedingung wird auch diskutiert.

In Kapitel 3 schließt sich die Erklärung des Pakets *PyTrajectory* an. Weiterhin wird die Funktion des Pakets für die Bestimmung der optimalen Überführungszeit erweitert.

Kapitel 4 befasst sich mit dem Entwurf des Steuer- und Regelgesetz für Brocketts nicht-holonomen Doppelintegrator.

Schließlich folgt in Kapitel 5 eine Zusammenfassung dieser Arbeit und ein Ausblick.

Kapitel 2

Brockett-Bedingung für unteraktuiertes mechanisches System

In diesem Kapitel wird das berühmte Brockett Theorem vorgestellt. Es ist wie folgend strukturiert: Abschnitt 2.1 geht es um die Erklärung einiger mathematischen Grundbegriffen, damit die Leser den Beweisvorgang für Brockett-Bedingung leicht verstehen können. Abschnitt 2.2 befasst sich mit der Beweisskizze des Theorems.

2.1 Mathematische Grundbegriffe

In Anbetracht von der mathematische Beschreibung und dem Beweis für Brockett-Bedingung ist zuerst die Erklärung einiger mathematischen Terme notwendig.

Def 2.1. ***Stetigkeit (engl.: continuity)*** *[7, S. 250]: Es sei $a \subseteq M$. Die Funktion $f : M \subseteq \mathbb{R} \to \mathbb{R}$ ist genau dann im Punkt a stetig, wenn es zu jeder reellen Zahl $\varepsilon > 0$ eine reelle Zahl $\delta > 0$ gibt, sodass $|f(x) - f(a)| < \varepsilon$ für alle $x \subseteq M$ mit $|x - a| < \delta$ gilt.*

D.h. eine stetige Funktion erfüllt die Bedingung: wenn der Abstand zweier Elemente der Definitionsmenge infinitesimal ist, muss der Abstand ihrer entsprechenden Funktionswerte auch infinitesimal sein.

Def 2.2. ***Stetige Differenzierbarkeit (engl.: continuously differentiable)*** *[21, S. 256]: Eine differenzierbare Abbildung $\boldsymbol{f} : E \subset \mathbb{R}^n \to \mathbb{R}^m$ heißt stetig differenzierbar in E, wenn $\boldsymbol{f'}$ eine stetige Abbildung von E in $L(\mathbb{R}^n, \mathbb{R}^m)$ ist, wobei E eine offene Menge und $L(X, Y)$ der Raum linearer Abbildungen ist.*

Def 2.3. ***Klasse C^k (engl.: class C^k)*** *[7, S. 265]: Eine Funktion gehört zur Klasse C^k, wenn sie auf einer offenen Umgebung des Punktes p stetige Ableitungen bis zur Ordnung k besitzt.*

Basierend auf der obigen Definition hat eine Funktion aus der Klasse C^1 immer eine erste Ableitung, die auch stetig ist.

Def 2.4. ***Glatte Funktion (engl.: smooth function)*** *[25, S. 5]: Ein Synonym für C^∞ ist* glatt.

Eine glatte Funktion ist nämlich eine Funktion mit stetigen Ableitungen bis zur unendlichen Ordnung.

Def 2.5. ***Surjektiv (engl.: onto)*** *[7, S. 931]: Gegeben sei die Abbildung $f : X \to Y$. Betrachtet die Gleichung $f(x) = y$. Wenn die Gleichung für jedes $y \in Y$ eine Lösung $x \in X$ besitzt, d.h. $f(X) = Y$, dann heißt f genau dann* surjektiv.

Mit anderen Worten: eine Abbildung ist surjektiv, wenn jedes Element y der Wertemenge Y erreicht werden kann.

Def 2.6. ***Häufungspunkt (engl.: limit point)*** *[21, S. 35]: Ein Punkt p ist ein* Häufungspunkt *der Menge E, wenn in jeder Umgebung von p ein Punkt $q \in E$ mit $q \neq p$ liegt.*

Def 2.7. ***Abgeschlossene Menge (engl.: closed set)*** *[21, S. 36]: E heißt abgeschlossen, wenn jeder Häufungspunkt von E in E liegt.*

Def 2.8. ***Beschränkte Menge (engl.: bounded set)*** *[21, S. 36]: E ist beschränkt, wenn eine reelle Menge M und ein Punkt $q \in X$ existieren, sodass der Abstand von (p, q) kleiner als M für alle $p \in E$ gilt. X ist hier ein metrischer Raum, dessen Teilmenge E ist.*

Def 2.9. ***Kompakte Menge (engl.: compact set)*** [1] *[21, S. 45]: Falls eine Menge E in $\mathbb{R}^k$ abgeschlossen und beschränkt, dann heißt sie kompakt.*

Z.B. das Intervall $[1, 2]$ ist eine kompakte Menge in $\mathbb{R}$ mit 1 und 2 jeweils dem linken und rechten Häufungspunkt.

Def 2.10. ***Niveaumenge (engl.: level set)*** *[25, S. 94]: Eine Niveaumenge einer Abbildung $f : N \to M$ ist die Submenge $f^{-1}(c) = \{p \in N \mid f(p) = c\}$ für $c \in M$.*

Also die Niveaumenge $f^{-1}(c)$ besteht aus der Elementen der Definitionsmenge, deren Bildmenge die Konstante c ist.

Def 2.11. ***Fixpunktsatz von Brouwer (engl.: Brouwer fixed-point theorem)*** *[9, S. 7]: Sei $K \subset \mathbb{R}^n$ eine abgeschlossene Kugel und $f : K \to K$ eine stetige Abbildung, dann besitzt f mindestens einen Fixpunkt p mit $f(p) = p$.*

Für K eine kompakte homotopie-äquivalente Kugel gilt der Satz auch.

[1] Um die Leser den Beweis von Brockett Bedingung leicht versteht, wird ein Korollar von Kompakter Menge zitiert. Eine allgemeine Definition davon ist: .

Def 2.12. ***Lipschitzstetigkeit (engl.: Lipschitz continuity)*** *[3, S. 553]: Als Lipschitz-Bedingung bezüglich x ist die Forderung $|f(x,t) - f(y,t)| \leq L|x-y|$ für alle (x,t) und (y,t) bezeichnet. Dabei ist L eine beliebige Konstante.*

Das heißt, wenn die Ableitung der Funktion von f beschränkt ist, impliziert das eine Lipschitz-Bedingung. Die Lipschitzstetigkeit ist stärker als Stetigkeit.

Def 2.13. ***Asymptotische Stabilität (engl.: asymptotically stable)*** *[11, S. 112]: Die Ruhelage $x = 0$ ist stabil, wenn für jede $\varepsilon > 0$ existiert ein $\delta = \delta(\varepsilon) > 0$, der $\|x(0)\| < \delta \Rightarrow \|x(t)\| << \varepsilon, \forall t \geq 0$ erfüllt.*

Falls die Ruhelage stabil ist, und ein δ kann so ausgewählt, dass $\|x(0)\| < \delta \Rightarrow \lim_{t\to\infty} x(t) = 0$ erfüllt, dann ist diese Ruhelage asymptotisch stabil.

2.2 Erläuterung zur Brockett-Bedingung

Ein nichtlineares Zustandsraummodell lässt sich durch

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad t \geq 0, \quad \boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \quad \boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{0}) = \boldsymbol{0} \tag{2.1}$$

darstellen, mit $\boldsymbol{x}$ dem n-dimensionalen Systemzustand, $\boldsymbol{f}$ der nichtlinearen Zustandsfunktion, $\boldsymbol{u}$ der m-dimensionalen Eingangsgröße und $\boldsymbol{x}_0$ dem initialen Zustand.

Jetzt stellt sich die Frage: gibt es die Möglichkeit, dass dieses nichtlineare System um die Ruhelage $\boldsymbol{x} = \boldsymbol{x}_e$ mit einer nichtlinearen Zustandsrückführung (nämlich $\boldsymbol{u}(\boldsymbol{x})$) asymptotisch stabilisiet werden kann? Zum Beantworten der Frage etablierte der amerikanische Mathematiker Roger W. Brockett das folgende berühmte Kriterium [2]:

Theorem 2.1 (Brockett-Bedingung [2])**.** *Betrachtet wird das System (2.1), wobei $\boldsymbol{f}$ stetig differenzierbar in der Umgebung von $(\boldsymbol{x}_e, \boldsymbol{0})$ ist. Angenommen, dass $(\boldsymbol{x}_e, \boldsymbol{0})$ in $\mathbb{R}^n \times \mathbb{R}^m$ asymptotisch stabil unter einem stetigen differenzierbaren Regelgesetz $\boldsymbol{u}$ ist, dann sind folgende notwendigen Bedingungen erfüllt:*

1. *Das linearisierte System enthält keine nicht steuerbares Teilsystem, dessen Eigenwerte in der rechte Halbebene liegen.*

2. *In einer Umgebung N von $(\boldsymbol{x}_e, 0)$ existiert für jeden $\boldsymbol{\xi} \in N$ immer ein zum Intervall $[0, \infty)$ gehörendes Regelgesetz $\boldsymbol{u}_{\boldsymbol{\xi}}$, womit die Lösung von $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_\xi)$ von $\boldsymbol{x} = \boldsymbol{\xi}\,|_{t=0}$ nach $\boldsymbol{x} = \boldsymbol{x}_e\,|_{t=\infty}$ überführt wird.*

3. *Das Bild der Abbildung*

$$\Gamma : (\boldsymbol{x}, \boldsymbol{u}) \mapsto \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$$

ist surjektiv bezüglich einer offenen Menge, die den Punkt $\boldsymbol{0} \in \mathbb{R}^n$ enthält.

Bevor eine Beweisskizze des Satzes erfolgt, sollen die drei Bedingungen kurz erläutert werden.

Die erste notwendige Bedingung bedeutet es, wenn $(\boldsymbol{x}_e, 0)$ asymptotisch stabil ist, hat das linearisierte System nur steuerbare Teilsysteme (mit oder ohne Eigenwerte in der rechte Halbebene) oder nicht steuerbare Teilsysteme mit nur stabilen Eigenwerte. Mit anderen Worten müssen instabile Teilsysteme steuerbar oder nicht steuerbare Teilsystem müssen stabil sein. Für eines kritisches System (nämlich einige Eigenwerte auf der imaginäre Achse liegen.) ist es nicht möglich, nur mit dieser Bedingung die asymptotische Stabilität der Ruhelage zu überprüfen. Die 2.- und 3. notwendige Bedingung sind wichtig für kritische Situationen.

Die zweite Bedingung ist ähnlich wie die Erläuterung der Steuerbarkeit eines Systems: das System kann mit einer Steuervektor $\boldsymbol{u}_t$ von einem beliebigen Anfangszustand (in einer Umgebung von der Ruhelage) in einem *ausgewählten* Endzustand (hier Ruhelage) überführt werden (aber die Überführungszeit kann bis zur unendlich sein).

Die surjektive Abbildung Γ in der dritten Bedingung bedeutet, dass die Zustandsgleichung $\boldsymbol{f}$ beliebigen/irgendeinen Wert in der Nähe von $\mathbf{0}$ erhalten kann.

Ein leicht verwirrender Punkt dieses Kriterium liegt in die Bedingungen von $\boldsymbol{f}$ und u in Gl. (2.1). Nach der Beschreibung des Theorems sind beide $\boldsymbol{f}$ und u *stetig differenzierbar*, und zitiert Brockett in ihrem Beweis aus [26, S.324] eine *stetig* differenzierbare Lyapunov Funktion, die aber im Original *glatt* ist. In anderen Literaturen sind auch unterschiedliche Annahmen ermöglicht: [5] und [19] setzen $\boldsymbol{f}$ und u *stetig und zeitinvariant* als bekannt voraus, während der von [24] einen strengeren Bedingung für $\boldsymbol{f}$ und $\boldsymbol{u}$ vorbringen:*lokal lipschitz*. Im Buch vom argentinischen Mathematiker Eduardo D. Sontag [22] werden die Bedingung von $\boldsymbol{f}$ und u gleich wie Brockett (C^1). Eine noch strengere Voraussetzung werde von G. Oriolo und Y. Nakamura in [18] aufgestellt, dass $\boldsymbol{f}$ *stetig differenzierbar* und u *glatt* sein muss. Als eine Zusammenfassung wird die folgende Tabelle aufgelistet:

Skizze des Beweises *([2],[15],[11],[20]).*

1. Ausgangspunkt ist die logische Beziehung: falls $A \Rightarrow B$ dann $\neg B \Rightarrow \neg A$. Betrachtet man das Theorem von Lyapunov Stabilität aus [11]: Sei $\boldsymbol{x} = \mathbf{0}$ eine Ruhelage vom nichtlinearen System $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ mit $\boldsymbol{f} : D \mapsto \mathbb{R}^n$ stetig differenzierbar und D einer Umgebung von der Ruhelage. $\mathbf{A}$ ist Systemmatrix. Dann ist die Ruhelage nicht stabil, wenn mindestens ein Eigenwert von $\mathbf{A}$ einen Realteil größer als 0 besitzt. Es ist auch bekannt, die Ruhelage in einem *steuerbaren* Teilsystem kann mit einer linearen Rückführung $u = K\boldsymbol{x}$ stabilisiert werden (wobei K so gewählt wird, dass alle Eigenwerte in der linken Halbebene liegen). Die Kombination von den zwei Argumenten beweist 1. Bedingung.

2. Unter der Berücksichtigung der Definition der Asymptotischen Stabilität ist die 2. Bedingung einfach bewiesen. Die asymptotische Stabilität (sieh. Def. 2.13) eines

Tabelle 2.1 – Die Verständnisse der Bedingung von Brockett Theorem in unterschiedlichen Literaturen

Literatur	Zeit	Bedingung
R. W. Brockett [2]	1983	Zustandsfunktion $\boldsymbol{f}$ stetig differenzierbar, Regelgesetz $\boldsymbol{u}(\boldsymbol{x})$ stetig differenzierbar und zeitinvariant. (Lyapunov Funktion V stetig differenzierbar.)
F. W. Wilson [26]	1967	$\boldsymbol{f}$ stetig differenzierbar, Lyapunov Funktion V glatt.
G. Oriolo, Y. Nakamura [18]	1991	$\boldsymbol{f}$ stetig differenzierbar, $\boldsymbol{u}(\boldsymbol{x})$ glatt.
E. D. Sontag [22, S. 252]	1998	$\boldsymbol{f}$ und $\boldsymbol{u}$ stetig differenzierbar.
R. J. Stern [24]	2002	$\boldsymbol{f}$ und $\boldsymbol{u}$ lipschitz stetig. (In der Literatur gibt es auch einen detaillierten Beweis davon.)
R. Orsi, L. Praly, I. Mareels [19]	2003	$\boldsymbol{f}$ stetig, Regelgesetz $\boldsymbol{u}$ stetig. (In der Literatur gibt es auch einen detaillierten Beweis davon.)
F. Colonius [4, S. 57]	2012	$\boldsymbol{f}$ lokal lipschitz stetig, $\boldsymbol{u}$ stetig differenzierbar und zeitinvariant.

geregelten Systems impliziert, die Trajektorie des Systemzustands mit irgendeinen Anfangswerten in einer Umgebung von Ruhelage kann immer in der Beschränkung ε bleiben und endlich zur Ruhelage überführt werden. Mit anderen Worten ist die Definition von asymptotischen Stabilität strenger als die Bedingung hier.

3. Nimmt man zuerst die Zustandsfunktion $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{a}(\boldsymbol{x})$ an. Falls die Ruhelage $(\boldsymbol{x}_e, \boldsymbol{0})$ asymptotisch stabil ist, existiert es nach [26] eine glatte Lyapunov Funktion V, welche eine Niveaumenge $V^{-1}(c)$ (c eine kleine Konstante) besitzt, die homotopieäquvalent zu einer Kugel ist. Damit V eine Lyapunov Funktion V ist, muss $\dot{V} = \frac{\mathrm{d}V}{\mathrm{d}t} = \frac{\mathrm{d}V}{\mathrm{d}\boldsymbol{x}} \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\mathrm{d}V}{\mathrm{d}\boldsymbol{x}} \cdot \dot{\boldsymbol{x}} < 0$ gelten. Nämlich zeigt der Vektor $\boldsymbol{a}(\boldsymbol{x})$ **in** das Innere der Menge $R := \{\boldsymbol{x} : V(\boldsymbol{x}) \leq c\}$. Es existiert auch $\boldsymbol{\xi} \in \mathbb{R}^n$ mit $\|\boldsymbol{\xi}\|$ genügend **klein**, dass $\boldsymbol{a}(\boldsymbol{x}) - \boldsymbol{\xi}$ auch **in** R zeigt. Deswegen gibt es innerhalb von R eine stetige Abbildung $\Psi : R \to R$.

   Mit dem Fixpunktsatz von Brouwer[2] gilt es $f(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{\xi} = \boldsymbol{0}$ (oder $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \xi$) mit $\boldsymbol{x}$ der Ruhelage des Systems. (Die Abblidung Ψ ist eine Fluss-Abbildung und B eine invariante Menge bezüglich davon. Dann existiert ein Fixpunkt $\boldsymbol{x}^*$, sodass $\Psi_t(\boldsymbol{x}^*) = \boldsymbol{x}^*$ ist. Das heißt, der Fluss aus $\boldsymbol{x}^*$ bleibt immer auf $\boldsymbol{x}^*$. Deswegen ist $\boldsymbol{x}^*$ die Ruhelage des Systems und $\dot{\boldsymbol{x}}^* = \boldsymbol{a}(\boldsymbol{x}^*) = \boldsymbol{0}$.) Weil $\|\boldsymbol{\xi} - \boldsymbol{0}\|$ beliebig sehr klein ist, bedeutet $\boldsymbol{f}$ surjektiv bezüglich einer Umgebung von $\boldsymbol{0}$ ist. (Und die Lösbarkeit von $\boldsymbol{f}$ impliziert die Lösbarkeit von $\boldsymbol{u}$.)

[2] In den Literatur von Brockett verwendet er *Lefschetz* Fixpunktsatz zu beweisen.

□

Hier ergibt sich ein Beispiel, das die Brockett-Bedingung nicht erfüllt.

Beispiel 2.1 (Brocketts Doppelintegrator)**.** *Betracht man zuerst einen mobilen Roboter mit einen flexiblen beweglichen Voreinrad in Abb. 2.1. Die zwei Hinterräder können nur entlang der zu der Hinterachse senkrechten Richtung rollen. Gleiten parallel zu der Hinterachse ist in diesem System unmöglich. Die Systemzustandsfunktion ist wie Gl. (2.2) gezeigt.*

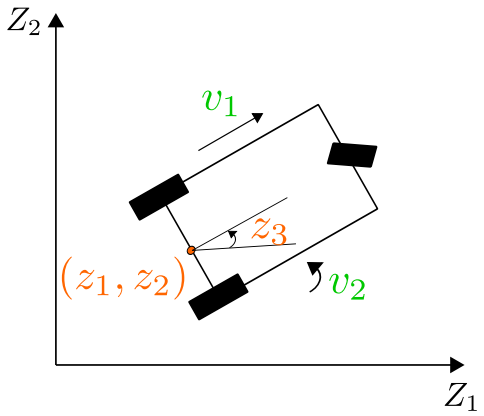


Abbildung 2.1 – Skizze des mobilen Roboters mit einem Voreinrad. (z_1, z_2, z_3) sind die Systemzustände und (v_1, v_2) sind die Systemeingänge.

$$\begin{aligned} \dot{z}_1 &= v_1 cos(z_3) \\ \dot{z}_2 &= v_1 sin(z_3) \\ \dot{z}_3 &= v_2. \end{aligned} \tag{2.2}$$

Das System hat 3 Zustandskomponente: z_1 und z_2 steht jeweils für die Position des Punktes in der Mitte der der Hinterachse. z_3 ist der Winkel des Vorderrades mit der z_1-Achse. In dem System ergibt sich zwei Eingänge v_1 und v_2, die die Vorwärts- und Winkelgeschwindigkeit bedeuten.

Mit der Rückkopplung der Systemzuständen und Eingängen [15]:

$$\begin{aligned} x_1 &= z_1 \cos(z_3) + z_2 \sin(z_3) \\ x_2 &= z_3 \\ x_3 &= 2 \cdot (z_1 \sin(z_3) - z_2 \cos(z_3)) - z_3(z_1 \cos(z_3) + z_2 \sin(z_3)) \\ u_1 &= v_2 \\ u_2 &= v_1 - v_2(z_1 \sin(z_3) - z_2 \cos(z_3)) \end{aligned} \tag{2.3}$$

kann Gln. (2.2) in der Form wie folgt transformiert werden:

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 u_1 - x_1 u_2. \end{aligned} \tag{2.4}$$

Diese Zustandsfunktion beschreibt das berühmte System "Brocketts nicht-holonomer Integrator". Das Zustandsraummodell dieses Systems davon lautet:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -u_{2,e} & u_{1,e} & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ x_{2,e} & -x_{1,e} \end{bmatrix} \boldsymbol{u} \tag{2.5}$$

Ersetzt man den linken Teil von Gln. (2.5) mit $\mathbf{0}$*, erhält man die Ruhelagen dieses Systems:* $(x_1, x_2, x_3) = (c_1, c_2, c_3)$ *beliebig mit* $u_1 = u_2 = 0$*. Das heißt, die Eigenwerte des Systems sind alle* 0*,* ***ohne*** *positive Realteile zu verfügen. Die Brockett* 1*. Bedingung ist nicht verletzt.*

Zur Konstruktiven der 2*. Bedingung kann man einen Regler wie folgt entwerfen:*

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow[Phase-I,1s]{\boldsymbol{u}_1} \begin{pmatrix} * \\ * \\ 0 \end{pmatrix} \xrightarrow[Phase-II,1s]{\boldsymbol{u}_2} \begin{pmatrix} * \\ * \\ \omega \end{pmatrix} \xrightarrow[Phase-III,1s]{\boldsymbol{u}_3} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{2.6}$$

mit $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$ *drei Regelgesetzen (nicht Regler* u_1 *oder* u_2 *in Gl. (2.4)) in drei Phasen (Nimmt man an, dass jede Phase* $1s$ *dauert.) und* $\begin{pmatrix} * & * & 0 \end{pmatrix}^T, \begin{pmatrix} * & * & \omega \end{pmatrix}^T$ *zwei Hilfspunkte. Es ist schon bekannt, wenn eine Trajektorie von Punkt A nach B existiert, gibt es unbedingt eine umgekehrte Trajektorie von B nach A. Ohne die Einfachheit zu verlieren, konstruiert man hier die Trajektorie von der Ursprung* ($\boldsymbol{x} = \mathbf{0}$) *nach einem beliebigen Punkt* ($\boldsymbol{x}_{end} = (\alpha, \beta, \gamma)$)*. In Phase-I soll der Zustandsvektor mit* $\boldsymbol{u}_1$ *aus der Ruhelage zu den Hilfspunkt* 1 *gehen, dann soll er in* $1s$ *zu den Hilfpunkt* 2 *erreichen. Und endlich soll er mittels* $\boldsymbol{u}_3$ *am beliebigen Punkt* $\begin{pmatrix} \alpha & \beta & \gamma \end{pmatrix}^T$ *stoppen können.*

EinAnsatz mit Konstanten u_1 *und* u_2 *wie:*

$$\begin{aligned} Phase - I: &\ u_1 = a_1, u_2 = b_1 \\ Phase - II: &\ u_1 = a_2, u_2 = b_2 \\ Phase - III: &\ u_1 = a_3, u_2 = b_3 \end{aligned} \tag{2.7}$$

ist genügend, die obere Trajektorienbedingungen zu erfüllen. Zum Zeitpunkt $t = 3s$ *gilt:*

$$\begin{aligned} x_1(3s) &= a_1 + a_2 + a_3 \\ x_2(3s) &= b_1 + b_2 + b_3 \\ x_3(3s) &= (b_1 + b_2)a_3 - (a_1 + a_2)b_3 + b_1 a_2 - b_2 a_1. \end{aligned} \tag{2.8}$$

Mit diesem Regelgesetz kann das System in $3s$ *irgendwo außerhalb dem Punkt* $(\alpha = 0, \beta = 0, \gamma = 0)^T$ *ankommen. Um diesen Endzustand erreichen zu können, muss* ω *in Hilfspunkt* 2 *äquivalent zu* γ *sein. Das ist die einzelne Anforderung von der Größe von* (α, β, γ)*. Deswegen kann das System mit diesem Regelgesetz die Brockett* 2*. Bedingung erfüllen.*

Letztlich wird die 3. Bedingung überprüft. Setzt man voraus, dass $\boldsymbol{\epsilon}$ *in der offenen Menge* $M = B(\boldsymbol{x}, \varepsilon)$ *liegt, dann gilt*

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} u_{1\epsilon} \\ u_{2\epsilon} \\ x_{2\epsilon}u_{1\epsilon} - x_{1\epsilon}u_{2\epsilon} \end{bmatrix}. \tag{2.9}$$

Die Menge M *enthält einen Punkt* $\boldsymbol{x}_{um} = (0, 0, \delta)$ *mit* $\delta < \varepsilon$*. Offensichtlich ist es unmöglich, das Bild der Abbildung* $\Gamma : (\boldsymbol{x}, u) \mapsto \boldsymbol{f}(\boldsymbol{x}, u)$ *einen Wert gleich* $\boldsymbol{x}_{um}$ *zu sein. Nämlich ist* Γ *nicht surjektiv zu* M*.*

D.h. das System verletzt die Brockett Bedingung. Deswegen existiert kein stetig differenzierbarer Regelgesetz, mit dem der Ursprung asymptotisch stabil ist.

Fazit: In diesem Kapitel wurde das Brockett Theorem für die Überprüfung der Existenz einer asymptotischen stabilen Ruhelage in einem nichtlinearen System mittels eines stetigen differenzierbaren Regelgesetz vorgestellt. In den Literatur [2] lieferte Brockett schon einen Beweis für das Theorem, der aber für die Leser ohne tiefen Kenntnisse der Topologie vergleichsweise schwer verständlich ist. Daher erklärt die Autorin hier den Beweisvorgang mit einfacheren Worten. Ein einfaches nichtholonom Doppelintegrator System wurde als Beispiel für die Anwendung der Brockett-Bedingung diskutiert.

Kapitel 3

Trajektorieplanung durch Lösung einer Randwertaufgabe

Dieses Kapitel befasst sich mit der Trajektorieplanung von dem nichtlinearen System mittels des Python-Pakets *PyTrajectory*. Im ersten Abschnitt geht es um die Vorstellung der allgemeinen Idee und den Algorithmen für die Trajektorieplanung mit dem Paket. In Abschnitt 3.2 werden einige Probleme mit den Lösungen aus den Algorithmen diskutiert. Die simulierten Ergebnisse verschiedener Systeme werden auch in diesem Abschnitt vorgestellt.

3.1 Grundlage zum Entwurf der Trajektorieplanung

PyTrajectory ist ein Python Paket zum Trajektorien-Entwurf mit gegebenen Randwertbedingungen (Anfangs- und Endwerte der Systemzuständen sowie Eingängen) für nichtlineare Systeme. Hier wird die Grundlage über Trajektorienplanung grundsätzlich vorgestellt. Für weitere Informationen steht die Dokumentation der Software [1] zur Verfügung.

3.1.1 Kollokationsverfahren

Gesucht ist die Lösung $t \mapsto \boldsymbol{u}(t)$ des Randwertproblems:

$$\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) - \dot{\boldsymbol{x}} \;=\; \boldsymbol{0}, \quad t \in [0, T] \tag{3.1}$$

mit dem Anfangswert $\boldsymbol{x}_0 = \boldsymbol{X}_a$, Endwert $\boldsymbol{x}_T = \boldsymbol{X}_b$ an. $\dot{\boldsymbol{x}}$ ist die Ableitung nach der Zeit t, $\boldsymbol{X}_a \in \mathbb{R}^n, \boldsymbol{X}_b \in \mathbb{R}^n$ sind gegeben und $\boldsymbol{u} : \mathbb{R} \mapsto \mathbb{R}^m$ bezeichnet das gesuchte Eingangssignal.

Das Zeitintervall $I = [0, T]$ kann wie Abb. 3.1 unterteilt werden:

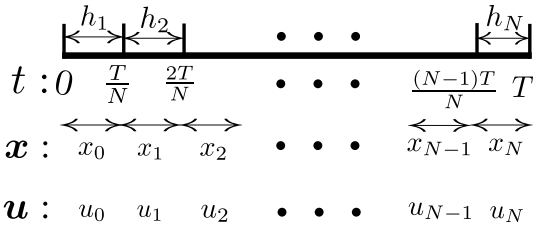


Abbildung 3.1 – Kollokationsabschnitte h_i und Kollokationspunkte $\boldsymbol{x}_i$, $\boldsymbol{u}_i$: für jeden Abschnitt ist die Darstellung von $\boldsymbol{x}$ und $\boldsymbol{u}$ jeweils ein Polynom über t. Die Anzahlen der Splineabschnitte von $\boldsymbol{x}$ und $\boldsymbol{u}$ brauchen nicht äquivalent zu sein. (In dieser Abbildung sind sie als spezieller Fall gleich.)

Das Intervall $[0, T]$ wird in N Teile mit gleichen Abstand eingeteilt ($h_1 = h_2 = \cdots = h_N = \frac{T}{N}$). Für jeden Abschnitt nimmt man eine Polynomfunktion dritten Grades jeweils für $\boldsymbol{x}$ und $\boldsymbol{u}$ an:

$$\begin{aligned}
\boldsymbol{S}_{x,i}(\boldsymbol{c}, t) = \boldsymbol{x}_i &= c_{xji0} + c_{xji1} \cdot (t - \frac{iT}{N_x}) + c_{xji2} \cdot (t - \frac{iT}{N_x})^2 \\
&+ c_{xji3} \cdot (t - \frac{iT}{N_x})^3, \quad t \in [\tfrac{iT}{N_x}, \tfrac{(i+1)T}{N_x}), \quad i = 0 \cdots N_x - 1, \quad j = 1 \cdots n \\
\boldsymbol{S}_{u,i}(\boldsymbol{c}, t) = \boldsymbol{u}_i &= c_{uji0} + c_{uji1} \cdot (t - \frac{iT}{N_u}) + c_{uji2} \cdot (t - \frac{iT}{N_u})^2 \\
&+ c_{uji3} \cdot (t - \frac{iT}{N_u})^3, \quad t \in [\tfrac{iT}{N_u}, \tfrac{(i+1)T}{N_u}), \quad i = 0 \cdots N_u - 1, \quad j = 1 \cdots m
\end{aligned} \tag{3.2}$$

Der Index i und j steht jeweils für den i-ten Abschnitt in Abb. 3.1 und die $(j+1)$-te Komponente von x bzw. u (Da $\boldsymbol{x}/\boldsymbol{u}$ Vektor ist). $\boldsymbol{c}$ bedeutet Koeffizient der Polynomen. Es gibt insgesamt $4 \cdot (N_x \cdot n + N_u \cdot m)$ Koeffizienten von $\boldsymbol{x}$ und $\boldsymbol{u}$ in $(N_x \cdot n + N_u \cdot m)$ kubischen Polynomfunktionen.

Jetzt berücksichtigt man die Stetigkeit von $\boldsymbol{S}$. Die Polynomfunktionen sollen an der Abschnittsgrenzen zweimal stetig differenzierbar sein. Mit anderen Worten erfüllen sie in jeder Übergangsstelle folgende Formel:

$$\begin{aligned}
\boldsymbol{S}_{x/u,i}\left(\boldsymbol{c}, t = \frac{iT}{N_x/N_u}\right) &= \boldsymbol{S}_{x/u,i+1}\left((\boldsymbol{c}, t = \frac{iT}{N_x/N_u}\right) && \text{(0. Ableitung stetig)} \\
\dot{\boldsymbol{S}}_{x/u,i}\left(\boldsymbol{c}, t = \frac{iT}{N_x/N_u}\right) &= \dot{\boldsymbol{S}}_{x/u,i+1}\left((\boldsymbol{c}, t = \frac{iT}{N_x/N_u}\right) && \text{(1. Ableitung stetig)} \\
\ddot{\boldsymbol{S}}_{x/u,i}\left((\boldsymbol{c}, t = \frac{iT}{N_x/N_u}\right) &= \ddot{\boldsymbol{S}}_{x/u,i+1}\left((\boldsymbol{c}, t = \frac{iT}{N_x/N_u}\right) && \text{(2. Ableitung stetig).}
\end{aligned} \tag{3.3}$$

Deswegen gibt es insgesamt $3((N_x - 1) \cdot n + (N_u - 1) \cdot m)$ Funktionen mit $(N - 1)$ der Anzahl der Übergangsstellen. Mit dem Anfangs- und Endwert von $\boldsymbol{x}$ und $\boldsymbol{u}$ (insgesamt $2 \cdot (n+m)$) werden $N_{cf} = (n \cdot (N_x + 1) + m \cdot (N_u + 1))$ freie Parameter $\boldsymbol{c}_f$ für Koeffizienten

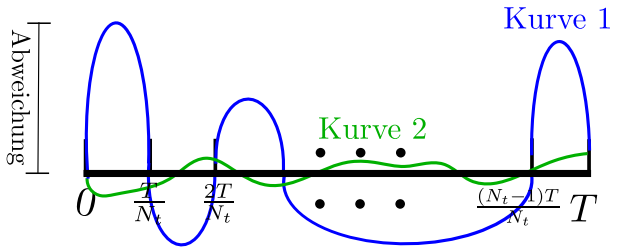


Abbildung 3.2 – 2 Trajektorienfälle von beispielsweise dem Systemzustand: aus Grund der Einfachheit ist $dim(\boldsymbol{x}) = 1$ gewählt. Die y-Achse ist die Abweichung zwischen der approximierten Lösung und dem idealen Fall. Zwar geht die Kurve 1 zu den gewählten Zeitpunkten die x-Achse durch, aber sie hat eine viel größer Abweichung zu anderen Punkten als Kurve 2.

sichergestellt. Die anderen $N_{ca} = (n \cdot (3N_x - 1) + m \cdot (3N_u - 1))$ Parameter von Polynomen $\boldsymbol{c}_a$ hängen durch die obengenannten Funktionen von $\boldsymbol{c}_f$ ab.

Der freie Parametervektor $\boldsymbol{c}_f$ kann durch (3.1) bestimmt werden. Dazu setzt man die Polynomform von $\boldsymbol{x}$ und $\boldsymbol{u}$ in Gl. (3.1) ein und wählt man $(N_t + 1)$ beliebigen Zeitpunkten[1]:

$$\boldsymbol{F}(\boldsymbol{c}_f) = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{S}_x(\boldsymbol{c}_f, t_0), \boldsymbol{S}_u(\boldsymbol{c}_f, t_0)) - \dot{\boldsymbol{S}}_x(\boldsymbol{c}_f, t_0) = \boldsymbol{0} \\ \boldsymbol{f}(\boldsymbol{S}_x(\boldsymbol{c}_f, t_1), \boldsymbol{S}_u(\boldsymbol{c}_f, t_1)) - \dot{\boldsymbol{S}}_x(\boldsymbol{c}_f, t_1) = \boldsymbol{0} \\ \cdots \\ \boldsymbol{f}(\boldsymbol{S}_x(\boldsymbol{c}_f, t_{N_t}), \boldsymbol{S}_u(\boldsymbol{c}_f, t_{N_t})) - \dot{\boldsymbol{S}}_x(\boldsymbol{c}_f, t_{N_t}) = \boldsymbol{0} \end{pmatrix} = \boldsymbol{0} \tag{3.4}$$

$$\text{mit} \quad \boldsymbol{f} \quad : \quad \mathbb{R}^{N_{cf}} \mapsto \mathbb{R}^n.$$

Die Dimension der unabhängigen Variablen $\boldsymbol{c}_f$ ist N_{cf} und die Dimension von $\boldsymbol{F}(\boldsymbol{c}_f)$ ist $n \cdot (N_t + 1)$. Wenn die Anzahl der Variablen gleich oder größer als der Funktionen ist, wird mindestens eine exakte Lösung ausgerechnet. Das sichert die Funktionswerte zu den $(N_t + 1)$ Zeitpunkten. Aber die Werte zu anderen Zeitpunkten können nicht garantiert werden. Wie Abb. 3.2 zeigt, liegt die Kurve 1 zu den $(N_t + 1)$ Zeitpunkten gerade auf der Achse, aber zu anderen Punkten liegt sie sehr weit von der Achse. Dagegen erhält die Kurve 2 zu jedem Zeitpunkt zwar keine exakte Lösung, aber die Norm ist sehr klein. Deswegen sollte man für das Kollocationsverfahren eine überbestimmte Funktion auswählen, nämlich $n \cdot (N_t + 1) > (n \cdot (N_x + 1) + m \cdot (N_u + 1))$.

Jetzt stellt sich die Frage: wie kann man eine optimale approximierte Lösung von der überdeterminierte Funktion (Gl. (3.5)) bestimmen? Das wird in nächsten Abschnitt beantwortet.

[1] In *PyTrajectory* wählt man Zeitpunkte mit festen Abstand: $t_k = \frac{kT}{N_t}, k = 0 \cdots N_t$.

3.1.2 Methode zur Lösung des Quadratmittelproblems

Das Quadratmittelproblem einer Funktionsvektor ist wie folgt definiert [13]:

Definition 3.1 (Quadratmittelproblem)**.** *Gegeben ist ein Funktionsvektor*[2] $\boldsymbol{F}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$*. Ein Vektor* $\boldsymbol{x}_*$ *ist zu finden, sodass die euklidische Norm des Funktionsvektors (oder: das "zugehörige Fehlerfunktional") minimiert wird:*

$$\boldsymbol{x}_* = \arg\min_{\boldsymbol{x}} |\boldsymbol{F}(\boldsymbol{x})|_2^2 = \arg\min_{\boldsymbol{x}} \left(\sum_{i=1}^{m} (f_i(\boldsymbol{x}))^2 \right) \tag{3.5}$$

mit $f_i : \mathbb{R}^n \to \mathbb{R}^1$*.* $\boldsymbol{x}_*$ *hießt auch der stationäre Punkt.*

Gauß-Newton-Verfahren

Ein solches Quadratmittelproblem kann man mit verschieden Verfahren lösen. Eine typische Methode ist das Gauß-Newton-Verfahren (GN-Verfahren), das durch die lineare Approximation des nichtlinearen Problems löst. Mit der Ableitung erster Ordnung der Funktion $\boldsymbol{F}(\boldsymbol{x})$ konvergiert das Iterationsverfahren mit linearer Konvergenzordnung[3]. Im Folgenden geht die Methode mit der Taylorentwicklung von $\boldsymbol{F}$ und f aus [16]:

$$\begin{aligned} f(\boldsymbol{x}+\boldsymbol{h}) &= f(\boldsymbol{x}) + f'(\boldsymbol{x}) \cdot \boldsymbol{h} \overset{f'=\mathbf{J}}{=} f(\boldsymbol{x}) + \mathbf{J}(\boldsymbol{x}) \cdot \boldsymbol{h} + o\left(\boldsymbol{h}^2\right) \\ \boldsymbol{F}(\boldsymbol{x}+\boldsymbol{h}) &= f(\boldsymbol{x}+\boldsymbol{h})^{\mathrm{T}} \cdot f(\boldsymbol{x}+\boldsymbol{h}) \approx (f+\mathbf{J}\boldsymbol{h})^{\mathrm{T}} (f+\mathbf{J}\boldsymbol{h}) \\ &= f^{\mathrm{T}} f + 2\boldsymbol{h}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}} f + \boldsymbol{h}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{J}\boldsymbol{h} := \boldsymbol{G}(\boldsymbol{h}). \end{aligned} \tag{3.6}$$

Das Symbol $\mathbf{J}$ steht für die Jacobi-Matrix. Nach der Ableitung des vorletzten Terms erhält man die 1. und 2. Ableitung von $\boldsymbol{G}$ bezüglich $\boldsymbol{h}$:

$$\boldsymbol{G}'(\boldsymbol{h}) = 2\mathbf{J}^{\mathrm{T}} f + 2\mathbf{J}^{\mathrm{T}}\mathbf{J}\boldsymbol{h} \tag{3.7}$$

$$\boldsymbol{G}''(\boldsymbol{h}) = 2\mathbf{J}^{\mathrm{T}}\mathbf{J}. \tag{3.8}$$

Es wir angenommen, dass $\boldsymbol{G}''$ positiv definit ist[4]. Man definiert hier die Schrittweite $h = h_*$ als lokaler Minimum, wenn $\boldsymbol{G}'(\boldsymbol{h}_*) = \mathbf{0}$ ist. Zur Berechnung von $\boldsymbol{h}_*$ wird Gl. (3.7) gleich 0 gesetzt:

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right) \cdot \boldsymbol{h}_* = -\mathbf{J}^{\mathrm{T}} f. \tag{3.9}$$

Der stationäre Punkt $\boldsymbol{x}_*$ wird aus der vorherige Punkt $\boldsymbol{x}_{*-1}$ und $\boldsymbol{h}_*$ berechnet:

$$\boldsymbol{x}_* = \boldsymbol{x}_{*-1} + \boldsymbol{h}_* = \boldsymbol{x}_{*-1} - \left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right)^{-1} \cdot \mathbf{J}^{\mathrm{T}} f. \tag{3.10}$$

[2] n und m hier sind anders wie die Dimensionen von $\boldsymbol{x}$ und $\boldsymbol{u}$ in letztem Abschnitt. $\boldsymbol{x}$ verweist nur die erforderlichen Parameter.

[3] Linear Konvergenzordnung bedeutet, dass $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_*\| \leqslant \alpha \|\boldsymbol{x}_k - \boldsymbol{x}_*\|$ mit $0 \leqslant \alpha \leqslant 1$.

[4] Das ist ein Nachteil von Gauß-Newton-Verfahren. Die Positivdefinitkeit kann nicht immer gesichert werden.

Neben der möglichen Nichtexistenz der inversen Matrix gibt es auch eine Beschränkung: mit der $\boldsymbol{x}_*$ durch das GN-Verfahren nicht gefunden kann, wenn der Anfangsschätzwert $\boldsymbol{x}_0$ nicht in der Näher von $\boldsymbol{x}_*$ liegt. Daher wird in nächsten Abschnitt eine andere Methode vorgestellt, damit man die obige Probleme vermeiden kann.

Levenberg-Marquardt-Algorithmus

Der nach Kenneth Levenberg und Donald Marquardt benannte Algorithmus ist tatsächlich eine Kombination von Methode des steilsten Abstiegs und Gauß-Newton-Verfahrens. Gl. (3.11) zeigt die detaillierte Form von Schrittweite $\boldsymbol{h}$ in LM-Algorithmus [8][16]:

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\mathbf{I}\right) \cdot \boldsymbol{h} = -\mathbf{J}^{\mathrm{T}} f. \tag{3.11}$$

$\mathbf{I}$ ist die Einheitsmatrix, μ ist ein "Dämpfungsparameter". Aufgrund der Existenz des Dämpfungparameters ist die Positivdefinitheit der Matrix $\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\mathbf{I}$ gesichert. Falls μ groß ist (das passiert am Anfang der Iteration), ist die Funktion von der Schrittweite ähnlich wie Methode des steilsten Abstiegs, damit die Funktion bei $\boldsymbol{x}$ weit von Zielpunkt schnell konvergiert. Andernfalls verhält sich der LM-Algorithmus bei kleinem μ oder in der Umgebung von $\boldsymbol{x}_*$ wie NM-Verfahren aus. Das heißt, in den letzten Schritten konvergiert die Funktion auch schnell.

Ein wichtiger Teil für LM-Algorithmus ist die Bestimmung von Dämpfungsparameter μ. Der Ausgangspunkt davon ist die Größe der Variable *Verstärkungsverhältnis* ρ, die durch die Relation von der aktuellen Abnehme des nichtlinearen Systems in einem Schritt und der vorhergesagten Abnehme des linearisierten Systems definiert wird:

$$\rho = \frac{\|\boldsymbol{F}(\boldsymbol{x})\|^2 - \|\boldsymbol{F}(\boldsymbol{x}+\boldsymbol{h})\|^2}{\|\boldsymbol{G}(\mathbf{0})\|^2 - \|\boldsymbol{G}(\boldsymbol{h})\|^2} = \frac{\Delta F}{\Delta G}. \tag{3.12}$$

Da $\boldsymbol{h}$ in die Richtung mit einer sinkenden Norm von $\boldsymbol{G}$ zeigt, soll der Wert von ρ größer als 0 sein. Wenn ρ sehr klein oder ΔG viel größer als ΔF ist, ist der vorherige Schritt $\boldsymbol{h} = \min_{\boldsymbol{h}} \left(\boldsymbol{G}(\boldsymbol{h}) + \frac{1}{2}\mu\boldsymbol{h}^T \cdot \boldsymbol{h}\right)$ zu groß und er muss im nächsten Schritt verkleinert werden. Das heißt, der Wert von μ muss vergrößert werden. Anderenfalls muss μ verkleinert werden, wenn der vorherige $\boldsymbol{h}$ zu klein (sonst kostet es mehrere Zeit, eine Lösung zu finden) ist. Ein typischer Entwurf von ρ sieht wie folgendem Algorithmus:

if $\rho < 0.2$ **then**
  $\mu_{new} := \mu_{old} * 2$
else if $\rho > 0.8$ **then**
  $\mu_{new} := \mu_{old}/2$
else
  $\mu_{new} := \mu_{old}$
end if

Setzt man Gl. (3.5) in Gl. (3.11) ein, ersetzt der Vektor $\boldsymbol{F}(\boldsymbol{c}_f)$ hier f und die freien Parameter $\boldsymbol{c}_f$ hier $\boldsymbol{x}$. Die Jacobi-Matrix $\boldsymbol{J}$ ist die Ableitungsmatrix von $\boldsymbol{F}(\boldsymbol{c}_f)$ nach $\boldsymbol{c}_f$. Mit dem LM-Algorithmus wird eine optimale approximierte Lösung von $\boldsymbol{c}_f$ ausgerechnet.

Nach der Iteration erhält man die optimalen freien Parameter, damit werden die Trajektorie von $\boldsymbol{u}(t)$ und $\boldsymbol{x}_{sp}(t)$ als Form der kubischen Polynom dargestellt wie Gl. (3.2). Setzt man $\boldsymbol{u}(t)$ in den Zustandsgleichungen ein, wird auch ein $\boldsymbol{x}_{sim}(t)$ durch die Integration erstellt. Wenn die Abweichungen 1.) zwischen $\boldsymbol{x}_{sp}(t)$ und $\boldsymbol{x}_{sim}(t)$ und 2.) zwischen gegebenen Randwerte von $\boldsymbol{x}$ und $\boldsymbol{x}_0$, $\boldsymbol{x}_T$ unter den voraus gelieferten Beschränkungen nicht erfüllt werden, wird das System nochmal mit einer höheren Anzahl an Splineabschnitten gerechnet. Dieser Vorgang wird solange wiederholt, bis eine approximierte Lösung gefunden werden kann[5].

Als Schluss dieses Abschnitts wird ein Beispiel mit *PyTrajectory* gerechnet.

Beispiel 3.1 (Einfacher Doppelintegrator mit *PyTrajectory*)**.** *Betracht man einen auf der X-Achse laufenden Wagen mit zwei Zustandskomponenten: die Verschiebung x_1 und die Geschwindigkeit x_2. Eine aufgeprägte Kraft wirkt auf den Wagen ein. Die Beschleunigung gilt als Systemeingang u. Die Systemgleichung lautet:*

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u. \end{aligned} \tag{3.13}$$

Es sei gefordert, dass der Wagen sich von 0 *bis zu* 1m *bewegt und am Anfang und Ende in Ruhe ist. Mit anderen Worten ist der Anfangs- und Endwert von x_1 und x_2 jeweils* (0,1) *sowie* (0,0)*. Die feste Überführungszeit ist* 1 *Sekunde.*

Die Anfangswerte der Parameter von dem PyTrajectory *ist wie folgt eingestellt: die Anfangsschätzwerte der freien Parameter für die Polynome $\boldsymbol{x}$ und u als Form der Gl.* (3.2) *sind alle* 0.1*, die Anfangszahl der Splineabschnitte für beide $\boldsymbol{x}$ und u ist* 2*. Falls nötig muss die Anzahl in dem nächsten Iteration verdoppeln.*

Abb. 3.3 und Abb. 3.4 zeigt die Simulationsergebnisse des Systems. Das System benötigt zwei Iterationen (nämlich endlich 4 *Spline-Abschnitte), um eine optimale Lösung von den freien Parameter $\boldsymbol{c}_f$ zu finden. Unter der Wirkung eines ungefähr sinusförmigen Eingangs mit maximal* $6m/s^2$ *läuft der Wagen entlang dem geplanten Weg. Das heißt, der Wagen beschleunigt sich in die ersten Halbzeit* (0.5s)*, und nach der Erreichung der maximale Geschwindigkeit (circa* 1.8m/s*) bremst er bis zum Stoppen.*

[5]Dieser Iterationsvorgang wird unter "Hauptiteration" genannt. Anders als die Iteration innerhalb des LM-Verfahrens (als Nebeniteration genannt) ist der Parameterwert nach einer Hauptiteration schon die optimale Lösung von LM-Verfahren, der aber die erforderte minimale Abweichung zwischen dem Soll- und Istendwert des Zustände überschreitet. Ein erneuter Iteraionsvorgang mit vergrößerten Anzahl der Splineabschnitte ist deswegen nötig.

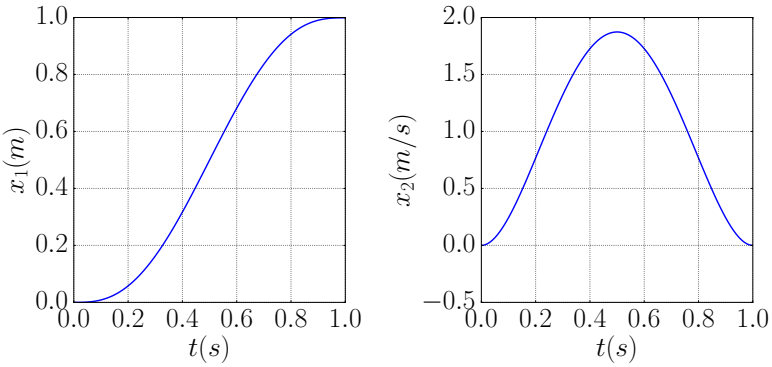


Abbildung 3.3 – Trajektorie der Systemzustandskomponente x_1 und x_2 (Position und Geschwindigkeit). Die Überführungszeit ist fest: $1s$.

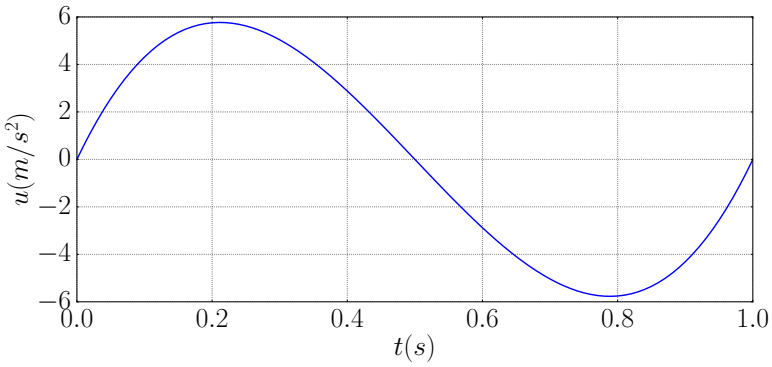


Abbildung 3.4 – Trajektorie des Systemeingangs u_1 (Wagenbeschleunigung). Die Überführungszeit ist fest: $1s$.

3.2 Erweiterung der Funktion von PyTrajectory

Vor beginn der Arbeit war es möglich, eine Trajektorie von $\boldsymbol{u}$ und $\boldsymbol{x}$ eines Systems mithilfe des Pakets *PyTrajectory* und der vom Nutzer voraus gelieferten Überführungszeit T zu finden. Dabei könnten aber folgende Probleme auftreten: Einerseits wären die Systeme mit diesem T nicht optimal (z.B. es bräuchte mehrmals Iterationen) , auf der anderen Seite kann bei manchen Systemen nur dann eine Trajektorie gefunden werden, wenn man zufällig die richtige Dauer gewählt hat. Deswegen liegt eine Aufgabe bei der Erweiterung der Funktion von PyTrajectory darin, eine geeignete Überführungszeit der Trajektorienplanung eines Systems ausrechnen zu lassen, anstatt sie vorgeben zu müssen.

3.2.1 Optimale Überführungszeit

Es wird angestrebt, dass die Überführungszeit T nicht vom Nutzer explizit vorgegeben wird, sondern anhand der Systemgleichungen mittels des Levenberg-Marquardt-Algorithmus automatisch ausgerechnet wird. Der Anfangspunkt ist die Ansetzung der Zeittransformation von $t = t$ zu $t = k\tau$, wobei k ein zusätzlicher freier Parameter (ohne Einheit) ist. Dann folgt die Transformation:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\text{ori}} &= \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}\left(\boldsymbol{x}, t\right) \\ \dot{\boldsymbol{x}}_{\text{new}} &= \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} \cdot \frac{\mathrm{d}t}{\mathrm{d}\tau} = k \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = k \cdot \boldsymbol{f}(\boldsymbol{x}, t) = \boldsymbol{f}_{\text{new}}\left(\boldsymbol{x}, t, k\right). \end{aligned} \tag{3.14}$$

Bei $k > 1$ braucht das System mehr Zeit, um die Endposition zu erreichen und bei $k < 1$ weniger.

Nach der Implementierung in Python werden zwei Beispielsysteme damit getestet, um die Gültigkeit und Brauchbarkeit zu überprüfen. Für jedes Beispiel wird die unter der Berücksichtigung von Zeit entworfene Trajektorie (abgekürzt als "Trajektorie mit k") mit dem zum originalen System geeigneten Trajaktorie (Kurz als "Trajektorie ohne k") verglichen.

Beispiel 3.2 (Doppelintegrator)**.** *Die Ergebnisse für das System "ohne k" wurde schon im letzten Abschnitt vorgestellt. Die erweiterte Systemgleichung von dem System "mit k" sind:*

$$\begin{aligned} \dot{x}_1 &= k \cdot x_2 \\ \dot{x}_2 &= k \cdot u. \end{aligned} \tag{3.15}$$

Als Anfangswerte bleiben alle Parameter außer der Überführungszeit T wie Bsp. 3.1 unverändert. T hängt von der freie Variable k mit dem Anfangsschätzwert $k_0 = 1.23$ ab.

Einstellung von Anfangswerte der freien Parameter $\boldsymbol{c}_f$ als 0.1: *Der Wagen bewegt noch von 0 bis zu 1m und am Anfang und Ende bleibt er in Ruhe. Die Anfangsschätzwerte von $\boldsymbol{c}_f$ sind noch 0.1, die Anzahl der Spline-Abschnitt am Beginn des Algorithmus und der entsprechende Multiplikator für die Iteration sind jeweils noch 2.*

Mehr als in der Randwertaufgabe-Lösung mit den originalen Systemgleichungen benötigt das erweitere System 4 Hauptiterationen[6] (also 16 Spline-Abschnitte), um eine optimale Lösung von $\boldsymbol{c}_f$ zu finden.

Aus Abb. 3.5 und Abb. 3.6 kann man erkennen, zwar die maximale Kraft und die Geschwindigkeit weit kleiner als die im originalen System sind, ist aber der Wert von k ist hier ziemlich groß (ungefähr $1662,53$)! Der Wert von k nimmt ab 1.23 in jeden

[6] Die Erklärung für Haupt- und Nebeniteration liest man 5.

Iteration zu. Der Wagen in diesem Beispiel muss sich deshalb 1662s *lang bewegen, bevor er in die Endposition ankommt. Das bedeutet, die Lösung von* k *ist zwar* ***richtig*** *aber* ***nicht brauchbar****.*

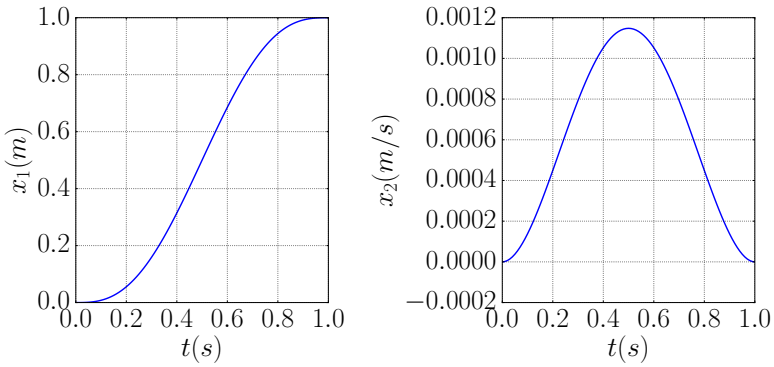


Abbildung 3.5 – Verlauf der Systemzustände vom System(3.15). x_1 und x_2 stehen für Position und Geschwindigkeit. Anfangswerte der freien Spline-Parameter sind alle 0.1. Die x-Achse steht nicht für die reale Überführungszeit, die tatsächlich $(1 \cdot k)$ Sekunde dauert.

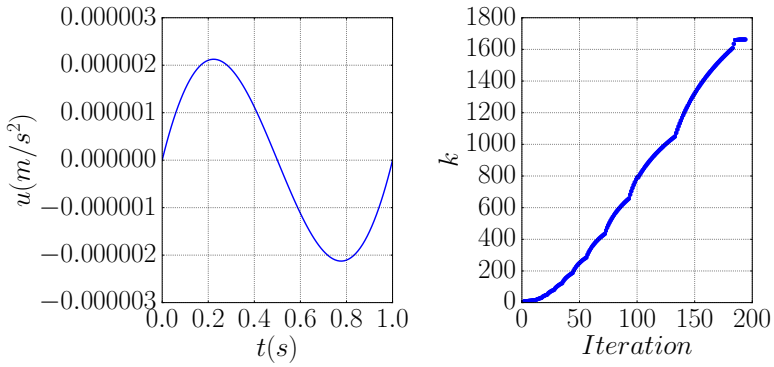


Abbildung 3.6 – Verlauf des Systemeingangs und die Veränderungskurve von k vom System (3.15). u_1 stehen für Wagenbeschleunigung. Der Wert von k steigt immer an. Anfangswerte der freien freien Parameter sind alle 0.1. Es gibt insgesamt ungefähr 2000 Nebeniterationen. Die "Iteraion" in der Abbildung von k ist tatsächlich Nebeniteration (im Folgenden ebenso).

Einstellung von Anfangswerte der freien Parameter c_f ***aus dem Simulationsergebnis der originalen Systemgleichungen:*** *Zur Verbesserung des Ergebnisses wird ein anderer Anfangswert für die Splineparameter gewählt. Statt aller Werte* 0.1 *sind die freien Parameter jetzt gleich den Rechenergebnissen der Splineparameter*

nach der ersten Iteration in dem System "ohne k". Die Kurven in Abb. 3.7 und 3.8 sehen besser als in Abb. 3.5 und Abb. 3.6 aus. In dem ganzen Iterationsverlauf steigt der Wert von k hauptsächlich mit 2*-mal Abstieg (bei der ersten und siebten Iteration). Im Vergleich zu dem Simulationsergebnis mit allen Anfangswerten* 0.1 *braucht diese nur zwölf Nebeniterationen. Der Wagen stoppt zum Zeitpunkt* 1.67*s.*

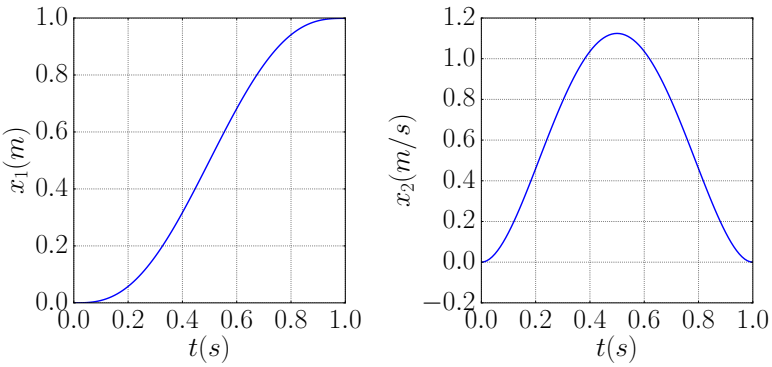


Abbildung 3.7 – Trajektorie der Systemzustandskomponente vom System (3.15). Anfangswerte von $\boldsymbol{c}_f$ sind aus den Ergebnissen von originalen Systemgleichungen (ohne k) gegeben.

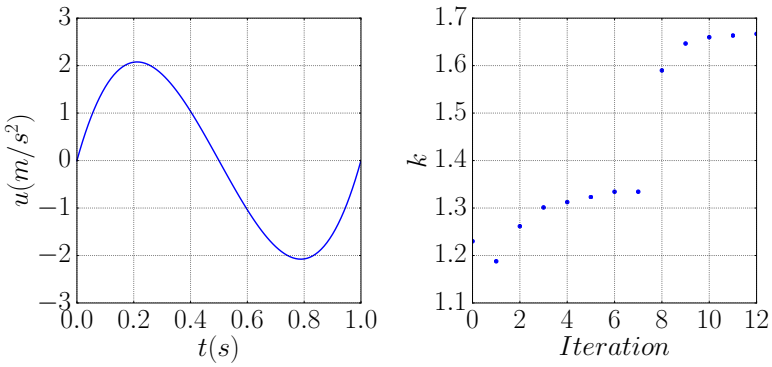


Abbildung 3.8 – Trajektorie des Systemeingangs vom System (3.15). Anfangswerte von $\boldsymbol{c}_f$ sind aus den Ergebnissen von originalen Systemgleichungen gegeben.

Dieses Beispiel zeigt eine starke Abhängigkeit der Konvergenz des Levenberg-Marquardt-Algorithmus von den Anfangsschätzwerten. Wenn die Anfangsschätzwerte der Parameter zu weit von der unbekannten Minimalstelle vorgegeben werden, kann die Lösung des Quadratmittelproblems eines Regelsystems mittels LM-Algorithmus nicht konvergieren oder erreicht nur einen lokalen, nicht global optimierten Punkt.

Das folgende Beispiel stellt die Anwendung der LM-Methode in einem verbreiterten nichtlinearen System vor.

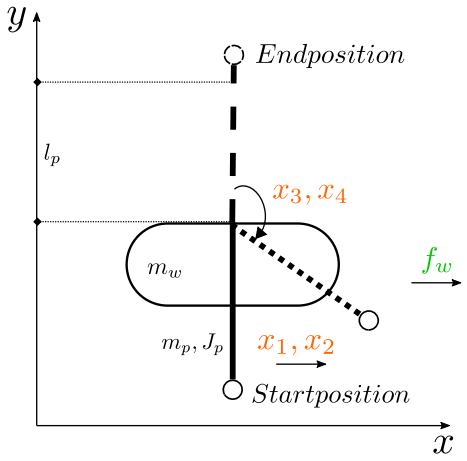


Abbildung 3.9 – Struktur des inversen Pendels: in dem System gibt es vier Zustände x_1 bis x_4. Das Pendel bewegt sich von 180° bis zur instabilen Ruhelage (nämlich vom ganz unten zu ganz oben). m_w und m_p sind jeweils die Masse des Wagens und des Pendels, J_p ist das Trägheitsmoment und l_p ist der Abstand zwischen dem Stützpunkt und dem Pendelschwerpunkt. Die Pfeile neben Zuständen geben die Vektorrichtung an. x und y sind Koordinatenachsen.

Beispiel 3.3 (Inverses-Pendel-System)**.** *Abb. 3.9 zeigt das Schema des inversen Pendels. Der Pendelarm ist auf einem horizontal bewegten Wagen montiert. Der Wagen bringt daher eine horizontale Kraft auf das Pendel auf. Beim Aufprägen der Kraft F auf den Wagen bewegt sich das Pendel von unten nach oben. Die instabile Ruhelage ist der Punkt, in dem das Pendel genau senkrecht zum Wagen steht und der Drehwinkel* 180° *beträgt. Systemvariable in diesem Modell sind die Wagenposition x_1, dessen Geschwindigkeit x_2, der Pendeldrehwinkel x_3, die Drehgeschwindigkeit davon x_4. Der einzige Systemeingang ist die auf den Wagen aufgeprägte Kraft $u = f_w$.*

Die dazugehörenden Differentialgleichungen des Modells sind wie folgt beschrieben [14]:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{m_p \sin(x_3)(-l_p x_4^2 + g\cos(x_3))}{m_w l_p + m_p \sin^2(x_3)} + \frac{\cos(x_3)}{m_w l_p + m_p l_p \sin^2(x_3)} u \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{\sin(x_3)(-m_p l_p x_4^2 \cos(x_3) + g(m_w + m_p))}{m_w l_p + m_p \sin^2(x_3)} + \frac{\cos(x_3)}{m_w l_p + m_p \sin^2(x_3)} u. \qquad (3.16)
\end{aligned}$$

Wenn die Wagenverschiebung x_1 als der Systemausgang gewählt wird, lässt sich die Systemzustandsdarstellung mit der Ein-Ausgang-Linearisierung einfacher beschreiben. Da der Systemausgang $y = x_1$ und $\dot{y} = \dot{x}_1 = x_2$ sowie $\ddot{y} = \dot{x}_2 = M_1(\boldsymbol{x}) + M_2(\boldsymbol{x}) \cdot u$ ist, ersetzt ein virtueller Eingang v mit $\dot{x}_2 = v$ den originale Systemeingang u in den

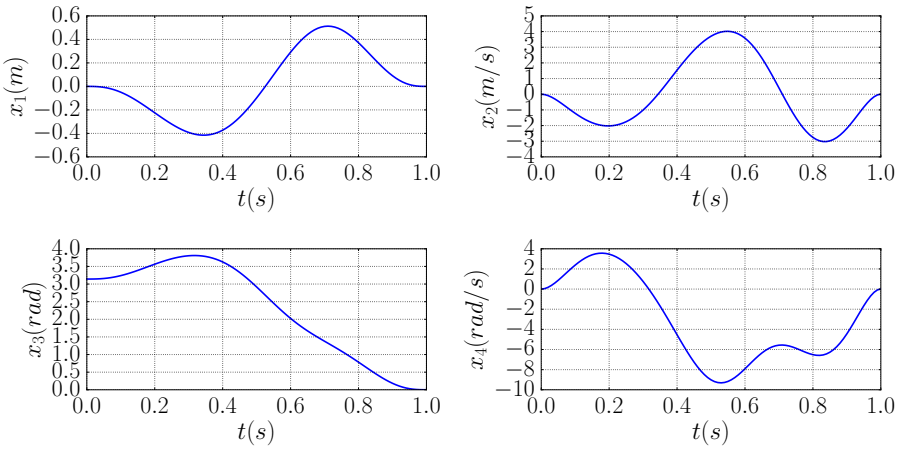


Abbildung 3.10 – Verlauf der Zustandsparameter des inversen Pendel System (ohne k) mit dem virtuellen Eingang.

Systemgleichungen und erhält man:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{g}{l_p}\sin(x_3) + \frac{1}{l_p}\cos(x_3)u. \end{aligned} \tag{3.17}$$

Die Verfahrensparameter sind wie folgt eingestellt: weil der Pendel sich von unten nach oben drehen und der Wagen endlich zum Startpunkt zurückgehen soll, sind die Randwerte von $\boldsymbol{x}$ wie $(0, 0, \pi, 0) \rightarrow (0, 0, 0, 0)$ erfordert. Die Anfangszahl der Spline-Abschnitte und das Iterationsvielfache sind jeweils 2*. Alle Anfangsschätzwerte der Polynomkoeffizienten sind* 0.1*. Der Anfangswert von k für das System mit der Wirkung von k ist noch* 1.23*. Die Ergebnisse sind in Abb. 3.10, Abb. 3.11, Abb. 3.12 und Abb. 3.13 dargestellt.*

Nach 5 Iterationen (also 32 Spline-Abschnitte) rechnen beide Systeme optimale Lösungen aus. Wie in Abb. 3.14 schwingt der Wert von k (nicht monotone Abbildung) und endet um 1.48*, was als plausibler Wert erscheint. Für das System ohne oder mit der Wirkung von k sind die Kurvenformen von Zustandsvariablen und dem virtuellen Eingang fast identisch. Aber Offensichtlich liefert die Trajektorie für das System mit k einen kleineren Bewegungsbereich für $\boldsymbol{x}$ und u (z.B. x_2 läuft jeweils $(-2$ bis $+4)m/s$ im System ohne Zeittransformation und $(-2$ bis $+2.5)m/s$ mit den geänderten Koordinaten). Deswegen wird weniger Eingangsenergie im letzteren System benötigt. In Bezug auf den relativ kleinen Unterschied bei der Überführungszeit (*1s *und* 1.48s*) ist die Trajektorieplanung für das System mit der Wirkung von k besser.*

Jetzt wird ein anderer Vorteil von System mit k diskutiert. Wie würde sich die Trajektorieplanung verändern, wenn auf die Systemzustände eine Beschränkung ausgeübt würde?

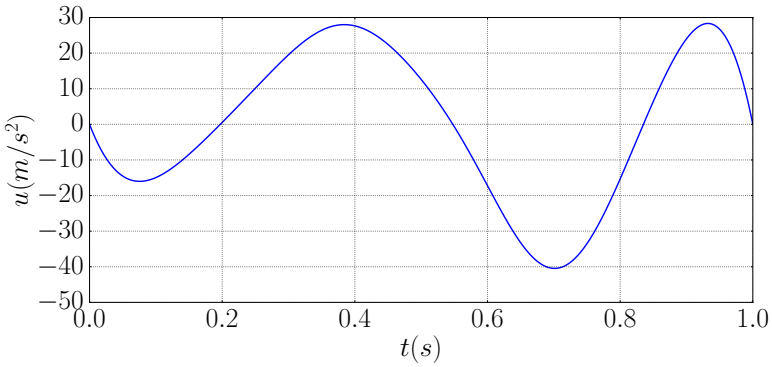


Abbildung 3.11 – Verlauf des virtuellen Eingangs des inversen Pendel System (ohne k). Die maximale Geschwindigkeit von u ist ungefähr $28m/s^2$.

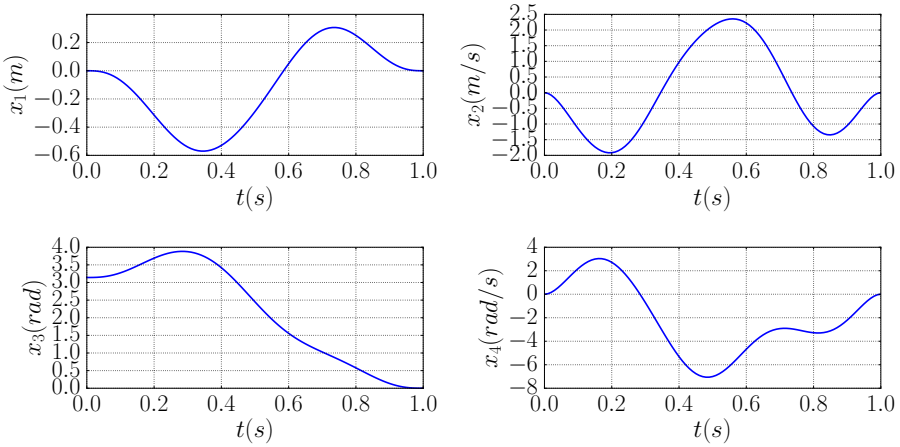


Abbildung 3.12 – Systemzustandskurven des inversen Pendel System (mit k) mit dem virtuellen Eingang.

Als ein Testzustand wählt man hier die Wagenverschiebung x_1. Aus Abb. 3.10 und 3.12 ist der Bewegungsbereich von x_1 jeweils $(-0.42$ bis $+0.5)m$ und $(-0.58$ bis $+0.3)m$, deswegen kann man eine Beschränkung für x_1 als einen zusätzlichen Bedingung des Systems einstellen: der Wagen bewegt nur im Bereich von $(-0.2$ bis $+0.4)m$. Nach acht Iterationen rechnet das System mit k eine Lösung: $k = 1.0535$ aus, dennoch kann das originale System bis zur achten Iteration kein Ergebnis erhalten.

Zusammengefasst aus oberen Beispielen stellt man eine Hypothese auf: Die Trajektorie kann für breitere/strengere Situationen/Bedingungen unter Berücksichtigung der Ausrechnung von einer geeigneten Überführungszeit eingeplant werden, sofern gute Anfangswerte der freien Splineparameter $\boldsymbol{c}_f$ vorgegeben werden. Die Richtigkeit dieser Idee wird im nächsten Abschnitt diskutiert.

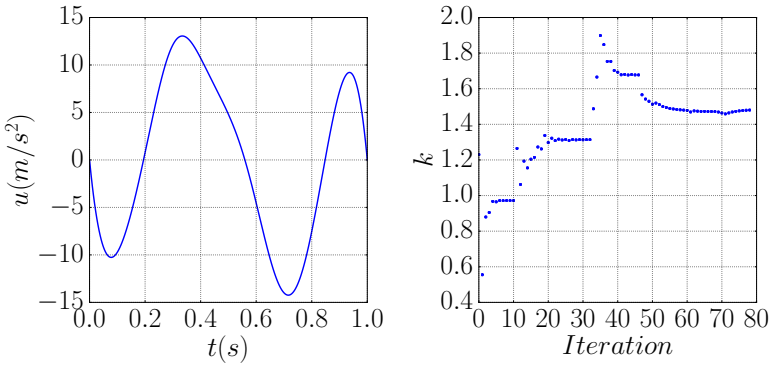


Abbildung 3.13 – Systemeingang des inversen-Pendels (mit k) mit dem virtuellen Eingang. Die maximale Beschleunigung trägt ca. $12m/s^2$ und nur Hälfte des Wertes in Abb. 3.11.

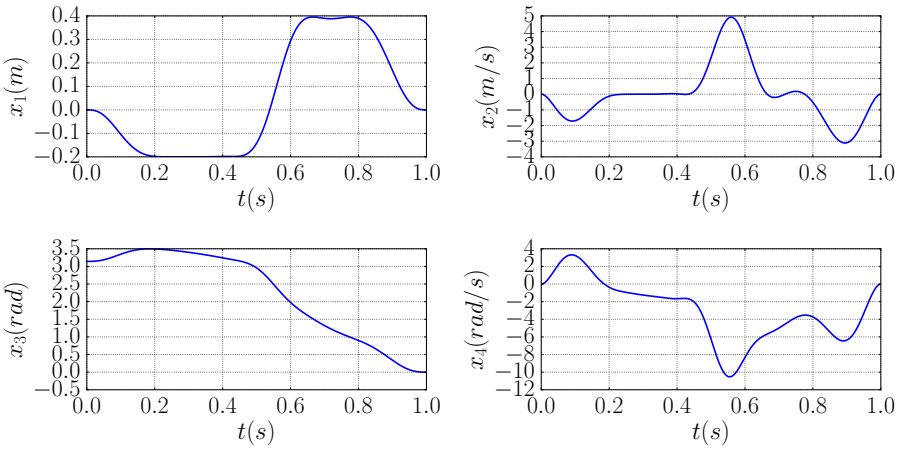


Abbildung 3.14 – Zustandsvariablenkurven des inversen-Pendels (mit k) mit dem virtuellen Eingang. x_1 ist zwischen $(-0.2, 0.4)$ eingeschränkt.

3.2.2 Einfluss der Startschätzung von c_f

Wie in Abb. 3.16 findet der Levenberg-Marquardt-Algorithmus zum Lösen des Quadratmittelproblems nur lokale Minimum auf. Mit verschiedenen Anfangsschätzwerten werden wahrscheinlich unterschiedlichen Ergebnisse von $\boldsymbol{c}_f$ ausgerechnet. Das Paket *PyTrajectory* stellt eine Möglichkeit zur Verfügung, $\boldsymbol{c}_f$ gemäß der Vorgabe eines Saat-Wertes (engl. seed) mit pseudo-zufälligen Werten im halboffenen Intervall $[0.0, 1.0)$ festzulegen.

Zum Überprüfen der Hypothese wählt man den Saat-Wert von 0 bis 99 für den Doppelintegrator, das inversen Pendel und einen zwei-Gelenke-Manipulator System (siehe Anhang A.1 für die Systemdarstellung). Hier berücksichtigt man außer der Situati-

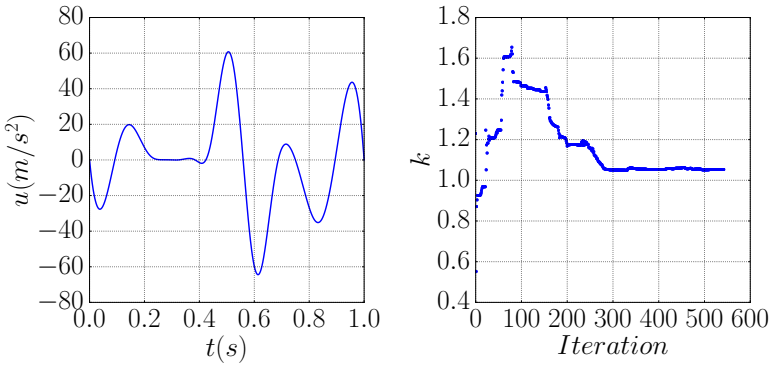


Abbildung 3.15 – Systemeingangskurve des inversen-Pendels (mit k) mit dem virtuellen Eingang. x_1 ist zwischen $(-0.2, 0.4)$ eingeschränkt.

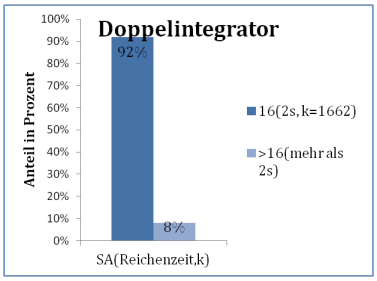
on in Beispiel 3.3 (Inverses-Pendel-System-A) noch eine Trajektorienplanung für das inversen-Pendel-System mit der Randwerte von $\boldsymbol{x}$: $\boldsymbol{x}_0 = (0, 0, 0, 0)$ und $\boldsymbol{x}_0 = (0, 0, \pi, 0)$ (Inverses-Pendel-System-B)[7]. Aus den vorher gezeigten Ergebnissen ist schon bekannt, dass bei allen Startschätzwerten gleich 0.1 gilt $k_{end} = 1662$ bei dem Doppelintegrator System und $k_{end} = 1.48$ bei dem inversen Pendel System, wenn sich das Pendel von unten nach oben dreht. Für beide Systeme ist die maximale Iterationsanzahl 6 (oder 64 Spline-Abschnitten.)

Abb. 3.16 stellt die statistischen Daten mit 100 Saat-Werten dar. Bei dem Doppelintegrator System ist es interessant, dass in 92% Fällen eine Losung von k ausgerechnet werden kann, die aber sehr ähnlich zueinander und zu den Ergebnissen bei $\boldsymbol{c}_f = \mathbf{0}.1$ ist (1660). In den restlichen 8 Fällen hat k bis sechsten Iteration auch einen ähnlichen Wert (-1660). Für das inverse-Pendel System mit dem von oben nach unten drehenden Pendel, egal welche Anfangswerte eingestellt werden, findet der Rechner keinen plausiblen Wert von k_{end}. Das ist gleich wie der Fall $\boldsymbol{c}_f = \mathbf{0.1}$. Aber für das Beispiel 3.3 sind die Lösungen von k mit allen Saat-Werten ungefähr identisch: $k \approx 0$. Da der Wert von k immer so klein ist, ist es anzunehmen, mit mehreren Iterationen auch keine Lösung gefunden wird. Bei dem Zwei-Gelenke-Manipulator funktioniert die Idee aber sehr gut. Deutlich hängt der Erfolg des Algorithmus zum Ausfinden eines brauchbaren k von der Anfangswerten ab.

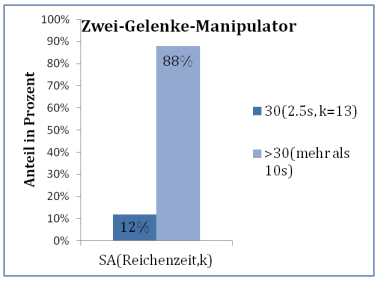
Aus dem Beispiel ist die Einstellung der Anfangswerte von freien Parameter mittels des Saat-Wertes für einige Regelsysteme geeignet. Aus der Sicht der Autorin liegt der Grund vermutlich in den Startwerte, die beliebig ausgewählt werden und noch weit von ihren tatsächlichen optimalen Werte liegen.

Die Vorgabe guter Startwerte für $\boldsymbol{c}_f$ ist also nicht einfach, deswegen werden einige andere Methoden zur Verbesserung der Lösung von k entworfen.

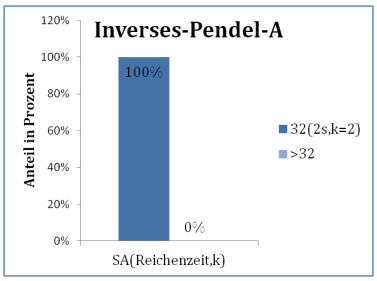
[7] Für dieses System kann kein geeignetes k mit den Startschätzwerte $\mathbf{0.1}$ ausgegeben werden.



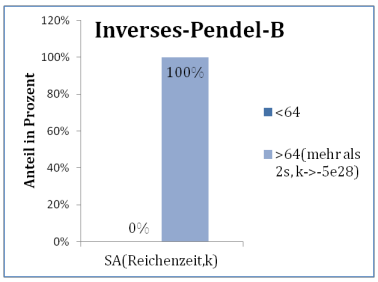
a) Die benötigten Spline-Abschnitten für den Doppelintegrator



b) Die benötigten Spline-Abschnitten für den zwei-Gelenke-Manipulator



c) Die benötigten Spline-Abschnitten für das Inverses-Pendel-System-A.



d) Die benötigten Spline-Abschnitten für Inverses-Pendel-System-B.

Abbildung 3.16 – Vergleich des Spline-Abschnittsanzahls mit unterschiedlichen Startschätzwerte von $\boldsymbol{c}_f$. SA ist die Abkürzung für Spline-Abschnittszahl. Das erste Element in der Klammer steht für die echte Rechenzeit des Rechners. Das zweite ist der Modalwert von k_{end}.

Zuerst versucht man den Wert von k, in einem gegebenen zu beschränken.

3.2.3 Beschränken des Wertbereichs von k

Die prinzipielle Idee kommt aus der Beschränkung von Systemzustände, die in *PyTrajectory* schon realisiert wird. Der Ausgangspunkt liegt in der Transformation der Darstellung von der freien Variable k von der originalen Systemkoordinaten mittels einer monotonen steigenden Sättigungsfunktion ψ in einen neuen Koordinaten, darin keine Begrenzung von dem neuen k besitzt. k in der neuen Koordinate wird wie "κ" genannt und wie folgt geschrieben:

$$k^- \leq k = \psi(\kappa, k^{\pm}) = k^+ - \frac{k^+ - k^-}{1 + e^{m \cdot \kappa}} \leq k^+, \quad m := \frac{4}{k^+ - k^-}. \tag{3.18}$$

Gl. (3.18) bedeutet, dass k in k^+ und k^- beschränkt wird. Die Ableitung von ψ nach κ:

$$\frac{\mathrm{d}\psi}{\mathrm{d}\kappa} = \frac{m(k^+ - k^-)e^{m \cdot sk}}{(1+e^{m\cdot\kappa})^2} = \frac{4 \cdot e^{m\cdot\kappa}}{(1+e^{m\cdot\kappa})^2} \tag{3.19}$$

ist immer positiv. k wird durch κ in den Systemfunktionen ersetzt und der Wert von κ wird mittels des Levenberg-Marquadt-Algorithmus ausgerechnet. Nach der Erhaltung von κ weißt man dann die Größe von k. Die Kurve von ψ ist wie Abb. 3.17 dargestellt. Siehe [6] und [14] für weitere Information. Zur Überprüfung dieser Idee stellt man

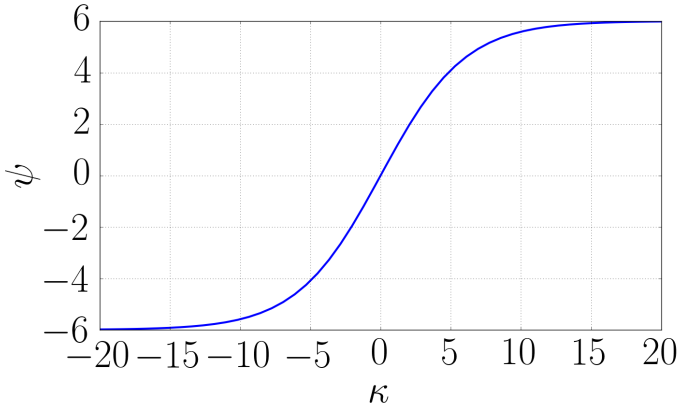


Abbildung 3.17 – Die Kurve der Sättigungsfunktion ψ mit $k^+ = 6.0$ und $k^- = -6.0$.

zuerst $k^+ = 5$ und $k^- = 0$ in dem Doppelintegrator System (Beispiel 3.2) ein. Der Anfangsschätzwert von k ist noch 1.23. Das damit berechnete Ergebnis wird in Abb. 3.18 gezeigt.

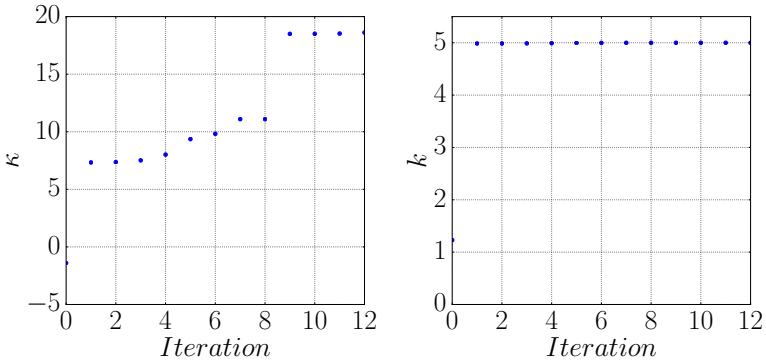


Abbildung 3.18 – Die Entwicklung von κ in der transformierten Koordinaten und von k in der originalen Koordinaten für das Doppelintegrator System. Der Wert von κ ist nicht begrenzt während k hier innerhalb von $(0-5)$ bleiben muss.

Dieses System benötigt 2 Iterationen, eine Lösung zu finden. Schließlich ist $k = 4.999$ und nähert der Obergrenze. Bei der zweiten Iteration ist κ schon bis ungefähr 8 und

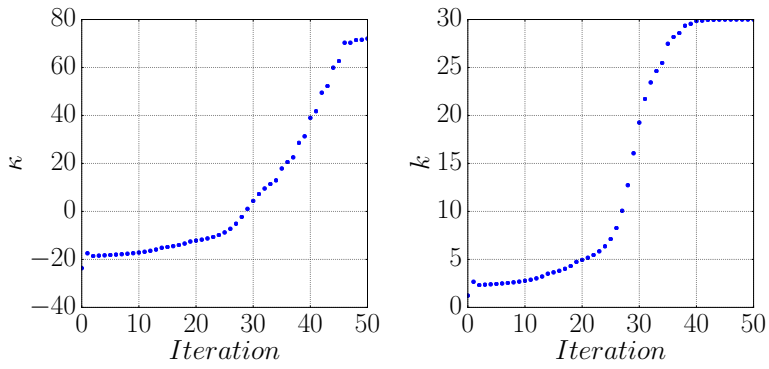


Abbildung 3.19 – Die Entwicklung von κ und k für Doppelintegrator System. k muss jetzt innerhalb von $(0 - 30)$ bleiben.

führt zu einem sehr großen Wert von $e^{m \cdot \kappa}$ mit $m = 0.8$ in diesem Fall. Setzt man es in Gl. (3.18) ein, dann versucht k sich zu der oberen Grenze k^+ anzunähern. In einen anderen Simulation mit der Vorgabe $k^+ = 30$ und $k^- = 0$ vergrößert sich der Wert von k zu 29.9999 (κ auch immer größer), wie Abb. 3.19 zeigt. Der Grund für das vergrößerten κ liegt in dem positiven Iterationsschritt h (siehe Gl. (3.11)) in jeder Iteration.

Das Ergebnis zeigt sich ganz anders beim Beispiel 3.3 (Inverses-Pendel-System). Die Vorgabe vom Bereich für k ist zwischen 0.1 und 10. Der Anfangswert k_0 ist 1.23 und am Ende verändert sich k zu 1.48. Der Wert ist ganz identisch wie das Ergebnis ohne Beschränkung von k (siehe Abb. 3.13). Simulierte Kurven in Abb. 3.20 zeigen, k vergrößert sich in manchen Iterationen und verkleinert sich in anderen Iterationen.

Aus den Beispielen erkennt man, eine Vorgabe des Bereichs von k verbessert das Ergebnis nicht. In den Fällen, worin ein sinnvoller k ohne der Begrenzung schon ausgerechnet werden kann, hat die Begrenzung keinen Einfluss/Wirkung. In anderen Fällen mit sehr groß oder klein k_{end}, vergrößert/verkleinert der Wert in der transformierten Koordinaten immer bis zur oberen/unteren Schränke.

3.2.4 Straffunktion von k

Eine weitere Methode zur Begrenzung des Bereichs von k liegt darin, eine Straffunktion $Pe(k)$ als die letzte Zeile der Systemzustandsfunktion hinzuzufügen, nämlich $\boldsymbol{f}_{pe} = (\boldsymbol{f}^T, Pe)^T$. Prinzipielle Anforderung davon ist es, wenn k die gegebene Beschränkung überschreitet, hat die Straffunktion Pe in $\boldsymbol{f}_{pe}$ einen Wert viel größer als 0. Anderenfalls bleibt Pe ungefähr Null und hat keinen Einfluss auf die originale Systemfunktion $\boldsymbol{f}$. (Dann wird $k_{end} \in [k_{min}, k_{max}]$ unter Berücksichtigung der Levenberg-Marquardt-Methode bei $\boldsymbol{f}_{pe} \to min$ also $Pe(k) \to min$ erfüllt.)

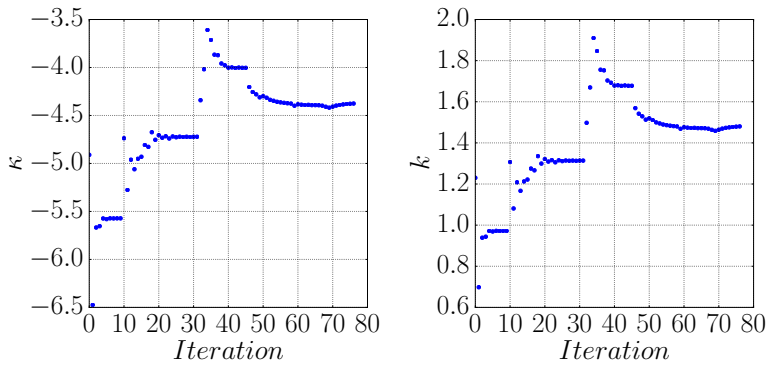


Abbildung 3.20 – Die Entwicklung von κ und k für das inverse-Pendel System. k muss innerhalb von $(0-10)$ bleiben.

Hier entwirft man die Straffunktion ähnlich wie eine Parabel. Wenn die Variable der von den zwei Beschränkungsparametern (k_{min} und k_{max}) definierten Definitionsmenge enthält wird, liegt die Zielmenge in der Nähe von 0. Dagegen gilt der Bildwert außerhalb dieses Bereichs ähnlich wie $(k-k_{mid})^2$, wobei k_{mid} der Mittelwert von k_{max} und k_{min} ist. Die konkrete Form von *Pe-Funktion* ist:

$$Pe(k, k_{min}, k_{max}) = \frac{(k-k_{mid})^2}{1+e^{5\cdot(k-k_{min})}} + \frac{(k-k_{mid})^2}{1+e^{5\cdot(k_{max}-k)}}. \tag{3.20}$$

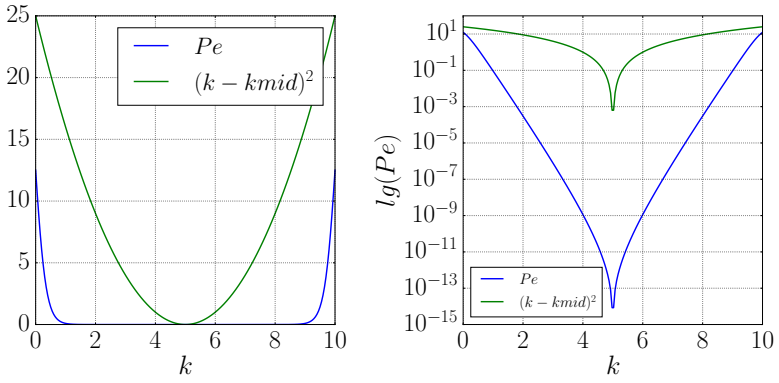
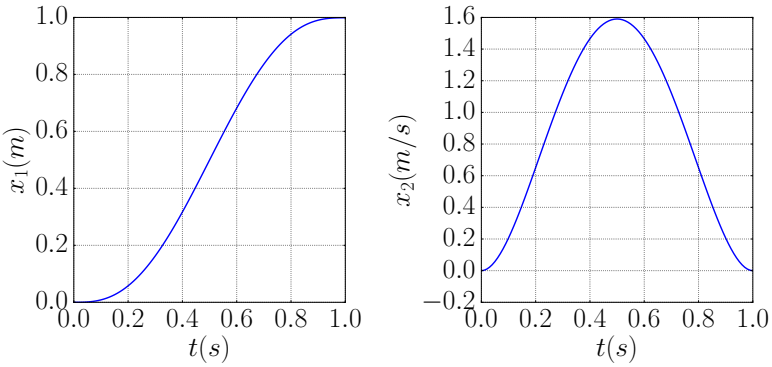


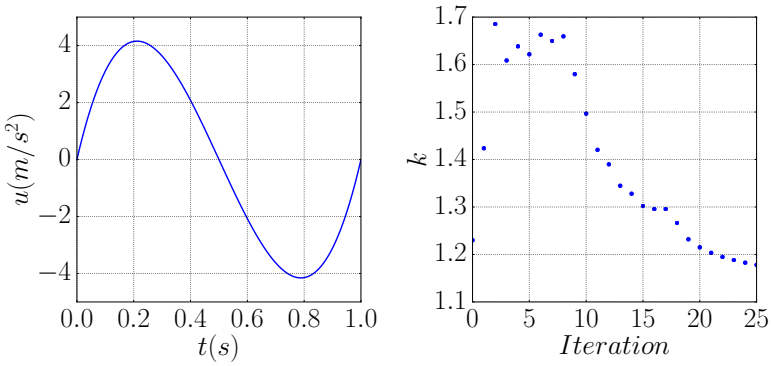
Abbildung 3.21 – Straffunktion von k: $k_{min}=0$, $k_{max}=10$, $k_{mid}=5$

Die Kurve von Pe mit $k_{min}=0$ und $k_{max}=10$ ist wie Abb. 3.21 gezeigt. Zwischen $(0,10)$ ist die Größe von Pe ungefähr 0. Außerhalb dieses Bereichs läuft Pe wie eine Parabel. Wenn beispielsweise $k_n=12$ bei n-ten Iteration ist Pe ungleich 0, danach geht der Wert von k_{n+1} im nächsten Iterationsschritt mittels der LM-Methode entlang der Richtung, worin k immer kleiner ist. Schließlich versucht das System unter Berücksichtigung der Systemzustandsfunktion mittels dieser Straffunktion eine Lösung in der Nähe von k_{mid} zu finden (Denn bei $k=k_{mid}$ ist der Wert von Gl. (3.20) gerade 0).

Beispiel 3.4. *Es wird wieder das Beispiel 3.2 (Doppelintegrator) betrachtet. Angenommen, dass k_{min} und k_{max} jeweils* 0.1 *und* 2 *beträgt. Die andere Bedingungen bleiben wie zuvor (der Anfangsschätzwert von k ist noch* 1.23 *und von $\boldsymbol{c}_f$ noch **0.1**). Die Ergebnisse sind in Abb. 3.22 dargestellt.*



a) Die berechnete Ergebnisse von $\boldsymbol{x}$.



b) Die berechnete Ergebnisse von u und k.

Abbildung 3.22 – Die Trajektorien von Systemzustandsvariablen und Systemeingang des Doppelintegrator-Systems.

Nach 2 *Hauptiterationen erhält man den noch sinusförmigen Zeitverlauf vom Eingang u aber mit dem maximalen Wert $4m/s^2$. In Anbetracht auf Abb. 3.3 ist der maximale Wert von der Wagengeschwindigkeit x_2 ähnlich wie das Anfangssystem ohne k: ca. $1.6m/s$. Der Wert von k steigt zuerst von* 1.23 *zu* 1.68 *auf, welche zu der oberen Grenze* 2.0 *approximiert. Danach sinkt k in dem nächsten Schritt ab, und dann Schritt für Schritt erreicht es endlich den optimalen Wert* 1.178.

Wenn k_{max} in Gl. (3.20) als 10 *eingestellt wird, ist k zum Ende ungefähr* 8. *Und bei einem größeren k_{max} (z.B.* 20*) wird auch ein größer k_{end} gefunden (z.B.* 17.5*). Mit anderen Worten ist je größer k_{max} eingesetzt, desto größer ist k_{end}. Grund dafür ist einfach: wie gesagt probiert der Levenberg-Marquardt-Algorithmus ein optimales k in*

der Nähe von k_{mid} zu finden. Dieses Beispiel zeigt darin, mit der Straffunktion kann eine optimale Überführungszeit ***in einem Bereich*** *gefunden werden. Ein wichtiger Vorteil der Straffunktion liegt darin, wenn die Trajektorieplanung in einem System mit einer festen Überführungszeit unmöglich entworfen zu können, kann man einen relativen großen Randwert von k_{min} und k_{max} einstellen, mit dem rechnet der Rechner einen Wert von k aus*[8].

Fazit: In diesem Kapitel wurde erstens die in dem *PyTrajectory* Paket bedingten Grundlage und Algorithmen vorgestellt. Zum Lösen des Quadratmittelproblems wird das Konzept des besonders wichtig Levenberg-Marquardt-Algorithmus eingeführt. Im Abschnitt 3.2 wurde das Python-Paket mit der Bestimmung der optimalen Überführungszeit durch die Koordinatentransformation von Zeit erweitert. Manche Probleme erschienen bei der Transformation, deswegen wurde eine Straffunktion dafür entworfen.

[8] Ob es erfolgreich ist, eine k_{end} zu finden, hängt noch von dem Startschätzwert von k ab.

Kapitel 4

Reglerentwurf durch Trajektorieplanung

In diesem Kapitel werden einige Regler für die sogenannten unteraktuierten Systeme entworfen, welchedie Brockett-Bedingung nicht erfüllen. Im Abschnitt 4.1 werden die Definition und der Charakter vom Unteraktuierten mechanischen System ausgegeben. Der Entwurf von Steuergesetze mit und ohne die Trajektorieplanung aus *PyTrajectory* werden separat in Abschnitt 4.2 vorgestellt.

4.1 Unteraktuierte mechanische Systeme

Unter einem unteraktuierten mechanischen System (UMS) versteht man ein spezielles System mit weniger unabhängigen Aktuatoren als der Anzahl von Freiheitsgrade. Im Vergleich mit einem vollständig aktuierten System kann ein UMS mittels weniger Eingängen ggf. gleiche Aufgabe lösen. Aufgrund der Dynamik des Systems oder zur Reduzierung der Kosten sind UMS in vielen Bereichen verbreitet.

Für die Bestimmung von Bewegungsabläufen eines Körpers relativ zu einem Inertialsystem nennt man die konstruktiv geometrischen oder mechanischen Einschränkungen der Pose oder Geschwindigkeit an den Körper *Zwangsbedingungen*.

Eine holonome Zwangsbedingung beschränkt die Position oder Geschwindigkeit des Körpers, die durch algebraische Gleichungen oder *integrierbare* Differenzialgleichungen dargestellt wird. Im Gegenteil dazu enthält eine nichtholonome Zwangsbedingung nicht integrierbare Differenzialgleichungen. Mit anderen Worten verfügt eine holonome Zwangsbedingung über die Form wie: $f(x, y, z, t) = 0$ und eine nichtholonome über die Form wie: $g(x, y, z, \dot{x}, \dot{y}, \dot{z}, t) = 0$ (die Ableitung kann nicht in der Form von f integriert werden).

Wenn die Systemgleichungen die Form $h(x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}, t) = 0$ besitzen, werden sie je nach Integrierbarkeit als *holonome* oder *nichtholonome Zwangsbedingung zweiter*

Ordnung [18] bezeichnet. Wenn h in die Form $g(x, y, z, \dot{x}, \dot{y}, \dot{z}, t) = 0$ integriert werden kann, heißt h eine *partiell integrierbare* Zwangsbedingung. Wenn h zwei mal mit der Schlussform $f(x, y, z, t) = 0$ integriert werden kann, heißt das *vollständig integrierbare* System holonomes System.

Der Freiheitsgrad eines Systems mit n_r redundanten Koordinaten und n_{ZB} holonomen Zwangsbedingungen ist $n = n_r - n_{ZB}$. Die Minimalkoordinate $\boldsymbol{q} = (q_1, q_2, ..., q_n)^T$ beschreibt dann die Konfiguration des Systems.

Zur Beschreibung der kinetischen Charakteristik nimmt man mit der EULER-LAGRANGE-Gleichungen nach [10]:

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}) = T^*(\boldsymbol{q}, \dot{\boldsymbol{q}}) - V(\boldsymbol{q}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \mathbf{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - V(\boldsymbol{q}) \qquad \text{(LAGRANGE-Funktion)} \tag{4.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{q}_i} - \frac{\partial L(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial q_i} = f_i, \qquad i = 1, \ldots, N. \qquad \text{(EULER-LAGRANGE-Gln.)} \tag{4.2}$$

an. Gl. (4.1) gibt die Darstellung von der kinetischen Koenergie T^* und der potenziellen Energie V. In Gl. (4.2) steht f_i für die bezüglich der Koordinate q_i eingeprägten und externen Kräften. Die Massen-/Trägheitsmatrix $\mathbf{M}$ ist symmetrisch und positiv definit.

Durch die Umformung von Gl. (4.2) in Vektorform erhält man [17]:

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \mathbf{G}(\boldsymbol{q}) = \boldsymbol{F}(\boldsymbol{q}) \tag{4.3}$$

wobei Matrix $\mathbf{C}$ die Wirkung von Zentrifugal- bzw. Corioliskraft und Matrix $\mathbf{G}$ konservative Kraft wie Gravitation darstellt.

Falls $rank(\boldsymbol{F}) = m \stackrel{!}{=} n$ heißt das System *vollständig aktuiertes* System. Wenn $m < n$ wird das System unter *unteraktuiertes* System genannt. Die Koordinaten $\boldsymbol{q}$ in einem UMS können in die vom Eingang $\boldsymbol{F}$ beeinflussten $\boldsymbol{q}_a$ und nicht beeinflussten $\boldsymbol{q}_u$ Koordinaten aufgeteilt werden: $\boldsymbol{q} = [\boldsymbol{q}_a, \boldsymbol{q}_u]^T$ mit $\boldsymbol{q}_a = (q_1, \cdots, q_m)^T$ und $\boldsymbol{q}_u = (q_{m+1}, \cdots, q_n)^T$. Ausgehend davon kann Gl. (4.3) in die folgende Form aufteilen:

$$\begin{bmatrix}\mathbf{M}_a \\ \mathbf{M}_u\end{bmatrix}\begin{bmatrix}\ddot{\boldsymbol{q}}_a \\ \ddot{\boldsymbol{q}}_u\end{bmatrix} + \begin{bmatrix}\mathbf{C}_a \\ \mathbf{C}_u\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{q}}_a \\ \dot{\boldsymbol{q}}_u\end{bmatrix} + \begin{bmatrix}\mathbf{G}_a \\ \mathbf{G}_u\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}_a \\ \mathbf{0}\end{bmatrix}. \tag{4.4}$$

Die erste Zeile in Gl. (4.3) bezeichnet man als aktuiertes Subsystem und die zweite Zeile unaktuiertes Subsystem, die auch als eine Zwangsbedingung zweiter Ordnung betrachtet werden kann. Gemäß [18] gehört die zweite Zeile zur nichtholonomen Zwangsbedingungen zweiter Ordnung. Nach [23] kann man bei der Linearisierung von System mit Systemfunktionen wie Gl. (4.4) $\ddot{\boldsymbol{q}}_a$ als den Eingangsvektor $\boldsymbol{v}$ wählen, damit sich der originale Systemeingang $\boldsymbol{u}$ als die Form von Systemzustandsvektor und $\boldsymbol{v}$ darstellen lässt. Aus diesem Grund wird die Wagenverschiebung x_1 in Beispiel 3.3 als der virtuelle Systemausgang ausgewählt.

4.2 Reglerentwurf mittels *PyTrajectory*

Mit dem Paket *PyTrajectory* kann eine Trajektorie durch Lösung einer Randwertaufgabe geplant werden. Die Idee ist es, durch die Analyse der entworfenen Trajektorien für $\boldsymbol{u}_t$ in der Umgebung einer Ruhelage ein Steuergesetz zu konstruieren und zwar für Systeme, die die Brockett-Bedingung nicht erfüllen.

Man kann die Umgebung M als eine Scheibe/ein Quadrat oder eine Kugel/einen Würfel betrachten, wenn die Anzahl der Systemzustände 2 oder 3 ist. Einige typischen Punkte werden daraus als die Anfangspunkte der Trajektorien ausgewählt. Die Trajektorieplanung wird durch *PyTrajectory* durchgeführt. Aus der Regelmäßigkeit der Trajektorien kann man dann versuchen, ein Steuergesetz zu finden.

Das erste Modell ist der nicht-holonome Doppelintegrator von Brockett.

4.2.1 Ausgang: Brocketts nicht holonomen Doppelintegrator

Das berühmte unteraktuierte Beispiel von Brockett wurde schon im Abschnitt 2.2, Beispiel 2.1 vorgestellt:

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 u_1 - x_1 u_2. \end{aligned} \tag{4.5}$$

Der Freiheitsgrad dieses Systems ist $n = 3$ während es nur $m = 2$ Eingänge gibt. Damit ist es ein unteraktuiertes System. Wegen $\operatorname{rang}(\boldsymbol{x}) = 3$ wählt man einen Würfel mit der Länge der Seite 0.2 als die Umgebung von $\mathbf{0}$ (siehe Abb. 4.1). Die Ruhelage liegt gerade um den Mittelpunkt und die typischen Anfangspunkte (oder als *Testpunkte* genannt) bestehen aus den acht Ecken, zwölf Mittelpunkten der Kanten, sechs Mittelpunkten der Begrenzungsflächen.

Für dieses Beispiel werden einige Methoden zum Entwurf der Trajektorien von $\boldsymbol{u}$ und dann $\boldsymbol{x}$ angewendet. Die erste Methode ist noch der Kollokationsverfahren mittels *PyTrajectory*.

Trajektorienplanung mit *PyTrajectory*

Die Einstellung von *PyTrajectory* ist wie folgt: Anfangswerte von freien Parameter $\boldsymbol{c}_f$ für Spline-Abschnitte sind 0.1, der Anfangsschätzwert von k ist 1.0, der Anfangszahl und das Vielfache für den Spline-Abschnitt in der nächsten Iteration ist jeweils 2.

Mit *PyTrajectory* wird eine Trajektorie von $\boldsymbol{u}(t)$ für jeden Ausgangspunkt geplant. Verschiedene Testpunkte benötigen verschiedene Iteration-Mal zwischen 1 und 3. Da

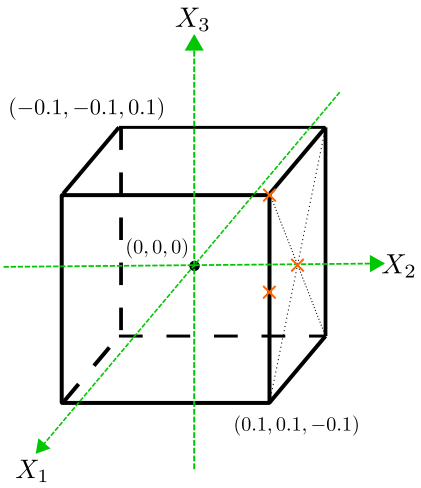


Abbildung 4.1 – Skizze des Würfels als die Umgebung von der Ruhelage. Die orange Kreuze sind die ausgewählten Anfangspunkte in einer Quadrate.

die Randwerte der Straffunktion sind $k_{min} = 0.1$ und $k_{max} = 5.0$ gewählt, schwingt der Wert von k_{end} bei unterschiedliche Testpunkte um 2.5. Abb. 4.2 und 4.3 zeigen die Simulationskurven der Eingangstrajektorien, die aber keine deutliche Regelmäßigkeit besitzen[1][2]. Die Tabelle in dem Anhang A.1 beschreibt die Details der Trajektorienkurven. Die Form der Kurven hängt nicht direkt von den Ausgangspunkten ab. In diesem Beispiel werden die Startwerte der Systemzustände gleichzeitig verändert, was die Findung einer Regelmäßigkeit erschwert. Deshalb wird als nächster Schritt die Regelmäßigkeit der Kurven bei der Veränderung nur eines Zustands untersucht.

Testpunkte von x_1 verändert von -0.1 bis zu -0.01 Zuerst wählt man, die Anfangspunkte von x_1 von -0.1 bis zu -0.01, mit der Schrittweite $\Delta x_1 = 0.01$. Die Anfangswerte von x_2 und x_3 bleiben in den 10 Situationen immer 0.1.

Jeder Fall braucht 2 oder 3 Iterationen mit k_{end} zwischen $2.0-2.7$. Wie Abb. 4.5 darstellt, geht die Amplitude der "minus Sinuskurven" von u_1 mit der Steigerung von $x_{1,0}$ im Bereich von $(-0.1, -0.03)$ (außer des Falls bei $x_{1,0} = -0.08$) immer herunter, während sich die Amplitude der sinusförmigen Kurve von u_2 immer vergrößert. Das heißt, in diesem Bereich ergeben sich beide Systemeingänge eine ähnliche Trajektorienform. Über -0.02 verkleinert die Amplitude der Kurven sehr schnell zu ungefähr -1. (Die Kurven von -0.02 und -0.01 liegen so nahe, dass nur eine Linie im Bild zu sehen ist. Dieser Effekt tritt z.B. auch bei x_3 in Abb. 4.4 in den Situationen $(-0.08, -0.02, -0.01)$.

Die Zustandverläufe verfügen bei der separaten Veränderung von $x_{2,0}$ und $x_{3,0}$ jeweils in einem kleinen Bereich über auch eine Regelmäßigkeit. Aber wenn die Anfangswerte zwei der drei Systemzustandsvariable gleichzeitig verändert werden, ergeben die Trajektorien

[1] Regelmäßigkeit hier bedeutet, dass die Kurven ca. die gleiche Form besitzen.
[2] Zum besser Lesen werden nur fünf Kurven angezeigt. Die gesamten Kurven sind in Abb. A.5 und A.6 dargestellt.

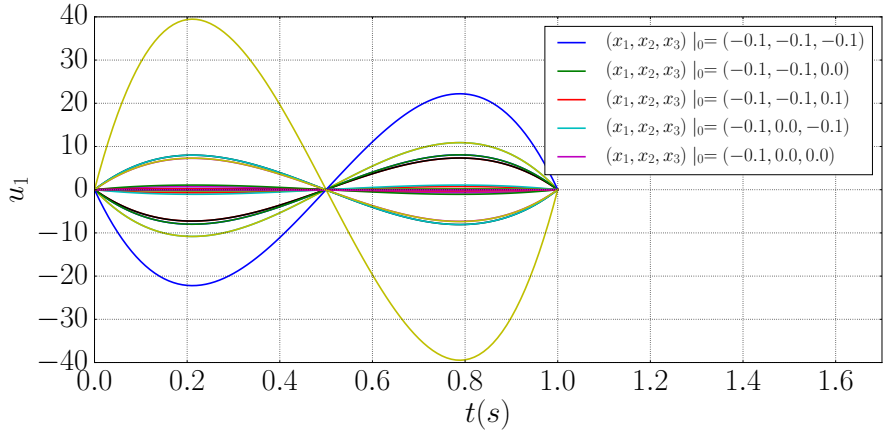


Abbildung 4.2 – Eingangsverläufe u_1 mit verschiedenen $\boldsymbol{x}_0$ für Brocketts nicht-holonomen Doppelintegrator mittels *PyTrajectory*.

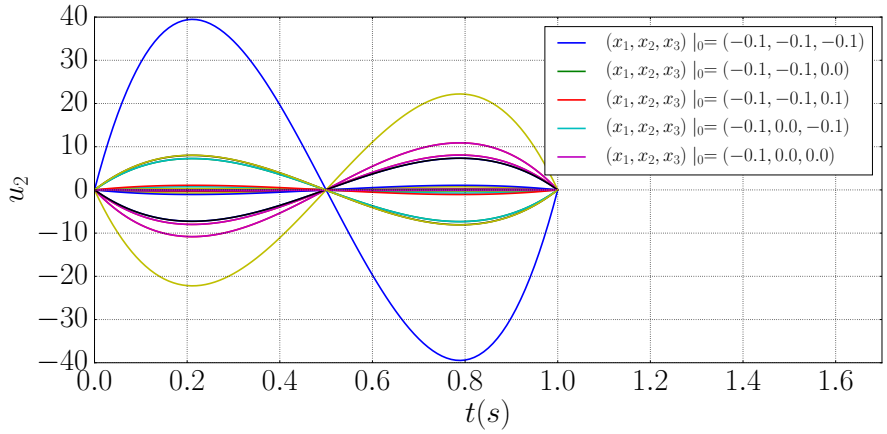


Abbildung 4.3 – Eingangsverläufe u_2 mit verschiedenen $\boldsymbol{x}_0$ für den Brocketts nicht-holonomen Doppelintegrator mittels *PyTrajectory*.

keine bestimmte Änderungsregelmäßigkeit.

Die Trajektorien zeigen eine Regelmäßigkeit in einem kleinen Bereich in dem Brocketts nicht-holonomen System, das die Brockett 2. Bedingung nicht erfüllt. Ob diese Regelmäßigkeit auch für das inversen-Pendel-System geeignet ist, lässt sich im nächsten Abschnitt untersuchen.

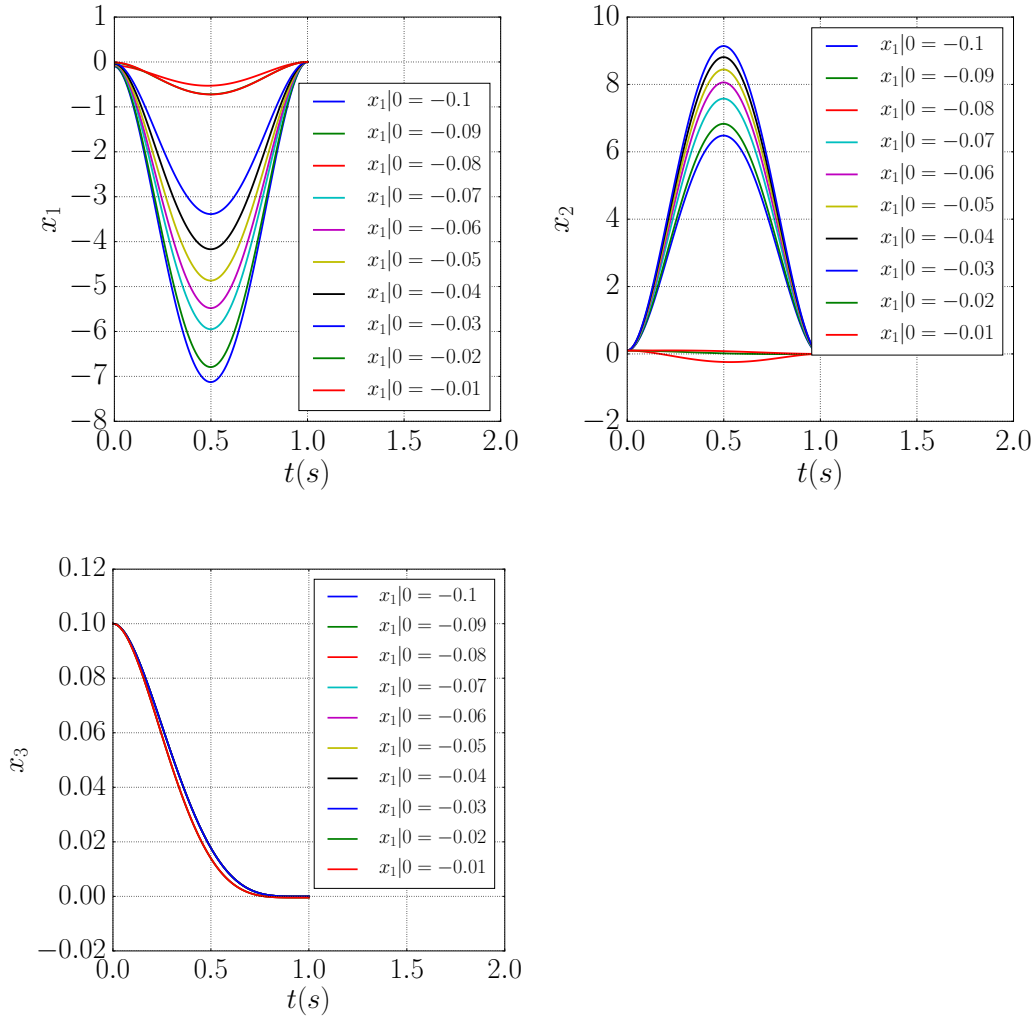


Abbildung 4.4 – Zustands- und Eingangsverläufe des Doppelintegrator-Systems.

4.2.2 Eingangsverlauf des inverse-Pendel-Systems

Die Zustandsfunktion des Benchmark-Systems erfüllt wegen des surjektiven Bildwerts die Brockett 2.Bedingung. Erinnert man die Gl. (3.17) und lässt sich der Wert von x_1 von -1.0 mit der Schrittweite $h = 0.1$ bei jedem Fall zu -0.4 vergrößern.

Mit dem Ergebnis in Abb. 4.6 zeigt die Kurven eine ähnliche Regelmäßigkeit wie zuvor: mit Anfangswerte von x_1 in einem begrenzten Raum $(-1.0, -0.5)$ kann man den Linientrend schätzen, außerhalb davon verändert die Kurve mit eine ganz verschiedene Form plötzlich.

In anderen Simulationen, wo $x_{2,0}$ zwischen $(0.2, 2.9)$ verändert und $x_{1,0} = x_{4,0} = 0, x_{3,0} = \pi$ ist, besitzen alle Trajektorien auch ähnliche Form. Aber die Kombination der obere

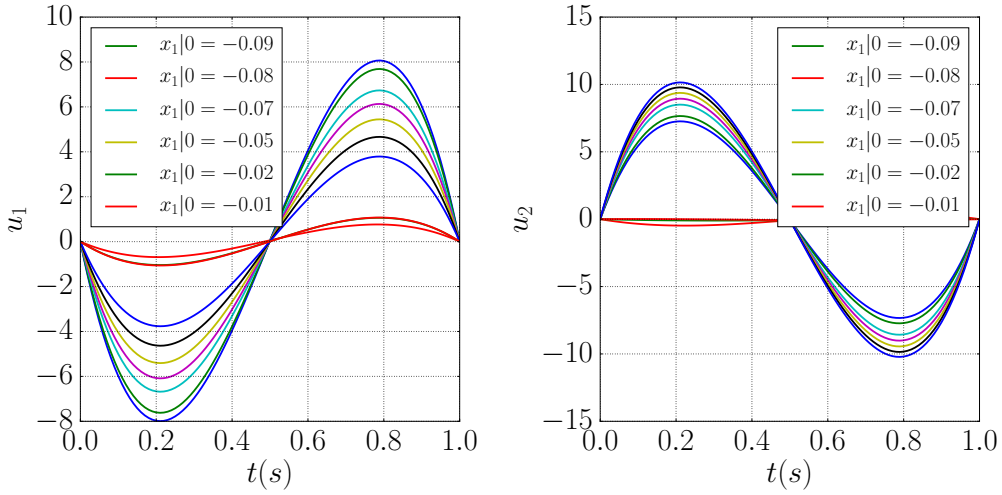


Abbildung 4.5 – Eingangsverläufe mit den Anfangswerten von x_1 zwischen -0.10 und -0.01.

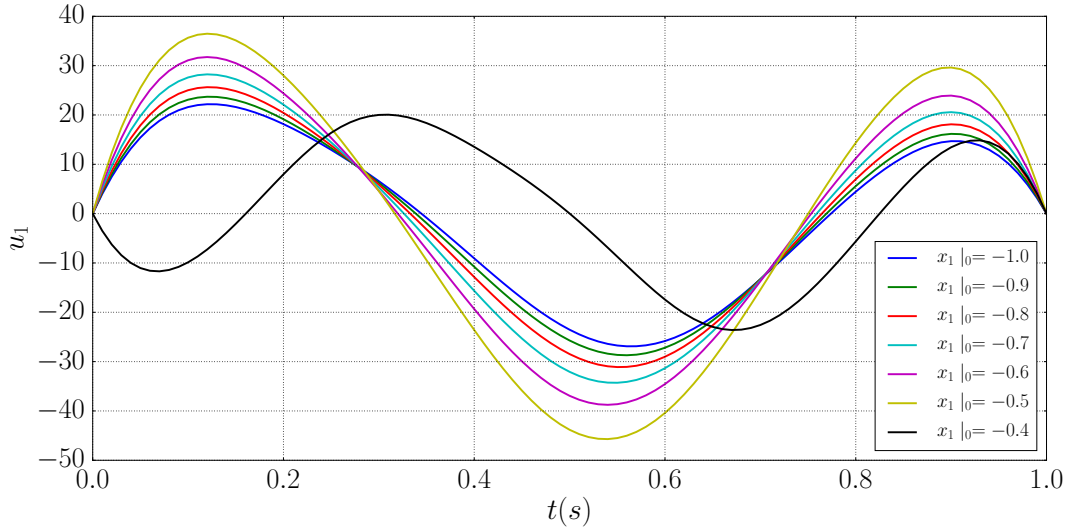


Abbildung 4.6 – Eingangsverläufe mit den Anfangswerten von x_1 zwischen -1.0 und -0.4.

Untersuchungen, nämlich lässt $x_{1,0}$ und $x_{2,0}$ jeweils in dem Bereich $(-1.0, -0.5)$ und $(0.2, 2.9)$ gleichzeitig annehmen, bleiben die Trends alle Kurven nicht identisch (wie Abb. 4.7).

Daraus ergibt sich die Konsequenz[3]: wenn die Anfangswerte von **nur einem Systemzu-**

[3]Das System des zwei-Gelenken-Manipulators wird auch untersucht, dessen Ergebnis im Anhang A.1

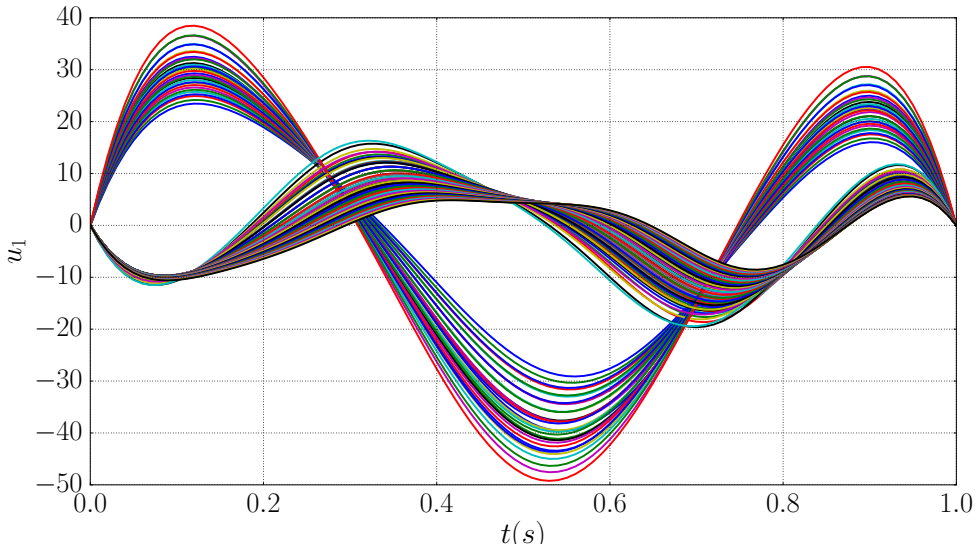


Abbildung 4.7 – Eingangsverläufe mit den Anfangswerten von x_1 zwischen -1.0 und -0.5 und x_2 zwischen 0.2 und 2.9.

standskomponente in einem **abgrenzten** Bereich abwandeln, besitzt die Amplitude und die Form der Trajektorienkurven mittels des Python-Pakets *PyTrajectory* eine Regelmäßigkeit. Das gilt in dem System, egal es die Brockett 2. Bedingung erfüllt oder nicht. Unter dieser Voraussetzung kann man dann die Möglichkeit eines Steuergesetz $\boldsymbol{u}(t)$ überlegen.

4.2.3 Entwurf des Steuergesetz mittels Trajektorien aus *PyTrajectory*

Da die Ueberfuehrungstrajektorien einen allmählich änderte Form in einem Bereich besitzen, ist es möglich, einen Steuergesetz für den gesamten Bereich zu konstruieren. Mit *PyTrajectory* kann im ersten Schritt jeweils eine Trajektorie für die Systemzustandsveränderung aus einigen Startposition in der Umgebung einer Ruhelage in die wie im letzten Abschnitt entworfen werden, dann werden die Trajektorien $\boldsymbol{u}_t$ mit Interpolation dieser bekannten Trajektorien erstellt.

Aus Zeitgründen wird in der Arbeit nur ein sehr einfacher Fall vorgestellt: Entwurf des Steuergesetz mit unterschiedlichen $x_{1,0}$.

Wie in dem letzten Abschnitt gezeigt wird, wählt man den Veränderungsbereich von $x_{1,0}$ zwischen $(-0.07, -0.03)$ mit der Schrittweite 0.01. Die bekannte $\boldsymbol{u}(t)$ von *PyTrajectory* mit beispielsweise p Zeitpunkten sind in einer List gespeichert. Die ausgesuchten Ein-

gezeigt ist.

gangstrajektorien mit z.B. $x_{1,0} = -0.043$ braucht zuerst die gehörenden Strecke nämlich einen Bereich zwischen $(-0.05, -0.04)$ zu finden. Damit die Größe der Trajektorie zu jedem Zeitpunkt p_i kann mittels der Punkte in der Kurven mit $x_{1,0} = -0.05$ und -0.04 linear interpoliert.

Die Interpolationsergebnisse sind in Abb. 4.8 und 4.9 gezeigt. Nur drei der Trajektorien wurden aus *PyTrajectory* vorgegeben, die anderen sieben sind dadurch interpoliert. In allen Fällen laufen Systemzustandsvariablen schließlich zu der Ruhelage.

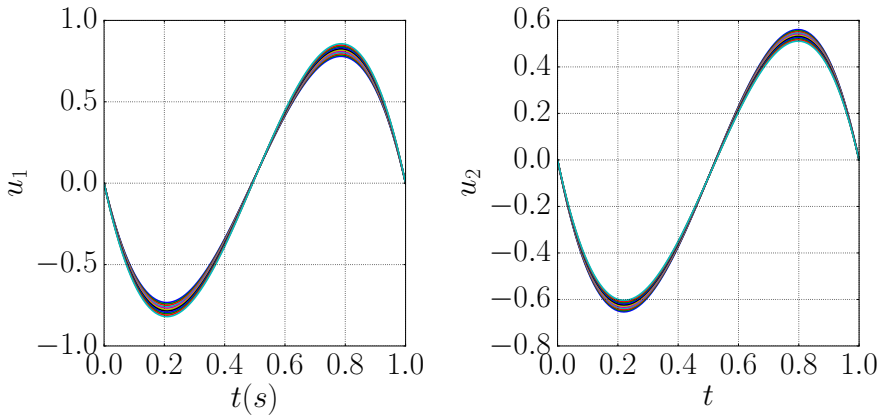


Abbildung 4.8 – Eingangsverlauf mit dem Anfangszustand von x_1 in $(-0.05, -0.04)$ mit Interpolation.

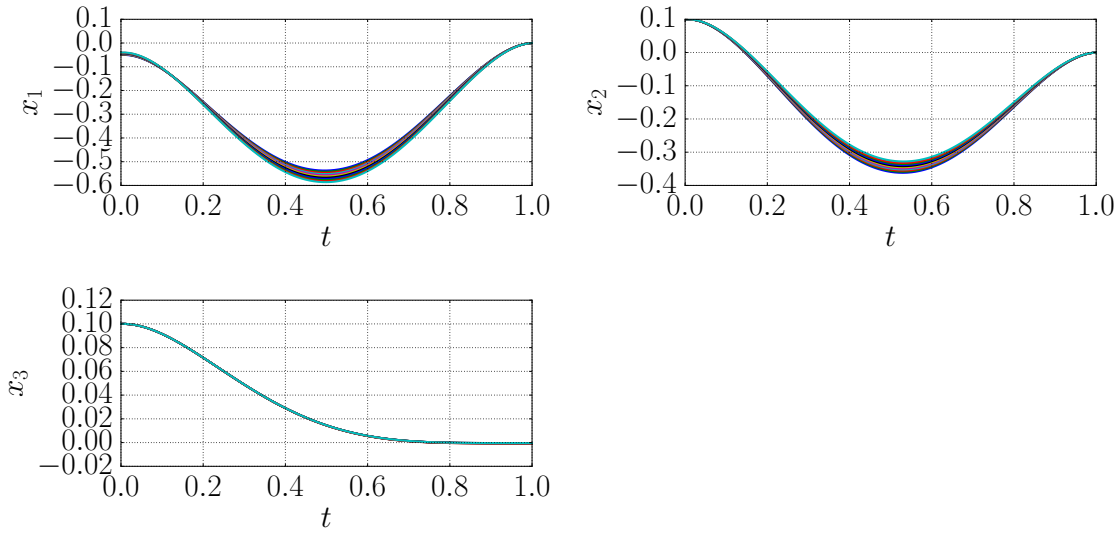


Abbildung 4.9 – Zustandsverlauf mit dem Anfangszustand von x_1 in $(-0.05, -0.04)$ mit Interpolation.

4.3 Trajektorienplanung mit Polynomform von $\boldsymbol{u}(t)$

In diesem Abschnitt wird ein Steuergesetz für das Brockett Beispiel ohne Berücksichtigung der geplannten Trajketorien aus *Pytrajectory* diskutiert. Zuerst wird eine Polynomform für die Systemeingänge aufgestellt, die die 2. Bedingung von Brockett erfüllt.

Es wird angenommen, dass u_1 und u_2 jeweils ein Polynom der Zeit t ist:

$$\begin{aligned} u_1 &= a_0 + a_1 t + a_2 t^2 \\ u_2 &= b_0 + b_1 t + b_2 t^2. \end{aligned} \tag{4.6}$$

Integriert man die ersten zwei Gleichungen in Gl. (4.6), erhält man:

$$\begin{aligned} x_1 &= a_0 t + \frac{1}{2} a_1 t^2 + \frac{1}{3} a_2 t^3 + \alpha \\ x_2 &= b_0 t + \frac{1}{2} b_1 t^2 + \frac{1}{3} b_2 t^3 + \beta. \end{aligned} \tag{4.7}$$

Setzt man Gln. (4.6), (4.7) in die letzte Zeile von Gl. (4.5) ein[4]:

$$\begin{aligned} \dot{x}_3 &= x_2 u_1 - x_1 u_2 = (a_0\beta - b_0\alpha) + t(a_1\beta - b_1\alpha) \\ &+ t^2(\frac{1}{2} a_1 b_0 - \frac{1}{2} a_0 b_1 + a_2\beta - b_2\alpha) + \frac{2}{3} t^3 (a_2 b_0 - a_0 b_2) + \frac{1}{6} t^4 (a_2 b_1 - a_1 b_2) \end{aligned} \tag{4.8}$$

daraus folgt die Form von x_3:

$$\begin{aligned} x_3 &= t(a_0\beta - b_0\alpha) + \frac{1}{2} t^2 (a_1\beta - b_1\alpha) \\ &+ \frac{1}{3} t^3 (\frac{1}{2} a_1 b_0 + \beta a_2 - \frac{1}{2} a_0 b_1 - \alpha b_2) + \frac{1}{6} t^4 (a_2 b_0 - a_0 b_2) + \frac{1}{30} t^5 (a_2 b_1 - a_1 b_2) + \gamma \end{aligned} \tag{4.9}$$

Falls die gesamte Überführungszeit $T = 1s$ ist, dann :

$$\begin{aligned} x_{1,end} &= a_0 + \frac{1}{2} a_1 + \frac{1}{3} a_2 + \alpha \stackrel{!}{=} 0 \\ x_{2,end} &= b_0 + \frac{1}{2} b_1 + \frac{1}{3} b_2 + \beta \stackrel{!}{=} 0 \\ x_{3,end} &= (a_0\beta - b_0\alpha) + \frac{1}{2}(a_1\beta - b_1\alpha) \\ &+ \frac{1}{3}(\frac{1}{2} a_1 b_0 + \beta a_2 - \frac{1}{2} a_0 b_1 - \alpha b_2) + \frac{1}{6}(a_2 b_0 - a_0 b_2) + \frac{1}{30}(a_2 b_1 - a_1 b_2) + \gamma \stackrel{!}{=} 0 \end{aligned} \tag{4.10}$$

[4]Der Koeffizient von t^5 ist gleich null.

Gl. (4.10) ist ein unterbestimmtes Gleichungssystem mit 6 Variablen und 3 Funktionen. Als Beispiel wählt man hier a_0, a_1 und b_0 als unabhängige Variablen, dann lassen sich a_0, b_1 und b_2 daraus darstellen:

$$\begin{aligned} a_2 &= -3.0a_0 - 1.5a_1 - 3.0\alpha \\ b_1 &= \frac{1}{a_0 + 6.0\alpha}\left(30.0a_0\beta + a_1b_0 + 6.0a_1\beta - 30.0\alpha b_0 + 60.0\gamma\right) \\ b_2 &= \frac{1}{a_0 + 6.0\alpha}\left(-3.0a_0b_0 - 48.0a_0\beta - 1.5a_1b_0 - 9.0a_1\beta + 27.0\alpha b_0 - 18.0\alpha\beta - 90.0\gamma\right) \end{aligned} \tag{4.11}$$

Zur Vermeidung der Singularität muss $a_0 \neq -6\alpha$ sein. Unter dieser Beschränkung können α, β, γ und a_0, a_1, b_0 beliebig gewählt werden. (Daher ist der Ursprung mit dem polynomischen Steuergesetz $\boldsymbol{u}(t)$ asymptotisch stabil.)

Das heißt, diese Form von $\boldsymbol{u}(t)$ ermöglicht es, eine Trajektorie von $\boldsymbol{x}(t)$ von irgend einem Punkt in einer Umgebung der Ruhelage in die Ruhelage führen zu lassen. Zur Verifizierung verwendet man hier auch den Würfel mit der Seitenlänge 0.1 als die Umgebung und die 27 Testpunkte als Anfangswerte von $\boldsymbol{x}$. Die anderen Parameter sind wie so eingestellt: $a_0 = a_1 = b_0 = 1$.

Die berechnete Kurven stehen in Abb. 4.10 und 4.11. Wie erwartet läuft die Trajektorie von $\boldsymbol{x}$ in jedem Fall von der bestimmten Ausgangspunkt zur Ruhelage. Ein interessantes Phänomen liegt in den Abbildungen von x_1 und u_1: es gibt nur 3 Kurven in jedem Bild. Die Erklärung dafür ist mit den Definitionsgleichungen Gl. (4.6) und (4.7) einfach: u_1 und x_1 sind nur von a_0, a_1 und a_2 abhängig, wobei der ersten zwei konstant 1 sind und $a_2 = -3a_0 - 1.5a_1 - 3\alpha$ mit α zu $(-0.1, 0.0, 0.1)$ gehört (Fall $1-9 : \alpha = -0.1$, Fall $10-18 : \alpha = 0.0$ und Fall $19-27 : \alpha = 0.1$).

Die Form von $\boldsymbol{u}$ ist zwar relativ einfach, aber zum Zeitpunkt $t = 0s$ und $t = 1s$ hat u_1 einen Sprung von 0 zu 1 und von -2.5 zu 0.0, und zu $t = 1s$ verschwindet u_2 die Stetigkeit auch. In dem nächsten Abschnitt werden zwei Regelgesetze entworfen.

4.4 Reglerentwurf für den Brocketts nicht-holonomen Doppelintegrator in [15]

Aus Abschnitt 2.2 ist schon bekannt, der Brocketts nichtholonomer Doppelintegratordie Brocketts Bedingung nicht erfüllt, deshalb existiert kein C^1 Regelgesetz, dass die Ruhelage dieses Systems asymptotisch stabilisiert wird. In der Literatur [15] stellte der Autor D. Liberzon einen Schaltregelgesetz auf, mit dem die Systemzustände aus einem Punkt in der Umgebung der Ruhelage endlich zur Ruhelage gehen können.

Ausgangspunkt ist die Transformation der Systemdarstellung in Kartesischen Koordina-

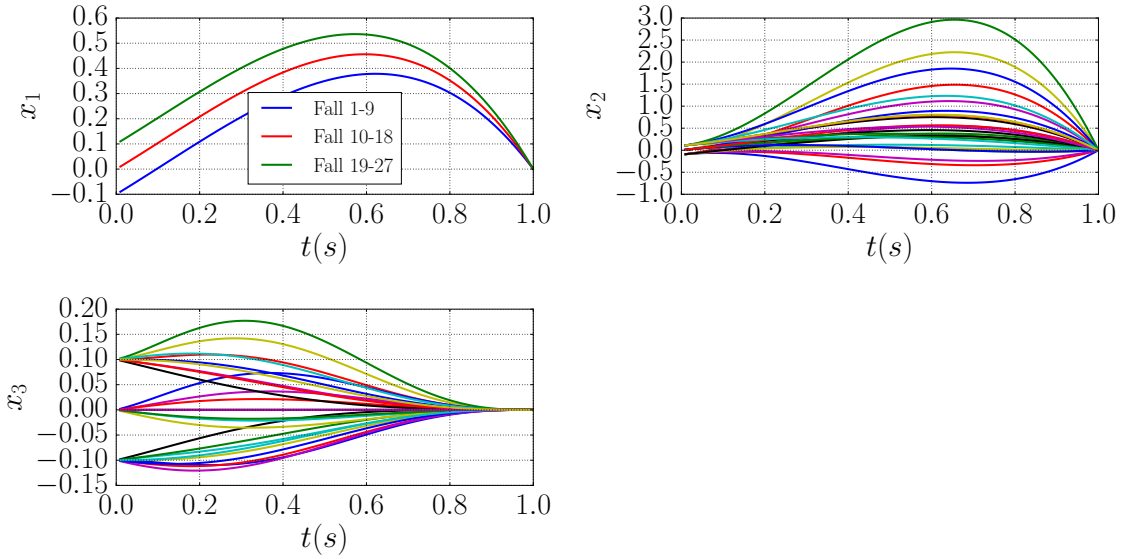


Abbildung 4.10 – Verlauf der Zustandskomponente mittels Polynom-Ansatz für $\boldsymbol{u}$.

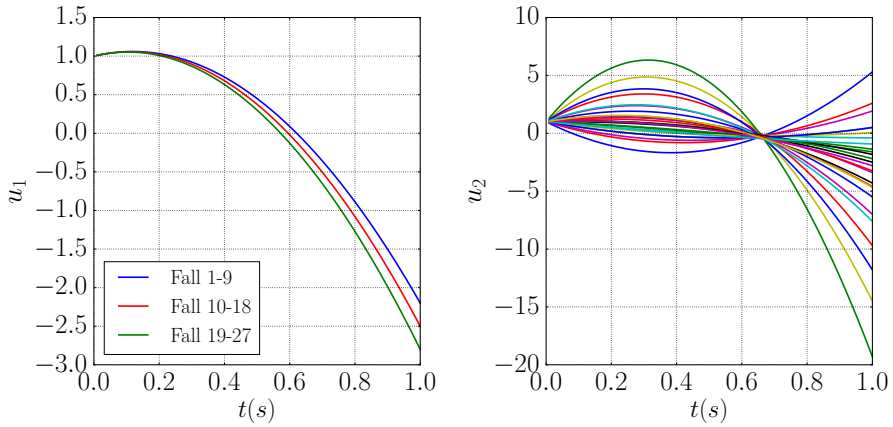


Abbildung 4.11 – Verlauf der Systemeingänge mittels Polynom-Ansatz für $\boldsymbol{u}$.

ten zur Zylinderkoordinaten. Anhand der Umrechnung:

$$\begin{aligned} r &= \sqrt{x_1^2 + x_2^2} \\ \varphi &= \arctan2(x_2, x_1) \\ z &= x_3 \end{aligned} \tag{4.12}$$

und

$$\begin{aligned} u_z &= \cos(\varphi)u_1 + \sin(\varphi)u_2 \\ v_z &= \cos(\varphi)u_2 - \sin(\varphi)u_1 \end{aligned} \tag{4.13}$$

erhält man die Systemgleichungen in Zylinderkoordinaten:

$$\begin{aligned} \dot{r} &= u_z \\ \dot{\varphi} &= \frac{v_z}{r} \\ \dot{z} &= -r \cdot v_z. \end{aligned} \tag{4.14}$$

In [15] wird das Regelgesetz:

$$\begin{aligned} u_z &= -r^2 \\ v_z &= z \end{aligned} \tag{4.15}$$

entworfen. Der Regler ist geeignet für alle Anfangspunkte außerhalb der Situation bei[5] $r_0 = 0$, deswegen muss man bei r_0 einen neuen Regler anwenden, um die Trajektorie die z-Achse zu verlassen. Ein möglicher Regler lässt sich wie z.B. $u_z = 1, v_z = 0$ in einem begrenzten Anfangszeit T_{phase1} entwerfen, damit r genug weit von z-Achse geht während φ und z still bleiben. Bemerkenswert ist, dass der Wert von r immer größer gleich wie 0 ist. Nach der Zeit T_{phase1} funktioniert wieder das Regelgesetz aus Gl. (4.15).

Zwei Fälle werden für die Idee auf das System getestet. Zunächst fängt man mit der Trajektorieplanung mit dem Anfangszustand gleich $(0, 5, 5)$ an. T_{phase1} ist als $0.2s$ eingestellt. Die Linie von r in Abb. 4.12 nimmt in den ersten $0.2s$ zu ungefähr 0.2 zu, danach fällt sie bis zur Endzeit T_{end} langsam ab. Innerhalb von $0.2s$ bleiben die Werte von φ und z unverändert, dann nehmen beide zu. Es ist zu beachten, dass r und z zu 0 konvergieren können, aber die Tendenz von φ neigt nicht zu 0 (aber mit einem großen oder kleinen φ_{end} kann das Systemzustandsvektor noch in die Ruhelage erreichen, sobald der Radius r und die Höhe z gleich null sind.). Abb. 4.13 zeigt die dreidimensionale Trajektorie mit zwei Phasen deutlich.

In der zweiten Situation stellt man unterschiedliche zufällige Anfangswerte von Zustände innerhalb von $[-0.5, 0.5)$ ein[6]. Wie in Abb. 4.14 zu sehen ist, konvergieren r und z zu 0 mit einer kleinen Geschwindigkeit, während die Kurven von φ in den meisten Fällen divergieren.

Zusammengefasst funktioniert das Regelgesetz, aber mit dem Schönheitsfehler einer langsamen Konvergenzgeschwindigkeit. Zur Verbesserung wird ein komplexeres Regelgesetz im nächsten Abschnitt vorgestellt.

[5] Die Erklärung kann man in der Literatur [15] lesen.
[6] Bei dem Fall $r < 0$ wird r durch der absolute Betrag ersetzt.

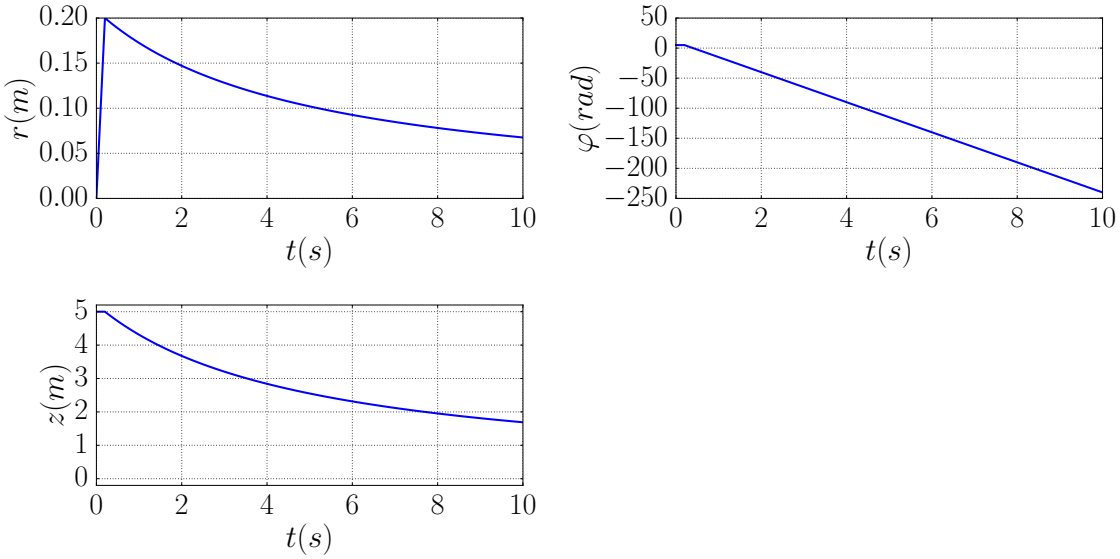


Abbildung 4.12 – Zustandsverläufe in Zylinderkoordinaten mit dem Startwert $(0, 5, 5)$.

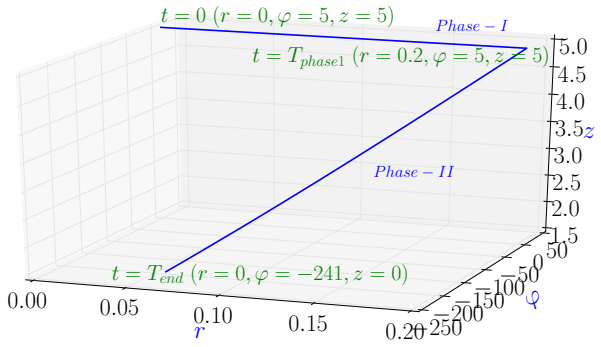


Abbildung 4.13 – 3D-Kurve von Systemzustand in Zylinderkoordinaten mit dem Startwert $(0, 5, 5)$.

4.5 Ein dreiphasiger Reglerentwurf für Brocketts nichtholonomen Doppelintegrator

In Zusammenarbeit mit dem Betreuer dieser Arbeit werde der folgende Ansatz entwickelt: ein schaltendes Regelgesetz mit zwei oder drei Phasen ist geeignet, um die Trajektorien aller Systemzustände in $(0, 0, 0)$ erfolgreich anzukommen. Der Gedankengang kommt aus der Analyse der Trajektorie des Zustandsvektors. Ähnlich wie Gl. (4.12) im letzten Abschnitt kann die Zustandsfunktion mit der Definition von Systemeingänge in die

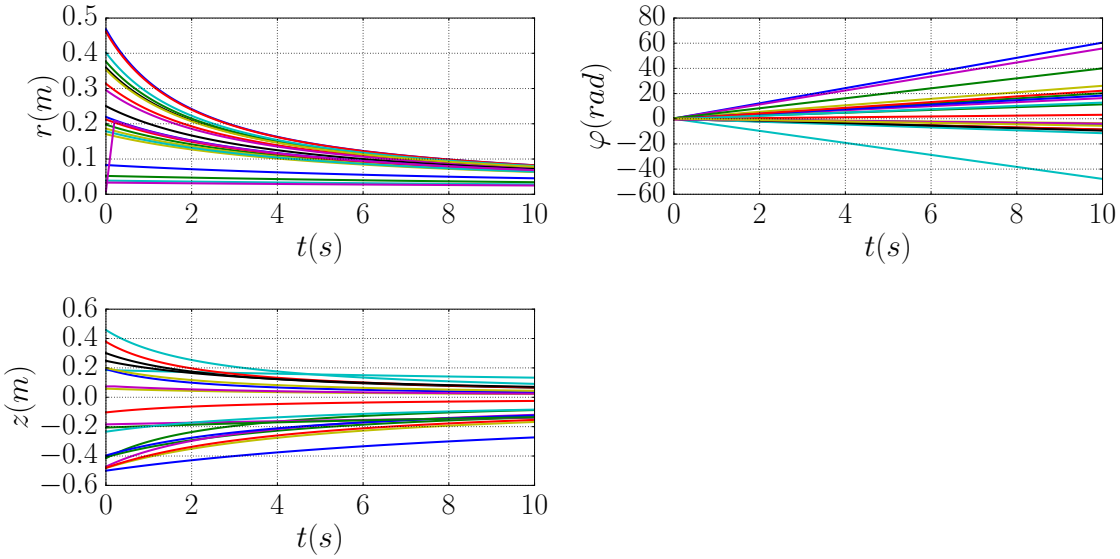


Abbildung 4.14 – Trajektorien von r, φ, z mit unterschiedlichen Anfangswerten.

Zylinderkoordinaten transformiert werden:

$$\begin{aligned} \dot{r} &= \cos(\varphi)u_1 + \sin(\varphi)u_2 =: u_z \\ \dot{\varphi} &= \frac{1}{r}(u_2\cos(\varphi) - u_1\sin(\varphi)) =: v_z \\ \dot{z} &= r(sin(\varphi)u_1 - \cos(\varphi)u_2) = -v_z r^2. \end{aligned} \tag{4.16}$$

Unter einer Schraubenlinie versteht man eine Kurve, die mit dem unveränderten Radius nach unten/oben rotiert. Auf der Basis kann eine Trajektorie von Systemzustand mit irgend einem Startwert weit von z-Achse ähnlich wie eine Schraubenlinie in die $z = 0$ Ebene gehen. Die Skizze ist in Abb. 4.15 zu sehen.

Wenn $r > 0$ läuft die Trajektorie von $\boldsymbol{x}$ in Phase-II von $x_{3,0}$ bis zur X_1-X_2-Ebene bei $x_{3,0} > 0$ abwärts, sonst von unten nach oben. Wenn der Punkt auf der X_1-X_2-Ebene weit von der Ruhelage liegt, kann sich der Regler in Phase-III so einstellen lassen, dass x_1 und x_2 (oder r in der Zylinderkoordinate) nach und nach bis zu 0 verkleinern und gleichzeitig x_3 (oder z in Zylinderkoordinate) noch 0 bleibt.

Ein Ausnahme steht in der Situation, wenn der Anfangswert von $\boldsymbol{x}$ auf der z-Achse liegt. Bei $x_{1,0}$ und $x_{2,0}$ beide 0 ist der Radius r_0 auch 0. Unter der Berücksichtigung der Unveränderlichkeit von r in der ganzen Phase-II ändert sich $\dot{z}$ in Gl. (4.16) auch nicht. Deswegen kann die Trajektorie nie nur mittels der vorherigen Idee zur Ruhelage erreichen. Dieses Problem kann man mit dem Entwurf einer zusätzlichen Phase vor Phase-II gelöst werden. Während der Phase-I verändert nur r, der am Ende dieser Phase weit von der z-Achse liegt.

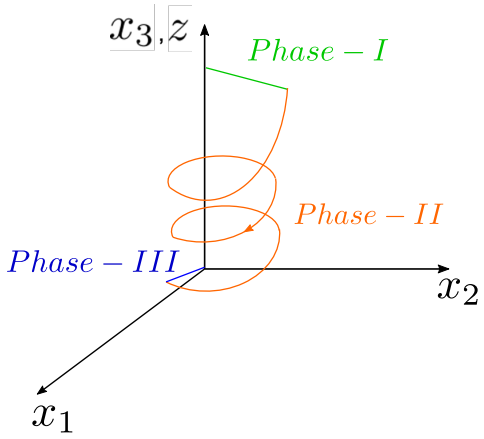


Abbildung 4.15 – Skizze der Trajektorie des Zustandsvektors mit der Schraubenlinie.

Und schließlich ergibt sich noch eine Frage: Wie weit muss sich r in Phase-I von z-Achse entfernen? Das Problem kann man mit dem kürzesten Weg der gesamten Trajektorie beantworten. Die Länge des Pfads der drei Pahsen ist[7]:

$$\begin{aligned} L_{Tra} &= r_1 + \int_0^{T_2} \sqrt{\dot{x}_1^2 + \dot{x}_2^2 + \dot{x}_3^2}\, dt + r_1 \\ &= \frac{z_0}{r_1}\sqrt{r_1^2 + 1} + 2r_1 \end{aligned} \tag{4.17}$$

wobei die Bogenlänge in Phase-I gleich ist wie in Phase-III und mit r_1 gekennzeichnet wird. T_2 steht für die Überführungszeit in Phase-II. Setzt man die Ableitung von L_{Tra} nach r_1:

$$\frac{\partial L_{Tra}}{\partial r_1} = \frac{z_0}{\sqrt{r_1^2 + 1}} - \frac{z_0}{r_1^2}\sqrt{r_1^2 + 1} + 2 \tag{4.18}$$

gleich null ein, wird eine von z_0 abhängigen optimale Bogenlänge in Phase-I $r_{1,opt}$ ausgerechnet, mit der L_{Tra} am kleinsten ist. Wenn der Anfangswert von r_0 kleiner als $r_{1,opt}$ ist, soll r noch weiter von z entfernen. Im anderen Fall geht das System mit dem Regelgesetz direkt in die Phase-II.

Der nächste Schritt ist der Entwurf von $\boldsymbol{u}$ gemäß der obere Idee. Fängt man mit dem Systemeingang in Phase-I an. Wie gerade diskutiert soll nur r innerhalb von diesen Phase verändern werden, dann lässt sich $u_z = 1$ und $v_z = 0$ einsetzen. In Phase-II ist es gerade umgekehrt, $u_z = 0$ und $v_z = sign(z)$, sodass der Absolutwert von z immer kleiner wird. Am Ende wenn die Trajektorie schon auf X_1-X_2-Ebene sehr nähe liegt, wird der Radius r mit $u_z = -1$ und $v_z = 0$ verkleinert.

Dieses Regelgesetz wird auch wieder in zwei Fällen getestet. Der erste ist ähnlich wie dem letzten Abschnitt in Bezug auf *einen* Startwert von $\boldsymbol{x}$. Abb. 4.16 und 4.17 zeigen

[7]Der detaillierte Rechenverlauf ist in Anhang A.3 dargestellt

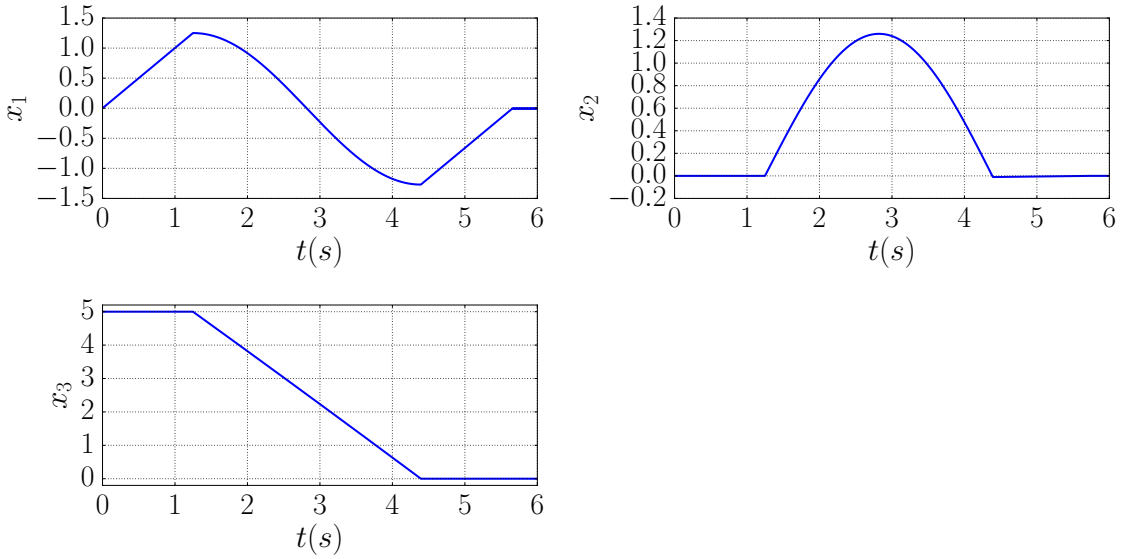


Abbildung 4.16 – Zustandsverläufe mit dem Startwert $(0, 5, 5)$.

die simulierten Ergebnissen mit $\boldsymbol{x}_0 = (0, 0, 5)$ und $T_{end} = 6s$. Die Ergebnisse entsprechen der Erwartung. $r_{1,opt}$ ist ungefähr 1.29. Mit $u_z/v_z = 1$ dauern die Phase-I und Phase-III jeweils $1.29s$.

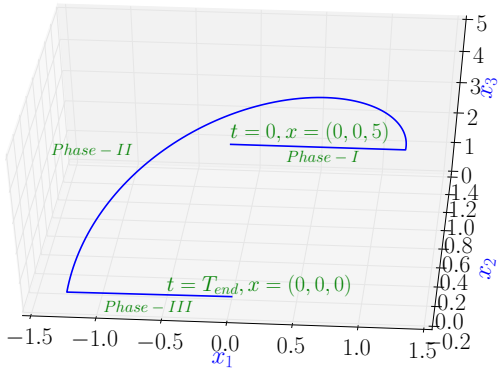


Abbildung 4.17 – 3D-Kurve des Systemzustands mit dem Startwert $(0, 5, 5)$.

Das zweite Beispiel zielt auf die Prüfung der asymptotischen Stabilität der Ruhelage $(0, 0, 0)$. Anfangswerte sind eine zufällige Liste mit 20 Elementen, die in dem Wertebereich $[-0.5, 0.5)$ liegen. Der Endzeit $T_{end} = 4s$. Als Ergebnis kann die Trajektorie in jedem Fall die Ruhelage erreichen. Die asymptotische Stabilität wird dadurch verifiziert.

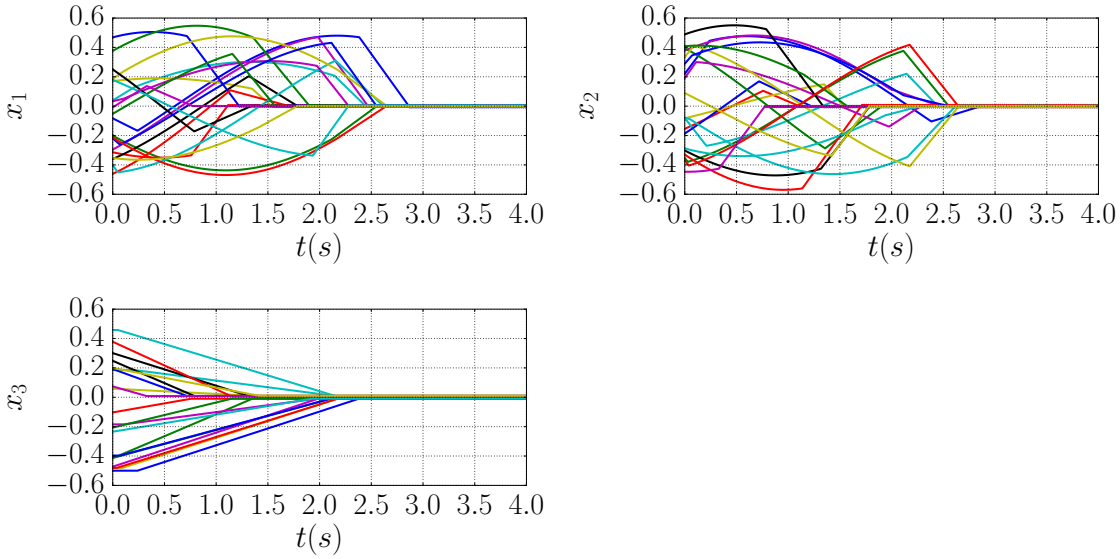


Abbildung 4.18 – Zustandsverläufe mit unterschiedlichen Anfangswerten.

Im Vergleich mit dem Regelgesetz aus Literatur [15] b.z.w. Abschnitt 4.4 hat dieses einige Vorteile. Ein wichtiger Punkt liegt in die kürze Konvergenz-Zeit der Trajektorie. Mit gleichen Anfangswerten von $\boldsymbol{x}_0$ ist die Trajektorie mit dem Regelgesetz in [15] in $10s$ noch nicht fertig, während die mit diesem Regelgesetz nach $3s$ schon zur Ruhelage ankommt. Die Konvergenzgeschwindigkeit ist deutlich schneller.

Bemerkenswert ist, dass das in drei Phasen schaltende Regelgesetz nicht kontinuierlich ist, nämlich die Brockett Bedingung nicht verletzt.

Fazit: In diesem Kapitel wurden einige Steuerung und Regler für die Untersuchung der asymptotischen Stabilität der Ruhelage in Brocketts berühmten nicht-holonomen Doppelintegrator vorgestellt. Jeder kann die Trajektorie schließlich in die Ruhelage führen, auch wenn jeder nicht stetig differenzierbar ist. Das steht im Einklang mit Brocketts Bedingungen.

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

Das Ziel der vorliegenden Diplomarbeit war die Analyse der Überführungstrajektorien eines unteraktuierten Systems aus der Umgebung einer Ruhelage in diese Ruhelage mittels des Python-Pakets *PyTrajectory*, welches um eine Funktion der Bestimmung der optimalen Überführungszeit erweitert wurde.

Das Theorem von Brockett zur Bestimmung der Existenz einer stetigen differenzierbaren Regelgesetz wurde zuerst in Kapitel 2 vorgestellt. Als Anwendung existiert für das in Kapitel 3 und 4 betrachtete Beispiel-das nicht-holonomen Doppelintegrator-System nach Brocketts Theorem keines stetig differenzierbares Regelgesetz zu verfügen, um die Ruhelage asymptotisch zu stabilisieren. Im Gegensatz dazu erfüllt das berühmte Benchmark System-“inverse-Pendel” diese notwendige Bedingung.

Zum Vergleich der Überführungstrajektorien dieser zwei Systeme wurde das Python-Paket *PyTrajectory* als das Werkzeug verwendet, um die Trajektorien durch Lösung einer Randwertaufgabe zu entwerfen. Eine Funktion davon wurde erweitert, damit die Überführungszeit des Regelvorgangs auch ein Berechnungsergebnis ist, statt vom Nutzer vorgegeben zu werden. Der Ausgangspunkt lag in der Transformation der Zeitkoordinaten, die durch die Ergänzung eines zusätzlichen Parameters in der Systemzustandsfunktion realisiert wurde. Aber wegen des nicht brauchbaren Ergebnis des Parameters wurde eine Straffunktion entworfen, mit der eine bessere Lösung ausgerechnet wurde.

Danach wurde die Überführugstrajektorie zweier Systeme mit unterschiedlichen Anfangswerten von Systemzuständen mittels *PyTrajectory* analysiert. Die Regelmäßigkeit der Trajektorien ist nur in einem **begrenzten** Bereich für beide Systeme effektiv hängt nicht davon, ob das System die Brockett-Bedingung bezüglich erfüllt. Mit anderen Worten ist die Konstruktion eines Regelgesetz zur Lösung der Randwertaufgabe mit Anfangszuständen in einer relativ großen Umgebung der Ruhelage schwierig. Aus diesem Grund wurden in Kapitel 4 einige nicht stetig-differenzierbaren Regelgesetze unter Berücksichtigung der Besonderheit der Systemzustandsdarstellung für den nicht-holonomen

Doppelintegrator entworfen, mit den die Trajektorien aus der Umgebung der Ruhelage in diese ermöglicht wurden.

5.2 Ausblick

Für die Zukunft empfiehlt die Autorin die Vervollständigung der Analyse der Überführungstrajektorien aus einer Umgebung der gewünschten Ruhelage in diese. In der Arbeit wurden nur drei typischen unteraktuierte Systeme untersucht. Das Modell wie das Acrobot-System [23] wäre beispielsweise ein guter Anfangspunkt.

Aus Zeitgründen erwägt die Autorin in der Arbeit den Entwurf eines Steuergesetzes durch die Interpolation der ausgerechneten Trajektorien aus *PyTrajectory*, das nur in einem sehr einfachen Fall verwendet wurde, wobei sich nur ein Systemzustandselement in einem Bereich veränderte. Die Konstruktion eines Regelgesetz zur Lösung der Randwertaufgabe mit den Anfangswerten von $\boldsymbol{x}$ in zweidimensionalen Umgebung der Ruhelage soll auch durch Interpolation bekannter Trajektorien umsetzbarmöglichst geringer Bogenlängemöglichst geringer Bogenlänge sein.

Anhang A

Appendix

A.1 Zwei-Gelenke-Manipulator

Das zwei-Gelenke-Manipulator besteht aus vier Zustandskomponente mit der Systemzustandsgleichung [12]:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u_1 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= -\eta x_2^2 \cdot \sin(x_3) - (1 + \eta \cos(x_3)) \cdot u_1. \end{aligned} \tag{A.1}$$

In Abb. A.1 ist x_1 und x_2 jeweils der Winkel und Winkelgeschwindigkeit des ersten Gelenks und x_3 und x_4 des zweiten Gelenks. $\eta = \frac{m_2 l_1 r_2}{J_2 + m_2 r_2^2}$ ist der Trägheitsparameter. Der Manipulator erfüllt die Brockett 3. Bedingung nicht (mit dem Bild $(0, 0, 0, \varepsilon)^T$).

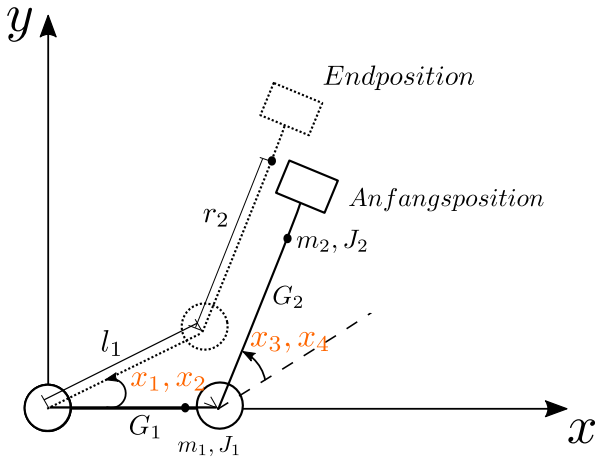


Abbildung A.1 – Skizze eines Zwei-Gelenke-Manipulators.

Für die Simulation in Abschnitt 3.2.2 sollte der Manipulator sich von $(0, 0, 72°, 0)$ zu

$(36°, 0, 36°, 0)$ bewegen. Die Randwerte von k sind wie 0.1 und 15 eingestellt. Anfangssplineabschnitte von $\boldsymbol{x}$ und $\boldsymbol{u}$ sind 10 und 20.

Für die Simulation in Abschnitt 4.2.2 ist die Umgebung von $\boldsymbol{x}_{3,0}$ zuerst eine Strecke von 63° zu 81°. Abb. A.2 zeigt, nur im Bereich von $(63° - 73.8°)$ verändert die Form der Trajektorien regelmäßig. Die Trajektorien bei der Veränderung von $x_{1,0}$ im Bereich

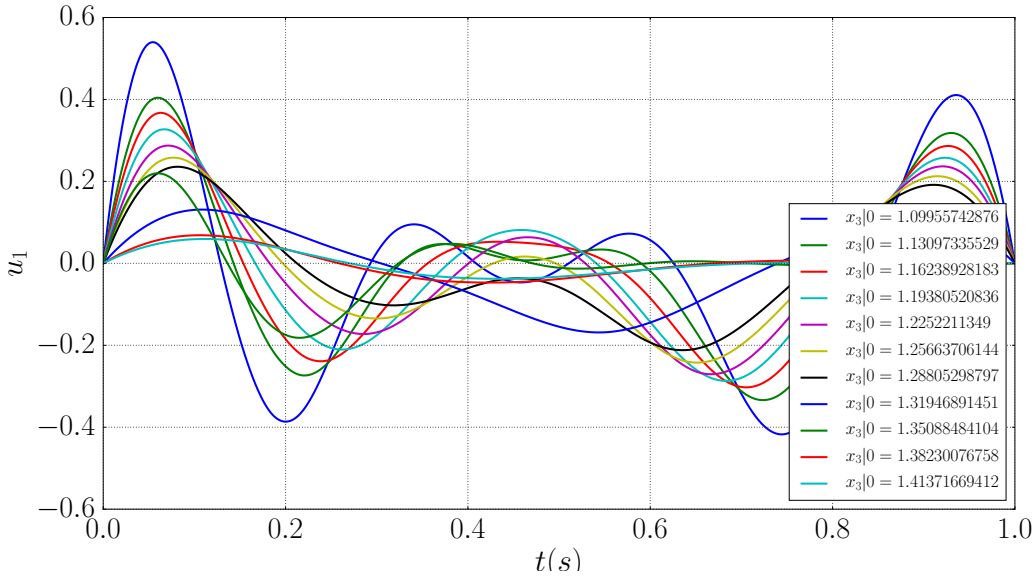


Abbildung A.2 – Trajektorien des Systemeingangs mit unterschiedlichen Anfangswerten von x_3

$(-9° - 9°)$ ist in Abb. A.3 dargestellt. Ähnlich wie Ergebnis von der Veränderung von x_3 ist die Form von u_t zwischen $-9°$ und 1.8° regelmäßig. Lässt sich $x_{1,0}$ und $x_{3,0}$ gleichzeitig verändern jeweils im Raum von $(-9° - 1.8°)$ und $(63° - 73.8°)$ ergibt sich für alle Trajektorien keine ähnliche Form. (Der regelmäßige Bereich ist weiterhin begrenzt.)

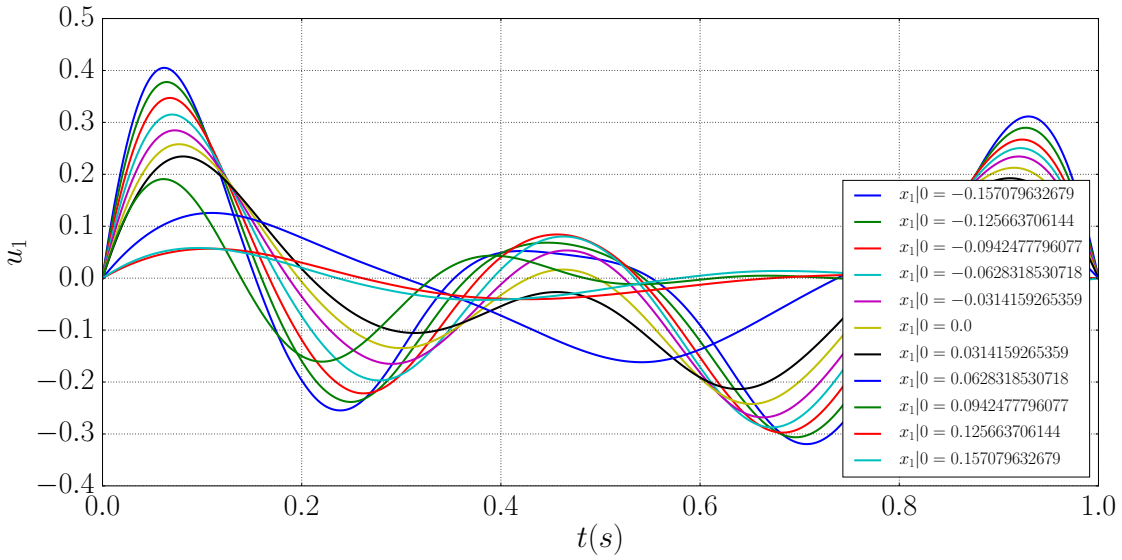


Abbildung A.3 – Trajektorien des Systemeingangs mit unterschiedlichen Anfangswerten von x_1.

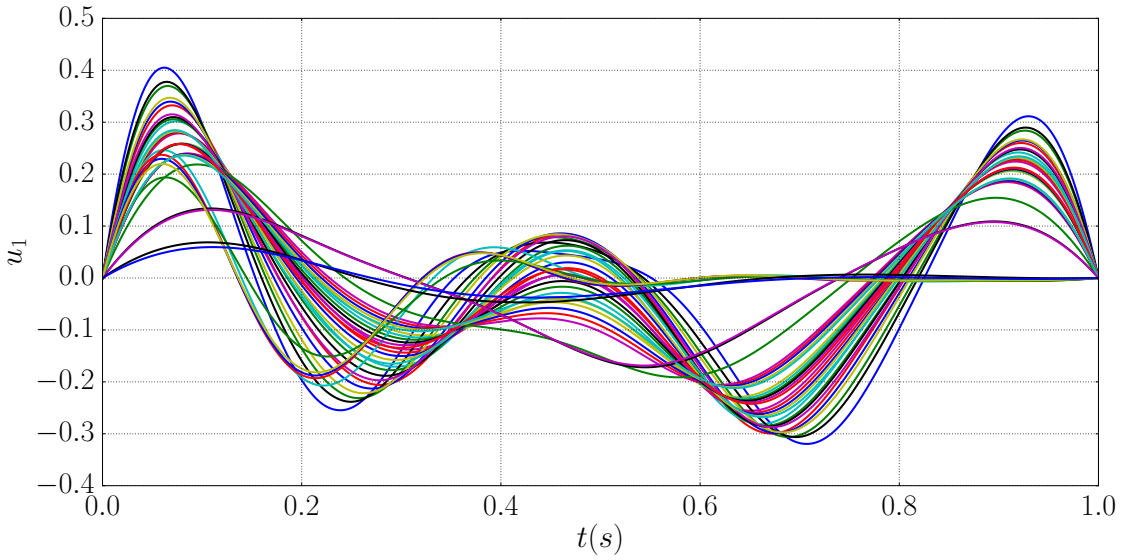


Abbildung A.4 – Trajektorien des Systemeingangs mit unterschiedlichen Anfangswerten von x_1 und x_3.

A.2 Tabelle für die Überprüfung der asymptotischen Stabilität des Brocketts nicht-holonomen Doppelintegrators

Tabelle A.1 – Die Form der Trajektorienkurve des Systemeingangs u_1 mit unterschiedlichen Anfangswerte von $\boldsymbol{x}$

Testpunkte $(x_{1,0}, x_{2,0}, x_{3,0})$	Form der Kurve
$(-1, 0, 0)$	sinusförmig
$(-1, 1, 0)$	sinusförmig
$(0, -1, 0)$	sinusförmig
$(0, -1, 1)$	sinusförmig
$(0, 0, -1)$	sinusförmig
$(0, 1, -1)$	sinusförmig
$(1, -1, -1)$	sinusförmig
$(1, -1, 1)$	sinusförmig
$(1, 0, -1)$	sinusförmig
$(1, 1, -1)$	sinusförmig
$(1, 1, 1)$	sinusförmig
$(1, 1, 0)$	parabelförmig
$(-1, -1, -1)$	kosinusförmig
$(-1, -1, 0)$	kosinusförmig
$(-1, -1, 1)$	kosinusförmig
$(-1, 0, -1)$	kosinusförmig
$(-1, 0, 1)$	kosinusförmig
$(-1, 1, -1)$	kosinusförmig
$(-1, 1, 0)$	kosinusförmig
$(0, -1, -1)$	kosinusförmig
$(0, 0, 0)$	kosinusförmig (Ruhelage)
$(0, 1, 0)$	kosinusförmig
$(0, 1, 1)$	kosinusförmig
$(1, -1, 0)$	kosinusförmig
$(1, 0, 0)$	kosinusförmig
$(1, 0, 1)$	kosinusförmig
$(1, 1, 0)$	kosinusförmig

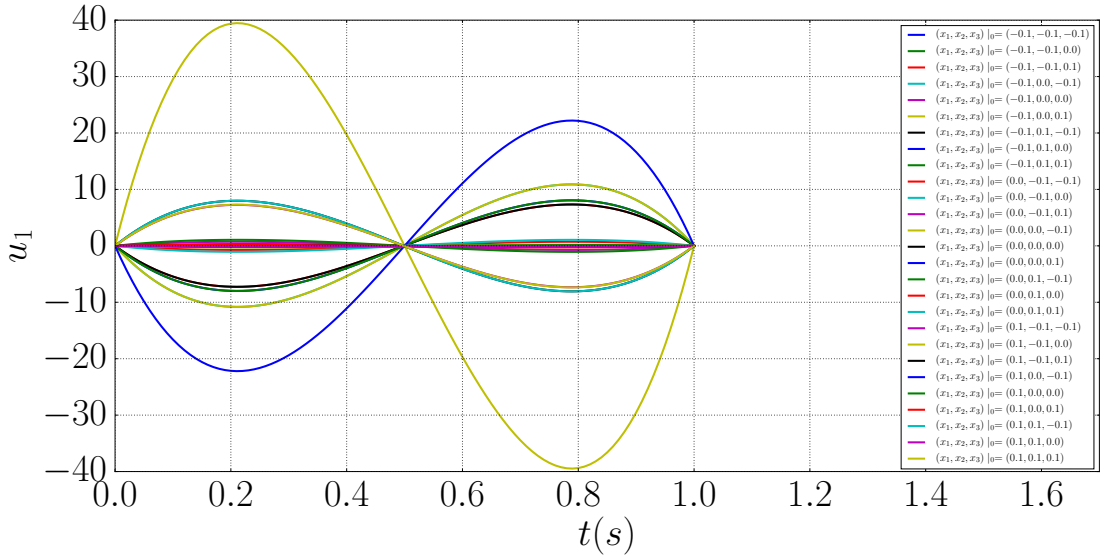


Abbildung A.5 – Eingangsverlauf u_1 des Brocketts nicht-holonomen Doppelintegrator.

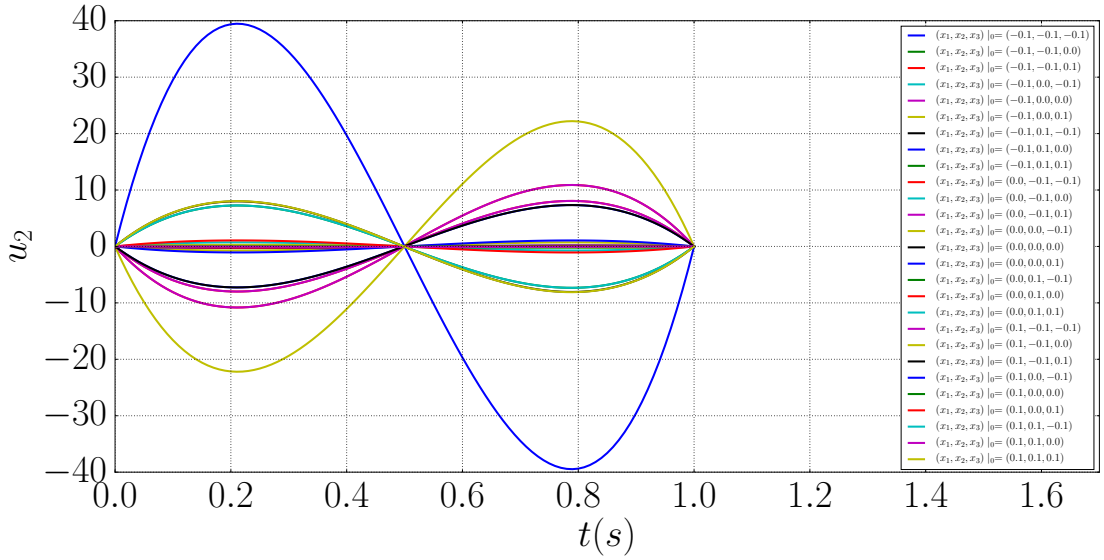


Abbildung A.6 – Eingangsverlauf u_2 des Brocketts nicht-holonomen Doppelintegrator.

A.3 Rechenverlauf für L_{Tra} in Abschnitt 4.5

$$\begin{aligned} L_{Tra} &= r_1 + \int_0^{T_2} \sqrt{\dot{x}_1^2 + \dot{x}_2^2 + \dot{x}_3^2}\, dt + r_1 \\ &= r_1 + \int_0^{T_2} \sqrt{(r_1\dot{z}_2)^2 + \dot{z}_3^2}\, dt + r_1 \\ &= 2r_1 + \int_0^{T_2} \sqrt{(r_1 v_2)^2 + r_1^4 v_2^2}\, dt \\ &= 2r_1 + (r_1 v_z)\sqrt{(1 + r_1^2)}T_2. \end{aligned} \tag{A.2}$$

Mit der Beziehung $z_0 = T_2 r_1^2 \cdot v_z$ lässt sich Gl. A.2 weiter vereinfachen:

$$\begin{aligned} L_{Tra} &= 2r_1 + (r_1 v_2)\sqrt{(1 + r_1^2)}\frac{z_0}{r_1^2 v_z} \\ &= \frac{z_0}{r_1}\sqrt{r_1^2 + 1} + 2r_1. \end{aligned} \tag{A.3}$$

A.4 Tabelle-Fehlerkorrektur

Tabelle A.2 – Fehlerkorrektur

Seite	Abschnitt	Beschreibung
21	Levenberg-Marquardt-Algorithmus	unter Gl. 3.12...Das heißt, der Wert von μ muss ...Anderenfalls muss μ...
33	Abschnitt 3.2.3	Abbildung 3.17
35	Abschnitt 3.2.4	Abbildung 3.21
46	Abschnitt 4.2.3	...die gehörenden Strecke nämlich einen Bereich zwischen **-0.05,-0.04** zu finden...mit $\mathbf{x_{1,0}} = \mathbf{-0.05}$ und **-0.04**... (In den Titel der Abb. auch.)
53	Abschnitt 4.5	Gln. 4.17, 4.18
63	Appendix A.3	Gln. A.2, A.3

Literaturverzeichnis

[1] *Welcome to PyTrajectory's documentation!*. https://pytrajectory.readthedocs.io/en/master/#welcome-to-pytrajectory-s-documentation.

[2] Brockett, R. W. et al.: *Asymptotic stability and feedback stabilization*. Differential geometric control theory, 27(1):181–191, 1983.

[3] Bronstein, I. N., J. Hromkovic, B. Luderer, H.-R. Schwarz, J. Blath, A. Schied, S. Dempe, G. Wanka, S. Gottwald, E. Zeidler et al.: *Taschenbuch der mathematik*, Bd. 1. Springer-Verlag, 2012.

[4] Colonius, F.: *Nichtlineare Kontrolltheorie Sommersemester 2012*. 2012.

[5] Coron, J.-M.: *Control and nonlinearity*. Nr. 136. American Mathematical Soc., 2007.

[6] Graichen, K. und M. Zeitz: *Inversionsbasierter Vorsteuerungsentwurf mit Ein- und Ausgangsbeschränkungen (Inversion-Based Feedforward Control Design under Input and Output Constraints)*. at-Automatisierungstechnik, 54(4/2006):187–199, 2006.

[7] Grosche, G., V. Ziegler, E. Zeidler und D. Ziegler: *Teubner-Taschenbuch der Mathematik*. Springer-Verlag, 2003.

[8] Harrach, B. von: *Einführung in die Optimierung*. Vorlesungsskript der Universität Stuttgart, Stand, 2015.

[9] Hund, C.: *Fixpunktsatz von Brouwer*. 2010.

[10] Janschek, K.: *Systementwurf mechatronischer Systeme: Methoden–Modelle–Konzepte*. Springer-Verlag, 2009.

[11] Khalil, H.: *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[12] Knoll, C.: *Steuerung und Regelung eines nicht-holonomen Manipulatormodells*. Diplomarbeit, Technische Universität Dresden, 2009.

[13] Knorrenschild, M.: *Numerische Mathematik: Eine beispielorientierte Einführung*. Carl Hanser Verlag GmbH Co KG, 2017.

[14] Kunze, A., O. Schnabel und C. Knoll: *PyTrajectory Documentation, 2015*, 2016.

[15] Liberzon, D.: *Switching in systems and control*. Springer Science & Business Media, 2012.

[16] Madsen, K., H. B. Nielsen und O. Tingleff: *Methods for non-linear least squares problems*. 2004.

[17] Olfati-Saber, R.: *Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles*. Doktorarbeit, Massachusetts Institute of Technology, 2001.

[18] Oriolo, G. und Y. Nakamura: *Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators*. In: *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, S. 2398–2403. IEEE, 1991.

[19] Orsi, R., L. Praly und I. Mareels: *Necessary conditions for stability and attractivity of continuous systems*. International Journal of Control, 76(11):1070–1077, 2003.

[20] Picci, G. und D. S. Gilliam: *Dynamical systems, control, coding, computer vision: new trends, interfaces, and interplay*, Bd. 25. Birkhäuser, 2012.

[21] Rudin, W.: *Analysis, 4*. Aufl. Oldenburg, München, Wien, 2009.

[22] Sontag, E. D.: *Mathematical control theory: deterministic finite dimensional systems*, Bd. 6. Springer Science & Business Media, 2013.

[23] Spong, M. W.: *The swing up control problem for the acrobot*. IEEE control systems, 15(1):49–55, 1995.

[24] Stern, R.: *Brockett's stabilization condition under state constraints*. Systems & control letters, 47(4):335–341, 2002.

[25] Tu, L. W.: *An introduction to manifolds*. Springer Science & Business Media, 2010.

[26] Wilson, F. W.: *The structure of the level surfaces of a Lyapunov function*. Journal of Differential Equations, 3(3):323–329, 1967.