Optimale Steuerung zur Kompensation der Pfadabdrängung von Industrierobotern

Optimal Control of Milling Industrial Robots

Master-Thesis by Elmar Brendel

August 2011

TECHNISCHE UNIVERSITÄT DARMSTADT

Optimale Steuerung zur Kompensation der Pfadabdrängung von Industrierobotern
Optimal Control of Milling Industrial Robots

Vorgelegte Master-Thesis von Elmar Brendel

Prüfer: Prof. Dr. Oskar von Stryk
Betreuer: Dr.-Ing. Christian Reinl

Tag der Einreichung: 06.08.2011

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 6. August 2011

(Elmar Brendel)

Kurzbeschreibung

Industrieroboter haben sich bereits erfolgreich in vielen Bereichen der Produktion bewährt. Seit einiger Zeit wird ihr Einsatz für Zerspanungsaufgaben wie dem Fräsen von Bauteilen immer interessanter. Roboter bieten aufgrund ihrer relativ niedrigen Anschaffungskosten, ihrem großen Arbeitsraum und ihrer universellen Verwendbarkeit einige Vorteile im Vergleich zu den üblicherweise eingesetzten Werkzeugmaschinen. Allerdings ist es momentan noch nicht möglich harte Werkstoffe mit derselben Genauigkeit bearbeiten zu lassen. Beim Fräsen entstehen sehr hohe externe Prozesskräfte, die eine Rückwirkung auf die Roboterstruktur ausüben. Der Roboter, der sich teilweise elastisch verhält, was sich vor allem in seinen Gelenken bemerkbar macht, weicht unter dem Einfluss dieser Kräfte von seiner vorgegebenen Sollbahn ab. Dies führt bei der Bearbeitung des Bauteils zu Ungenauigkeiten. Ziel dieser Arbeit ist es, die Pfadabweichung zu minimieren. Dabei soll ein allgemein gültiger Ansatz entwickelt werden, der unabhängig vom verwendeten Robotertyp und den verwendeten Prozessparametern ist und keine zusätzliche Ausstattung benötigt. Die Optimale Steuerung ist ein geeignetes Werkzeug um diese Problemstellung zu bearbeiten. Somit ist das verwendete Robotermodell flexibel austauschbar und beliebige Pfadvorgaben werden ermöglicht. Weitere Nebenbedingungen und Sytembeschränkungen können einfach hinzugefügt werden. Damit bleibt der Ansatz für eine Vielzahl an Szenarien mit unterschiedlichen Robotertypen, Werkzeugparametern und Werkstoffeigenschaften einsetzbar. Aufgrund der Fortschritte im Bereich der Optimierung und dank moderner numerischer Lösungsalgorithmen sind Optimalsteuerungsprobleme heute auf herkömmlichen Computern effizient berechenbar.

Abstract

Nowadays industrial robots are successfully being used in many domains of manufacturing and production. Recently their utilization for cutting applications like the milling of parts gets more and more into the focus of scientists and engineers. Robots have several essential advantages in comparison to common machinery tools. Their aquisition costs are low, they have a very large work space and can also be used for various other tasks. However robots are currently not able to mill hard materials with competitive accuracy. High external forces emerge during the milling process and take effect on the robot structure. Industrial robots possess a certain elastic behavior, especially their joints. This causes the robot to drift away from the desired path. The result is the production of less accurate parts. The goal of this work is to compensate the derivation during the path tracking in milling applications. The resulting method should be gernerally applicable, so that it does not depend on a specific robot model or an unique set of parameters. Additional hardware, which would prevent an universal adoption of the method, is not allowed. Optimal Control offers a convinient tool to solve such kinds of problems. In this way the robot dynamics model can be replaced and arbitrary path can be provided. Additional constraints and system restrictions can be added easily. So the developed approach stays adaptable for a broad range of scenarios with differing robot types, tool parameters and materials. Because of the advances in the field of optimization and with the help of modern numerical algorithms, Optimal Control Problems can efficiently being solved on today's common computer hardware.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Roboter haben die industrielle Fertigung von Produkten und Konsumgütern maßgeblich beeinflusst. Sie haben eine Vielzahl an verschiedenen Aufgaben übernommen und sind aus modernen Fabriken und Produktionsstätten nicht mehr wegzudenken. Roboter helfen dabei Abläufe zu automatisieren, die Qualität sowie Präzision der Fertigung zu erhöhen und sie effizienter zu machen. Hauptsächlich werden Industrieroboter als Handhabungsgeräte, z.B. zum Palettieren, Verpacken, Sortieren oder Montieren eingesetzt, als Schweißroboter zum Lichtbogen-, Punkt- oder Laserschweißen, als Messroboter in der Qualitätssicherung oder als Lackierroboter. Oft wird behauptet, dass diese Maschinen Arbeitsplätze wegrationalisieren würden. Aber es ist vielmehr der Fall, dass Industrieroboter dabei helfen, Produktionsstandorte in einem Hochlohnland wie Deutschland zu erhalten, da die Effizienzsteigerung dafür sorgt, konkurrenzfähig gegenüber Niedriglohnländern zu bleiben. Außerdem sind viele Produktionsschritte ohne Roboter kaum noch durchführbar, weil manche Aufgaben von Menschen nur ungenau oder zu langsam ausgeführt werden können oder nicht praktikabel sind. Zusammgefasst haben Industrieroboter folgende Vorteile:

- Umgang mit hohen Traglasten
- präzise Positionierung des angeschlossenen Werkzeugs oder der Nutzlast
- hohe Arbeitsgeschwindigkeit
- in gefährlichen Umgebungen einsetzbar
- keine Ermüdung und gleichbleibende Arbeitsqualität

Deshalb ist es ein Bestreben, diesen universell einsetzbaren Maschinen möglichst viele Anwendungsbereiche zu erschließen. Aus diesem Hintergrund beschäftigt sich die vorliegende Masterarbeit mit Industrierobotern, die zum Fräsen von Bauteilen eingesetzt werden sollen. In diesem Anwendungsgebiet sind Industrieroboter noch weitgehend ausgeschlossen. Zur zerspanenden Bearbeitung von komplexen Bauteilen mit hohen Genauigkeitsanforderungen sind heute Fünf-Achs-CNC-Fräsen Stand der Technik. Die Anschaffung solcher Werkzeugmaschinen ist jedoch meistens sehr kostspielig. Häufig ist zur Bedienung und Wartung speziell geschultes Personal notwendig. Außerdem lassen sich diese Maschinen nur für ein spezielles Fertigungverfahren einsetzen. Der Bearbeitungsraum ist sehr beschränkt, sodass nur relativ kleine Werkstücke gefräst werden können. Maschinen mit größerem Volumen ziehen wiederum vielfach höhere Kosten nach sich. Im Vergleich dazu bieten Industrieroboter ein sehr gutes Verhältnis zwischen Anschaffungskosten und Größe des Arbeitsraums. Sie sind universell einsetzbar und können zu einem anderen Zeitpunkt auch weitere Aufgaben übernehmen. Dies erhöht die betriebswirtschaftliche Flexibilität, was vor allem für mittelständische Unternehmen eine Rolle spielt. Fräsende Industrieroboter könnten den Unternehmen helfen, die Fertigung von Kleinserien und Prototypen durch Drittanbieter zu vermeiden und somit Entwicklungszyklen von Produkten zu verkürzen. Vor Wettbewerbern mit einem Produkt in die Marktphase einzutreten, ist heute ein entscheidender Faktor für den wirtschaftlichen Erfolg vieler Firmen. Allerdings sind Industrieroboter im Vergleich zu CNC-Fräsen bei der Bearbeitung von harten Werkstoffen wie Metallen noch zu ungenau. Trotz der starren Konstruktion dieser Roboter besitzen sie vor allem in den Gelenken eine gewisse Elastizität. Bei der Bearbeitung von harten Werkstoffen entstehen sehr hohe externe Prozesskräfte, die eine Rückwirkung auf den Roboterarm ausüben. Aufgrund seiner Nachgiebigkeit weicht der Arm vom gewünschten Pfad ab. Somit wird die Bearbeitung des Bauteils zu unpräzise. Ziel dieser Arbeit ist es, ausgehend von einem ausreichend genauem Roboterdynamikmodell und einer Beschreibung der auftretenden externen Fräskräfte, ein Verfahren zu

entwickeln, das die Pfadabdrängung des Roboters kompensiert, indem die Motormomentenverläufe angepasst werden, die den Roboter auf seiner Solltrajektorie halten sollen. Die dazu notwendigen Momentenverläufe bzw. die dazugehörige kompensierende Werkzeugtrajektorie werden dabei im Voraus über Methoden der Optimalen Steuerung [1] berechnet. Dazu ist es notwendig, einem speziellen numerischen Lösungsalgorithmus eine korrekte Formulierung des mathematischen Optimalsteuerungsproblems zu übergeben. Das erarbeitete Verfahren soll möglichst flexibel und adaptiv gehalten werden, was eine spätere Änderung und Anpassung der einzelnen Komponenten ermöglicht. Das Roboterdynamikmodell, die Fräskraftberechnung sowie die Pfadgenerierung sind getrennte, unabhängige Komponenten, sodass das Problem nicht über eine exakte und detailreiche Modellierung des Gesamtprozesses gelöst werden kann. Vielmehr soll ein allgemeingültiges Verfahren auf Basis der Optimalen Steuerung gefunden werden, das es ermöglicht eine unabhängige fehlerminimierende Steuerungsstrategie zu finden. Dadurch wird gewährleistet, dass prinzipiell verschiedene Zerspan- und Fräsprozesse von unterschiedlichen Industrierobotertypen entlang beliebiger definierbarer Trajektorien durchgeführt werden können.

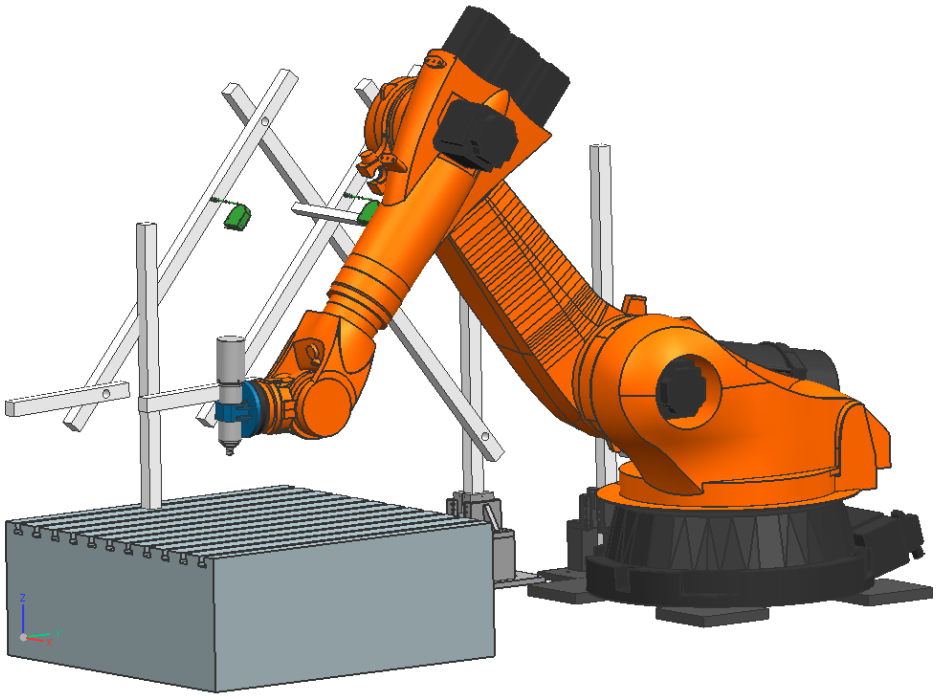


Abbildung 1.1: CAD-Modell des Versuchsaufbaus des fräsenden Industrieroboters

1.2 Aufbau dieser Arbeit

Im Folgenden wird die Struktur dieser Arbeit zusammengefasst.

Kapitel 2
Dieses Kapitel beschäftigt sich mit dem aktuellen Stand der Technik und dazu verwandten Arbeiten aus der Forschung. Es werden hierbei die drei Hauptthemengebiete „Optimale Steuerung“, „Pfadverfolgung und Fräsen mit Industrierobotern“ sowie „Fertigungsprozess Fräsen und Methoden der Produktentwicklung“, die für diese Arbeit ausschlaggebend sind, beleuchtet.

[1] engl.: Optimal Control

Kapitel 3
In Kapitel 3 werden die Grundlagen zur Dynamik und Modellierung von Systemen sowie speziell von Industrierobotern dargestellt, welche später für die Formulierung des Optimalsteuerungsproblems benötigt werden. Zunächst wird sich der Dynamik über die Kinematik von Standard-Manipulatorarmen genähert. Es werden ausgewählte Beispielsysteme gezeigt, die während dieser Arbeit entwickelt und untersucht wurden. Die Herleitung des Lagrange-Formalismus bietet bereits in diesem Kapitel einen Einblick in die dynamische Optimierung und zeigt, wie diese über das „Prinzip der kleinsten Wirkung“ und der Variationsrechnung mit der Dynamik von Systemen zusammenhängen. Des Weiteren wird kurz die numerische Lösung der entstandenen gewöhnlichen Differentialgleichungen erläutert, da einige Optimierungsverfahren die Systemdynamik fortlaufend nach der Zeit integrieren.

Kapitel 4
Hier werden Grundlagen zur nichtlinearen kontinuierlichen Optimierung beschrieben. Zunächst werden statische Optimierungsprobleme (NLP Probleme) und deren Lösungsmethoden vorgestellt. Das in dieser Arbeit zu lösende Problem hat jedoch die Form eines dynamischen Optimierungsproblems, nämlich die eines Optimalsteuerungsproblems. Einige Lösungsalgorithmen (direkte Verfahren) übersetzen diesen Typ in ein NLP Problem, weswegen anfangs auch auf deren Lösung eingegangen wird. Optimalsteuerungsprobleme besitzen Nebenbedingungen in Form von Differentialgleichungen. Die Struktur eines solchen Problems wird erörtert und die notwendigen mathematischen Bedingungen an seine Lösung werden aufgezeigt. Die unterschiedlichen Klassen von Lösungsalgorithmen werden vorgestellt und deren Funkionsweisen skizziert.

Kapitel 5
Dieses Kapitel behandelt die Formulierung des Optimalsteuerungsproblems des zum Fräsen eingesetzten Industrieroboters. Die für den Lösungsansatz verwendeten Werkzeuge werden vorgestellt und deren Benutzung skizziert. Die einzelnen Nebenbedingungen, Restriktionen sowie das Zielkriterium werden beschrieben und mögliche Alternativen erläutert. Die resultierenden Lösungen werden anhand von Beispielen dargestellt. Auch die Möglichkeit von Optimalsteuerungsproblemen mit mehreren Phasen wird untersucht.

Kapitel 6
In diesem Kapitel werden die erarbeiteten Resultate beurteilt. Die Bewertungskriterien werden begründet und auf die Lösungen des Kompensationsansatzes angewendet.

Kapitel 7
Am Ende dieser Arbeit erfolgt eine kurze Zusammenfassung. Vor- und Nachteile der durchgeführten Methoden werden diskutiert. Die Schwierigkeiten und Probleme, die während der Arbeit auftraten werden verdeutlicht und daraufhin mögliche zukünftige Verbesserungen und Erweiterungen aufgezeigt.

2 Stand der Forschung

2.1 Optimale Steuerung

2.1.1 Historische Entwicklung

Optimalsteuerungsprobleme sind dynamische Optimierungsprobleme (Variationsprobleme), bei denen gewöhnliche Differentialgleichungen als Nebenbedingungen auftreten. Dynamische Optimierungsprobleme gab es bereits sehr früh in der Geschichte. Das wohl älteste ist das *Problem der Dido* [1], welches ca. 825 vor Christus auftrat und mit der Gründung von Kathargo einherging. Es sollte die optimale Form eines Landstückes gefunden werden, das nur durch ein Seil aus Ochsenhaut und der Küstenlinie eingegrenzt war. Dazu musste also die Form einer begrenzten Strecke gefunden werden, die eine Fläche mit möglichst großem Inhalt umschließen sollte. Die Lösung war bekanntermaßen ein Kreis, was aber erst im 19. Jahrhundert formell bewiesen werden konnte. Nachdem 1669 Johann Bernoulli das so genannte *Brachistochroneproblem* veröffentlichte, begann die Entwicklung allgemeiner Lösungsmethoden für Variationsprobleme solchen Typs (s. Abschnitt 4.2.1). In dieser Fragestellung rollt ein Massepunkt entlang einer Bahn von Punkt A zu einem tiefer gelegenen Punkt B. Die optimale Bahn soll nun gefunden werden, sodass dabei möglichst wenig Zeit vergeht.

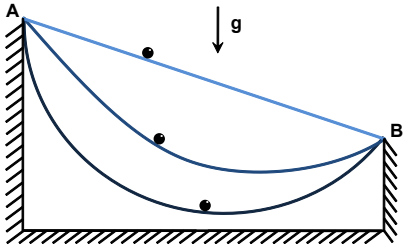


Abbildung 2.1: Eine Punktmasse, die von A zu B zeitminimal hinabrollen soll

Neben vielen anderen Einflüssen geht die heutige Optimale Steuerung im Bereich der Technik vor allem auf Pontryagin zurück, der das so genannte *Maximum Prinzip* [2], [3] verfasste. Aufgrund von komplizierten Systemdynamiken mit hoher Nichtlinearität und Dimensionalität sowie vieler komplexer Nebenbedingungen sind in nahezu allen realitätsnahen Anwendungsgebieten nur noch numerische Lösungsmethoden anwendbar. Es ist somit auch dem Einzug von billiger und schneller Computertechnologie in den letzten Dekaden zu verdanken, dass die Optimale Steuerung eine so große Bedeutung erlangt hat.
Zu Beginn dieser Entwicklung wurde *Dynamische Programmierung* [1] auf Probleme der Optimalen Steuerung angewandt. Durch Diskretisierung der *Hamilton-Jacobi-Bellmann-Gleichung* [2], [1], die den Zusammenhang zwischen der optimalen Steuerung $\boldsymbol{u}^*$ und den Startwerten $(\boldsymbol{x}(t_0), t_0)$ beschreibt, wurde versucht ein Pfad vom Zielzeitpunkt t_f zum Anfang t_0 zu finden. Jedoch steigt der Rechenaufwand dieser Methode sehr stark mit der Dimensionalität des Zustandsraums von $\boldsymbol{x}$, weshalb sie für die meisten Anwendungsgebiete völlig unpraktikabel ist.
Darauf folgend wurden *Gradientenverfahren* [1] immer populärer. Dabei wird von einer Startschätzung für den Systemzustand $\boldsymbol{x}(t)$ sowie der so genannten adjungierten Variablen $\boldsymbol{\lambda}(t)$ ausgegangen und damit das Minimum der Hamiltonfunktion (s. Formel (4.51)) an diskreten Zeitpunkten für $0 \leq t_i \leq t_f$ mit einem NLP Gradientenverfahren (s. Abschnitt 4.1.1) berechnet. Mit der somit erhaltenen Näherung für $\boldsymbol{u}(t)$ berechnet man einerseits durch Vorwärtsintegration der Systemdynamik $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$

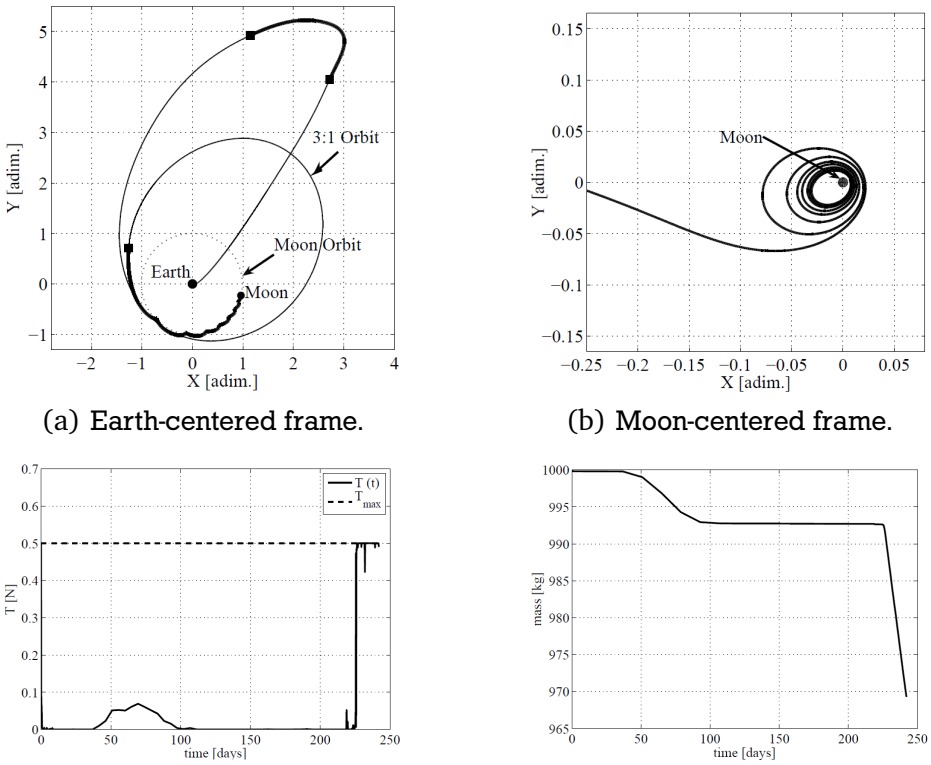
mit Randwert $\boldsymbol{x}(0) = \boldsymbol{x}_0$ und andererseits durch Rückwärtsintegration der adjungierten Differentialgleichung $\dot{\boldsymbol{\lambda}}(t) = -\frac{\partial H}{\partial \boldsymbol{x}}$ mit Randwert $\lambda_i(t_f) = \frac{\partial \phi(\boldsymbol{x}(t_f), t_f)}{\partial x_i(tf)}$. Durch numerische Integration (s. Abschnitt 3.4) dieser beiden Differentialgleichungen erhält man neue Näherungen für $\boldsymbol{x}(t)$ und $\boldsymbol{\lambda}(t)$ und die Iteration kann von vorne beginnen, bis das Abbruchkriterium erfüllt wird. Aktuelle Lösungsalgorithmen können im Wesentlichen in zwei unterschiedliche Verfahrensklassen eingeteilt werden, den indirekten und direkten Lösungsmethoden [1].
Indirekte Verfahren greifen in expliziter Weise auf die notwendigen Bedingungen aus der Optimalsteuerungstheorie (s. Abschnitt 4.2.2) zurück. Diese Klasse setzt eine große Erfahrung des Benutzers und eine sehr gute Kenntnis über die Problemstruktur voraus um das entstehende Mehrpunkt-Randwertproblem zu lösen. Hierbei müssen sehr schwer zu ermittelnde Startschätzungen für die adjungierten Variablen vorliegen und eine möglichst genaue Beschreibung der Schaltstruktur, die angibt zu welchen Zeitpunkten welche Restriktionen aktiv sind. Das indirekte *Mehrfachschießverfahren* [1] oder indirekte *Kollokationsverfahren* sind bekannte Vertreter dieser Klasse. Indirekte Verfahren liefern sehr exakte Lösungen und sind deshalb für Anwendungsgebiete mit hohen Genauigkeitsanforderungen interessant. Das immer populärer werdende automatische Differenzieren [4], [5] kann dazu eingesetzt werden um das analytische Differenzieren zu umgehen. So kann z.B. die komplexe rechte Seite der Systemdynamik, vorliegend als effizienter Programmcode, direkt abgeleitet werden. Sie muss dazu nicht erst formelmäßig beschrieben werden um dann die resultierenden Gleichungen mit analytischen Methoden zu differenzieren. Das kann die Herleitung einiger notwendiger Bedingungen erleichtern und die Rechenzeit verkürzen.
Direkte Methoden überführen das unendlichdimensionale dynamische Optimierungsproblem mittels Diskretisierung in ein endlichdimensionales nichtlineares Optimierungsproblem (NLP Problem, vgl. Gleichungen (4.7), (4.8), (4.9)). Es können entweder nur die Steuervariablen oder die Steuervariablen und der Systemzustand transformiert werden. Das resultierende NLP Problem kann dann mittels Methoden aus der statischen Optimierung gelöst werden (s. Abschnitt 4.1). Hierbei werden fast ausschließlich SQP (s. Absatz in 4.1.1) oder Innere-Punkt-Verfahren (s. Absatz in 4.1.1) eingesetzt, da diese besonders geeignet sind sehr hochdimensionale NLP Probleme mit vielen Nebenbedingungen zu lösen. Direkte Methoden erfordern nur geringe Vorkenntnisse im Bereich der Optimalsteuerungtheorie und sind dank moderner NLP Lösungsverfahren sehr effizient. Sie sind robuster bezüglich schlechter Startwerte und benötigen keine Anfangschätzung der adjungierten Variablen oder der Schaltstruktur. Deshalb zählen direkte *Kollokations-* und *Mehrfachschießverfahren* zu den am häufigsten eingesetzten.
Hybride Ansätze [6] nutzen sowohl direkte als auch indirekte Methoden. Hierbei können die leicht anzuwendenden direkten Verfahren dazu genutzt werden um das Problem in einem ersten Schritt zu lösen. Dadurch ist nun die Schaltstruktur des Problems bekannt und die zur Lösung dazugehörigen adjungierten Variablen können ermittelt werden. Des Weiteren können die berechneten Verläufe für Steuerung und Systemzustand als Anfangswerte für ein indirektes Verfahren dienen. Somit kann aus den Ergebnissen des direkten Verfahrens eine hochgenaue Lösung mittels einem indirekten Verfahren berechnet werden.

2.1.2 Aktuelle Ansätze und Anwendungsgebiete

Optimale Steuerung wird in vielen Bereichen eingesetzt und hat die Umsetzung von Lösungen zu Problemen aus vielen verschiedenen Gebieten ermöglicht. Die Luft- und Raumfahrttechnik war eines der ersten Anwendungsgebiete, in dem Optimale Steuerung große Erfolge erzielen konnte. Die Optimierung von Aufstiegsbahnen von Raketen und deren Schubsteuerung um Nutzlasten in den Orbit zu bringen wird

[1] auch Mehrzielmethode, engl.: multiple shooting

über die Lösung von Verallgemeinerungen des *Goddard's Rocket Maximum Ascent Problems* bestimmt [7], [8]. Die Bestimmung optimaler Satelliten- und Raumsondentrajektorien ist ein weiteres Beispiel aus diesem Bereich. Diese müssen so gewählt werden, dass die gewünschten Missionsziele erreicht werden können. Bei den dazu nötigen Manövern soll der notwendige Treibstoffverbrauch (bzw. das Δv-Budget) minimiert werden. Je nach Antriebskonzept (z.B.: chemisch [2] oder elektrisch [3]) können sich dabei völlig unterschiedliche Flugbahnen [9], [10] ergeben.



(a) Earth-centered frame.

(b) Moon-centered frame.

(c) Thrust profile.

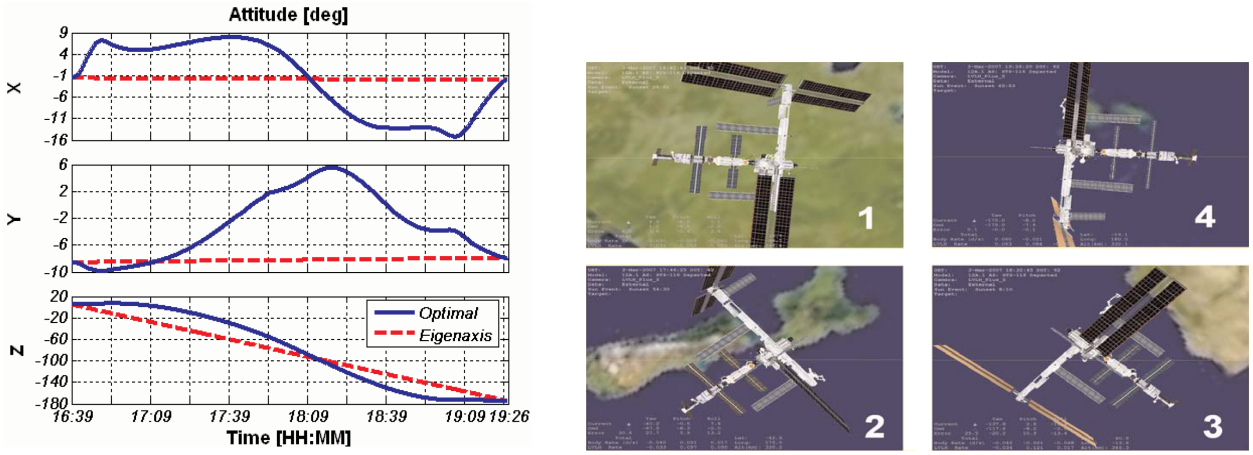
(d) Mass consumption.

Abbildung 2.2: Lösung eines Optimalsteuerungsproblems für eine „low-thrust transfer" Trajektorie vom Erd- zum Mondorbit (aus [10]).

Die Lagekontrolle [4] von Satelliten spielt in diesem Anwendungsfeld eine weitere wichtige Rolle. Beispielsweise müssen zu bestimmten Zeiten die Solarkollektoren zur Sonne ausgerichtet werden um die Energieversorgung sicherzustellen oder die Antennen zur Erde zeigen um Steuerbefehle zu erhalten und Daten zu versenden. Dazu ist eine Lageänderung von einer Ausgangsorientierung R_A zu einer Zielorientierung R_B des Vehikels nötig. Das kann über eine Rotation um die Eigenachse, die für jedes Paar R_A und R_B existiert, erreicht werden. Statt Schubdüsen können dazu auch spezielle interne Schwungräder bzw. Gyroskope verwendet werden. Sind sie in der Lage einen Drehimpuls in adäquater Größe zu erzeugen, kann ein solches Manöver durchgeführt werden ohne wertvollen Treibstoff zu verbrauchen. In 2006 wurde ein so genanntes *Zero Propellant Maneuver* (ZPM) für die Internationale Raumstation ISS durchgeführt, bei der eine Lageänderung von 180° vorgenommen wurde [11]. Die internen *Control Moment Gyroscopes* (CMGs) waren allerdings nicht im Stande die massereiche ISS direkt um die gewünschte

2 impulsiv: hoher Schub T, niedriger *spezifischer Impuls* I_{sp}
3 low-thrust: niedriger Schub T, hoher *spezifischer Impuls* I_{sp}
4 engl.: attitude control

Eigenachse zu rotieren, da sie nicht ausreichend hohe Drehmomente erzeugen konnten. Mit Hilfe der Optimalen Steuerung wurde ein dynamischer Verlauf der Rotationsachse gefunden, sodass die Orientierung geändert werden konnte ohne die maximale Drehmomentskapazität der CMGs zu überschreiten [12], [13]. Bei diesem ca. dreistündigen Manöver sparte die NASA Schätzungen zur Folge 1$ Mio. US-Dollar an Treibstoffkosten [14]. Die numerische Berechnung des Rotationsachsenverlaufs wurde mit dem Programm *DIDO* [15] durchgeführt, welches auf einer *pseudospektralen direkten Kollokationsmethode* beruht. Das in dieser Arbeit verwendete Programm *PSOPT* [7] benutzt ebenfalls diese in der Praxis bereits bewährte Lösungsmethode. Sie bietet theoretisch exponentielle Konvergenzraten, was nicht mit Verfahren basierend auf stückweiser Approximation mit Polynomansätzen erreicht werden kann. Selbst bei relativ wenig Diskretisierungspunkten liefern pseudospektrale Methoden gute Ergebnisse, da sie global auf dem ganzen Zeitintervall $[t_0, t_f]$ arbeiten.



(a) Resultierende Rotationsachse (b) ISS Lageänderung

Abbildung 2.3: ZPM-Rotationsachsenverlauf und Visualisierung der ISS Orientierung (aus [14])

In der Luftfahrt wurde Optimale Steuerung dazu eingesetzt das Verhalten einer Boeing 727 bei gefährlichen Fallwinden zu verbessern [16], [6]. Dabei wurde eine *Minimax*-Formulierung der Zielfunktion gewählt, die die minimale Höhe des Flugzeuges während des Verlaufs durch die Fallwindzone maximieren soll, um möglichst viel Sicherheitsspielraum zu gewährleisten. In der Verfahrenstechnik wird Optimale Steuerung dazu eingesetzt effizient Stoffe zu synthetisieren, indem der optimale Verlauf des Mischungsverhältnisses der Ausgangschemikalien berechnet wird [16], [6]. Zur Parameteridentifizierung von dynamischen Systemen kommen typischer Weise indirekte Verfahren [17] zum Einsatz. Allerdings können auch direkte Verfahren [16] solche Problemstellung bearbeiten, falls die Genauigkeitsanforderungen nicht zu hoch sind. Die Parameter von Mehrköpersystemen wie Roboter oder Fahrzeuge wurden bereits mit Mitteln der Optimalen Steuerung identifiziert [18]. In der Finanz- und Betriebswirtschaft werden sie eingesetzt um Gewinne bei Verkäufen zu maximieren oder Kosten, die z.B. durch Lagerhaltung entstehen, zu minimieren [2], [19]. Die Robotik ist ein weiteres großes Anwendungsgebiet der Optimalen Steuerung, in dessen Rahmen sich auch die vorliegende Arbeit bewegt. Zunächst konzentrierte sich das Interesse auf die Optimierung von Punkt-zu-Punkt-Bahnen von Industrierobotern. Einerseits sollte der Zeitbedarf dieser Trajektorien minimiert werden, damit die Produktion in Fabriken erhöht werden kann, andererseits sollte der Energieverbrauch und die Belastung der Servomotoren minimiert werden [20]. Diese Methoden wurden unter anderem für den Manutec R3 Industrieroboter erprobt und umgesetzt [6], [21], [22], [23]. In den beiden folgenden Abbildungen ist die zeitminimale (Abb. 2.4) und energieminimale (Abb. 2.5) Trajektorie zwischen jeweils denselben zwei Punkten illustiert.

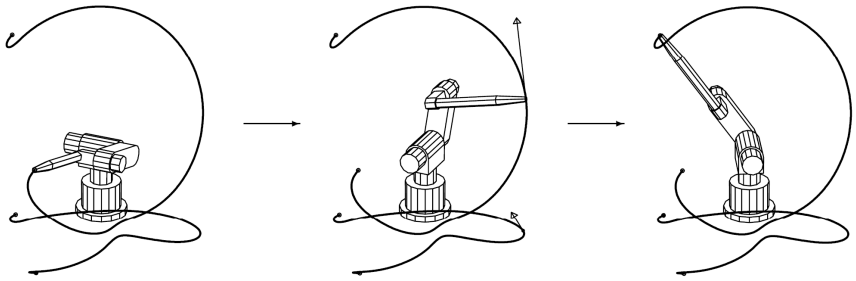


Abbildung 2.4: Zeitminimale Punkt-zu-Punkt-Bahn für den Manutec R3 Roboterarm (aus [6])

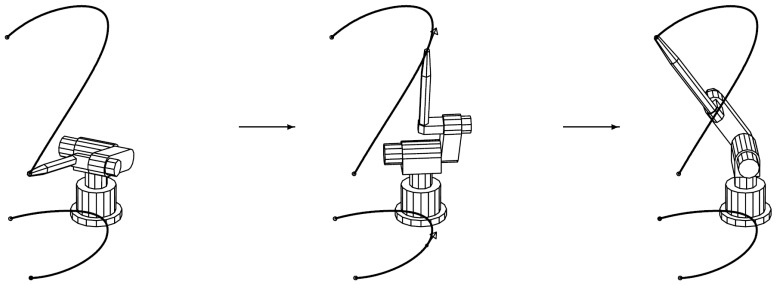


Abbildung 2.5: Energieminimale Punkt-zu-Punkt-Bahn für den Manutec R3 Roboterarm (aus [6])

Nicht nur die Optimierung von Punkt-zu-Punkt-Bahnen werden untersucht, sondern auch die Optimierung ganzer Pfade, was im nächsten Abschnitt näher beleuchtet wird. Für mobile Roboter werden Trajektorien durch Terrains ermittelt, die Hindernisse effizient vermeiden [7]. Für Gruppen kooperierender mobiler Roboter können Steuerungen ermittelt werden um vordefinierte Missionziele wie die Erkundung oder Überwachung eines bestimmten Gebiets optimal durchzuführen [24].

2.2 Pfadverfolgung und Fräsen mit Industrierobotern

Wie am Ende des letzten Abschnitts erwähnt, zählt die Optimierung von Trajektorien für Manipulatorarme zu den wichtigsten Standardaufgaben in der Robotik. Allerdings ist es nicht immer erwünscht eine freie Bahn zwischen zwei Punkten zu finden, die einem bestimmten Optimierungskriterium genügt. Häufig gibt es auch Restriktionen oder Bedingungen, die dabei zusätzlich beachtet werden müssen. Beispielsweise sind Hindernisse oder Selbstkollisionen im Arbeitsraum mit in Betracht zu ziehen [25]. In vielen Fällen ist jedoch bereits die komplette Endeffektorbahn vorgegeben. Bestimmte Aufgaben wie das Anbringen von Schweißnähten oder das Lackieren von Oberflächen können so durchgeführt werden. Dann ist es nötig einen Pfad vom Roboter abfahren zu lassen, also ein so genanntes Path Tracking anzuwenden. In [26] wird ein Verfahren beschrieben, das es ermöglicht Winkel- und Winkelgeschwindigkeitsbeschränkungen der Gelenke sowie Drehmomentbeschränkungen der Motoren zu verarbeiten um mit einem starren Robotermodell eine Trajektorie zeitminimal abzufahren. In [27] wird ein effizienteres Verfahren vorgestellt, welches außerdem viskose Dämpfung im Roboterdynamikmodell zulässt. [26] und [27] beschränken sich jeweils auf ein zeitoptimales Abfahren der Trajektorie. [28] und [29] berücksichtigen hingegen einen Kompromiss zwischen Zeit- und Energieoptimalität, der sich über einen Gewichtungsfaktor γ einstellen lässt. In beiden Arbeiten wird der Sollpfad in Gelenkwinkelkoordinaten mit Hilfe einer Variablen s (Pseudotime) parametrisiert. Die Variable $s(t)$ kann normiert werden, sodass zu Beginn $s(0) = 0$ und am Ende $s(t_f) = 1$ gilt. Die Gelenkwinkel $\boldsymbol{q}(s)$ sind nun nicht mehr direkt von t abhängig. Durch Ableiten mit Kettenregel können die Gelenkgrößen $\dot{\boldsymbol{q}}(s)$ und $\ddot{\boldsymbol{q}}(s)$ bestimmt werden,

welche nun in der Roboterdynamikgleichung (3.51) substituiert werden. Roboterdynamik und alle weiteren Neben- und Randbedingungen hängen nur noch von s und $\dot{s}$ ab, was den Zustandsraum auf zwei Dimensionen reduziert. Es kann gezeigt werden, dass die dabei entstehende Formulierung des Optimalsteuerungsproblems konvex ist [28]. [29] übersetzt das resultierende Optimalsteuerungsproblem mittels einer direkten Methode in ein konvexes (vgl. Konvexitätsbedingungen (4.10)-(4.13)) SOCP [5] [30] statt wie üblich in ein allgemeines NLP Problem. Diese lassen sich effizient berechnen und garantieren die global beste Lösung als Ergebnis. Die hier bisher beschriebenen Ansätze sind jedoch nur bedingt für diese Arbeit anwendbar. Sie benutzen jeweils nur Standardroboterdynamiken und haben bisher nicht gelenkelastische Robotermodelle untersucht. Der gravierendste Unterschied ist, dass bisher keine großen extern einwirkenden Kräfte am Endeffektor berücksichtigt wurden. Die meisten Arbeiten beschränken sich auf zeit- und/oder energieminimale Pfadverfolgung. Zielkriterien der Optimalsteuerung, die die Genauigkeit aufgrund von extern hervorgerufenen Abdrängungeffekten berücksichtigen, existieren nach aktuellem Kenntnisstand nicht. So beschäftigt sich [31] zwar mit dem fehlerminimalen Abfahren von Trajektorien mit einem flexiblen Roboterarm, ist aber nicht allgemeingültig anwendbar, da versucht wird den Fehler durch eine möglichst genaue Modellierung des Roboters und einer darauf basierenden Kompensationsstrategie zu reduzieren. Externe Prozesskräfte, die vorher nicht bekannt und somit nicht in die Modellierung miteinzubeziehen sind, können nicht berücksichtigt werden.
Industrierroboter werden mehr und mehr für die zerspanende Bearbeitung interessant. Hier werden ebenso komplette Endeffektortrajektorien vorgegeben, die der Manipulatorarm verfolgen muss. Dank niedrigeren Investitionskosten und dem größeren Bearbeitungsraum der Roboter gegenüber Werkzeugmaschinen werden sie bereits zum Entgraten von Gussbauteilen oder der Vorverarbeitung von Bauteilen eingesetzt [32]. Bei diesen Arbeiten entstehen nur kleine bis mittlere externe Prozesskräfte oder es genügt eine relativ niedrige Genauigkeit bei der Durchführung. Die Idee Industrieroboter in der Zerspanung einzusetzen reicht bereits bis in die 1980er Jahre zurück. In [33] wurde die Gussnachbearbeitung mit Industrierobotern untersucht. Der Einsatz von nachgiebigen Werkzeugen [34], [35] wurde populär, weil dadurch die Rückwirkung auf die Roboterstruktur verringert werden konnte. Trotzdem wurde die Genauigkeit stark beeinflusst. Um dem entgegen zu wirken, wurden vor allem der Einsatz von zusätzlicher Sensorik und aufwändigen Regelungsverfahren angewandt [36], [37], [38]. Jedoch sind mit diesen Onlineverfahren nur relativ langsame Schnittgeschwindigkeiten möglich. Zudem sind die Verfahren nicht sehr robust, was auf ungenauen Sensordaten und trägen Reaktionszeiten des Systems beruht. Außerdem können herumfliegende Späne die Messungen der Sensoren beeinträchtigen. Im Allgemeinen machen die zusätzlichen Anschaffungskosten der Sensorik, der hohe Kalibrierungsaufwand und der um ein Vielfaches höher liegende Rechenaufwand diese Feedback-Strategien unpraktikabel. Deshalb ist eine exakte Offlinefehlerkompensation wünschenswert. Bisher sind Industrieroboter in Bezug auf die Herstellung von Bauteilen aus Materialien höherer Festigkeit (z.B. Aluminium) mit hohen Genauigkeitsanforderungen und besserer Oberflächenqualität gegenüber den viel steiferen Werkzeugmaschinen nicht konkurrenzfähig. Die entstehenden externen Prozesskräfte beeinflussen Werkzeugmaschinen nur sehr gering, haben aber einen erheblichen Einfluss auf die teilweise deutlich elastischere Roboterstruktur. Bis heute gibt es keine robuste und genaue Methode zur Kompensation der Pfadabdrängung beim High-Speed-Cutting. Daher beschränken sich aktuell in der Industrie zum Zerspanen eingesetzte Roboter überwiegend auf die Bearbeitung von weicheren Materialien wie Kunststoffen und CFK-Werkstoffen [39].

[5] engl.: Second-Order Cone Program

2.3 Fertigungsprozess Fräsen und Methoden der Produktentwicklung

Bei der Fertigung von Bauteilen wird in der modernen Produktion in der Regel nach Prozessketten gearbeitet. Diese geben Regeln und Vorschriften vor, in welcher Weise Informationen verarbeitet werden müssen, um ein Prozessziel wie z.B. die Herstellung eines Produkts zu erreichen [40]:

> ***Definition - Prozesskette:***
> *Unter einer Prozesskette wird die formale, hierarchisch strukturierte Zusammenfassung von Informationsverarbeitungsprozessen (Erzeugung, Verarbeitung und Austausch von Information), die einem gemeinsamen Prozessziel dienen, verstanden.*

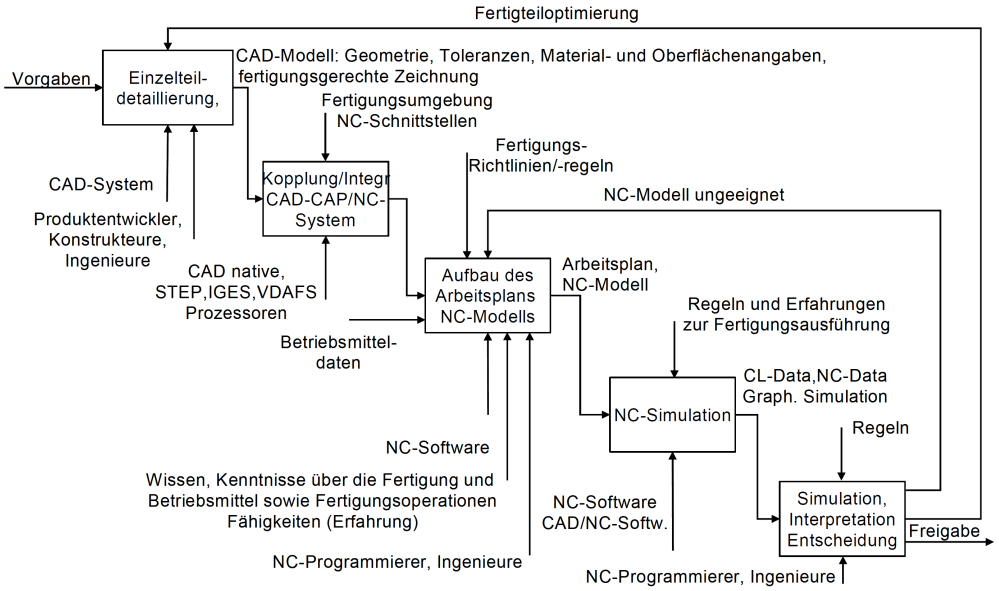
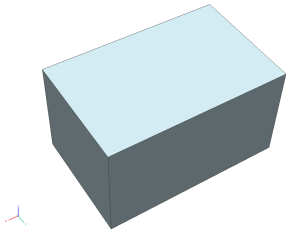


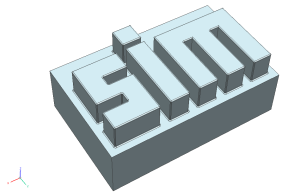
Abbildung 2.6: SADT-Diagramm der Prozesskette CAD-NC (aus [40])

Die Verwendung von 3D-CAD[6]-Programmen [41],[42] nimmt dabei eine zentrale Rolle ein. Das fertige Bauteil wird mit diesen Werkzeugen virtuell am Computer konstruiert, sodass eine exakte geometrische Beschreibung mit einer Vielzahl an weiteren Informationen, wie z.B. Werkstoffeigenschaften, Massenschwerpunkte und Trägheitstensoren vorliegt. Auch das Rohteil, aus dem das Fertigteil später entstehen soll, wird auf diese Weise modelliert. Üblicherweise ist dies im Wesentlichen eine einfache geometrische Grundform (Quader oder Zylinder). Sind allerdings schon andere Fertigungsprozesse vorausgegangen, kann es sich auch um eine komplexere Form handeln. Häufig sind in moderner CAD-Software bereits CAM[7]-Module integriert. Damit ist es möglich Werkzeugdefinitionen wie den Fräskopftyp und Bearbeitungsparameter (Spindeldrehzahl, Vorschubgeschwindigkeit, Eilganggeschwindigkeit) festzulegen und im Anschluss automatisch durch die boolesche Differenz aus Fertig- und Rohteil die vom Werkzeug abzufahrende Trajektorie zu generieren. Diese wird benötigt um aus dem Rohteil das Fertigteil herzustellen. Die Bearbeitung und die Bewegung des Werkzeugs sowie die zeitbezogene Zerspanung kann am Computer simuliert werden, damit Toleranzen überprüft oder mögliche Kollisionen festgestellt werden können. Die dabei in der Realität entstehenden Fräskräfte und dynamischen Effekte werden hierbei nicht berechnet. Abschließend wird der maschinenunabhängige Steuercode im Format CLDATA [42] ausgegeben.

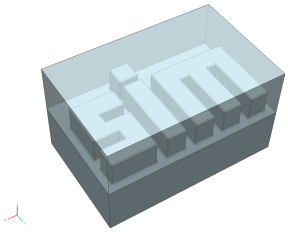
6 engl.: Computer Aided Design
7 engl.: Computer Aided Manufacturing



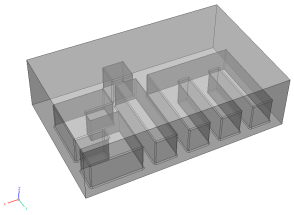
(a) CAD-Rohteil



(b) CAD-Fertigteil



(c) Überlagerung der Geometrien



(d) Abtrag ermittelt aus boolescher Differenz

Abbildung 2.7: Übersicht der CAD-Modelle zum Fertigen eines „SIM Profils“

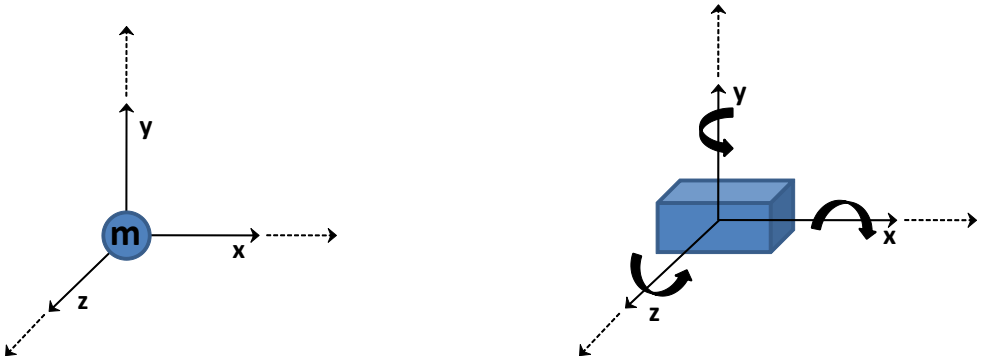
Für das Fräsen von Bauteilen sind CNC-Fräsen Stand der Technik. Diese können das generierte NC-Programm verarbeiten und damit das Bauteil fertigen. Sollen auch Roboter für diese Aufgaben zum Einsatz kommen, ist es notwendig die entstehenden Kräfte entlang des Fräspfads zu kennen um den Wechselwirkungen zwischen der Roboterstruktur und den Prozesskräften angemessen entgegensteuern zu können. Diese auftretenden Kräfte lassen sich mittels unterschiedlicher Methoden beschreiben. Es existieren drei Ansätze zur Zerspankraftberechung: analytische, empirische und mechanistische [43], [44]. Sie hängen im Wesentlichen von technologischen Parametern ab, wie z.B. dem Vorschub, der Drehzahl sowie der Schnitttiefe des Fräswerkzeugs. Die analytischen Ansätze können weiter in Scherebenmodelle [45], Scherzonenmodelle [46], Fließlinienmodelle [47] und energetische Methoden [48] unterteilt werden. Sie haben den Vorteil, dass sie unabhänging zur Prozesskinematik einsetzbar sind. Jedoch gehen sie nur von einer orthogonal verlaufenden Zerspanung aus. Außerdem beruhen sie auf der Annahme einer kontinuierlichen Spanbildung, was allerdings beim High-Speed-Cutting (HSC) durch die teilweise Spansegmentierung nicht vorliegt. Die empirischen Modelle [49] basieren auf Versuchsreihen. Anpassungen der aufgestellten Schnittkraftgleichung können aus Tabellenwerken entnommen werden. Weitere Methoden zur Fräskraftberechnung bilden die mechanistischen Modelle. Neben linearen Ansätzen [50] existieren auch nichtlineare Ansätze [51], die die Zerspankraftanteile in radiale, tangentiale und axiale Richtung bestimmen können. Im Gegensatz zu rein empirischen Modellen kann hier Bezug auf die Werkzeuggeometrie und Reibungseffekte genommen werden. Für komplexere Zerspanvorgänge sind diese Beschreibungen meist nicht ausreichend genau, weswegen auf numerische Verfahren zurückgegriffen werden muss, die die aktuelle Werkstückoberfläche miteinbeziehen. Zum Einen kann dabei das Werkstück als diskretes Modell, wie z.B. als Nagelbrettmodell bzw. Höhenmodell [52], dargestellt werden. Zum Anderen können auch kontinuierliche Darstellungen von Werkzeug und Werkstück über CSG[8]-Modelle [53], [41] realisiert werden. Die numerische Berechnungsarten benötigen aber für komplette Fräsbahnen einen sehr hohen Rechenaufwand.

[8] engl: Constructive Solid Geometry

3 Dynamik und Systemmodellierung

3.1 Kinematik

Um das Verhalten eines Systems zu modellieren, ist die Dynamik von Körpern oder Massepunkten herzuleiten. Dazu ist es zunächst notwendig ihre Kinematik zu kennen. Das heißt, die Lage und Orientierung muss in Abhängigkeit zur Zeit beschrieben werden. Es kann zwischen einer freien und geführten Bewegung unterschieden werden. Ein freier Massepunkt verfügt über drei Freiheitsgrade (DoF [1]). Er kann eine Translation in x-, y- und z-Richtung durchführen. Ein Starrkörper hingegen verfügt über sechs DoF, den translatorischen Freiheitsgraden bezüglich seines Schwerpunktes sowie den drei weiteren rotatorischen Freiheitsgraden. Diese beschreiben jeweils eine Rotation um die x-, y- und z-Hauptachse, welche die Orientierung des Starrkörpers im Raum ändert.



(a) Freiheitsgrade eines Massepunktes

(b) Freiheitsgrade eines Starrkörpers

Abbildung 3.1: Darstellung der Freiheitsgrade für Massepunkt und Starrkörper

Liegt eine geführte Bewegung vor, werden diese Freiheitsgrade eingeschränkt. Wird z.B. ein Massepunkt an einem Pendelseil befestigt, kann dieser nur eine rotatorische Bewegung um den Aufhängepunkt durchführen. Die Anzahl der Freiheitsgrade reduziert sich somit auf einen. Jedoch treten zusätzliche Zwangskräfte auf, die den Massepunkt auf seiner vorgeschriebenen Bahn halten. In diesem Fall sind dies die Seilkräfte $\boldsymbol{S}$ des Pendels.

(a) Pendel mit einem Freiheitsgrad

(b) Freikörperbild mit Zwangskräften

Abbildung 3.2: Geführte Bewegung eines Massepunktes

[1] engl.: Degree of Freedom

3.1.1 Kinematik eines Industrieroboters

Standard-Industrieroboter lassen sich als offene kinematische Ketten beschreiben. Die starren Roboterglieder (links) sind über Gelenke (joints) miteinander verbunden, die die Freiheitsgrade der Glieder einschränken. Üblicherweise werden Drehgelenke, die genau einen rotatorischen Freiheitsgrad zwischen zwei Gliedern zulassen, und Schubgelenke, die genau einen translatorischen Freiheitsgrad zwischen zwei Gliedern zulassen, verwendet. Um nun die relative Lage der einzelnen Roboterglieder und des Endeffektors (TCP [2]) zu beschreiben, werden lokale Koordinatensysteme in den einzelnen Gelenken und im TCP eingeführt. Über Koordinatentransformationen können die Positionen und Orientierungen in jeweils unterschiedlichen Systemen ausgedrückt und ins Inertialsystem umgerechnet werden. In der Robotik treten zwei Probleme von wesentlicher Bedeutung auf. Beim *Vorwärtskinematik-Problem* sind die Gelenkwinkelstellungen des Roboters bekannt und die Endeffektorposition gesucht. Beim *inversen Kinematik-Problem* ist die Endeffektorposition vorgegeben und die nötigen Gelenkwinkelstellungen zum Erreichen dieser Position werden gesucht.

Vorwärtskinematik

Die Vorwärtskinematiktransformation kann mit Hilfe der DH[3]-Parameter θ, d, a, α [54], [55] (s. Appendix A) aufgestellt werden. Die sich ergebende homogene Transformationsmatrix zwischen zwei aufeinander folgenden Gelenken lautet:

$$
\begin{aligned}
{}^{i-1}T_i &= \mathrm{Rot}(z;\theta_i)\cdot\mathrm{Trans}(0,0,d_i)\cdot\mathrm{Trans}(\alpha_i,0,0)\cdot\mathrm{Rot}(x;\alpha_i) && (3.1)\\
&= \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & \alpha_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & \alpha_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} && (3.2)
\end{aligned}
$$

Durch die Matrix-Vektor-Multiplikation eines Punktes in homogenen Koordinaten [55] bezüglich des Koordinatensystems S_i mit der Transformationsmatrix ${}^{i-1}T_i$ ergibt sich die Darstellung des Punktes in S_{i-1} Koordinaten.

$$
{}^{i-1}\boldsymbol{p} = {}^{i-1}T_i \;\cdot\; {}^{i}\boldsymbol{p} \tag{3.3}
$$

Die komplette Vortwärtskinematik ${}^0T_E$ einer kinematischen Kette mit n Gelenken lässt sich dann aus den einzelnen DH-Transformationsmatrizen zusammensetzen.

$$
{}^0T_E(q_1,\ldots,q_n) = {}^0T_1\cdot{}^1T_2\cdot\ldots\cdot{}^{n-2}T_{n-1}\cdot{}^{n-1}T_n\,,\quad \text{mit } q_i = \begin{cases} \theta_i, & \text{falls Gelenk } i \text{ Drehgelenk} \\ d_i, & \text{falls Gelenk } i \text{ Schubgelenk} \end{cases} \tag{3.4}
$$

Werden in diese Transformationsmatrix die Gelenkwinkel θ_i (bzw. Gelenkverschiebung d_i) für $i = 1,\ldots,n$ eingesetzt, erhält man die Position des TCP S_E bezüglich der Koordinaten des Inertialsystems S_0.

Beispiel 3.1. Vorwärtskinematik für einen 3-DoF Roboterarm

Die DH-Tabelle 3.1 des Roboters aus Abbildung 3.3 entstand mit Hilfe des Algorithmus aus [55]. Mit der Transformationsvorschrift (3.1) ergeben sich die folgenden Transformationsmatrizen zwischen je zwei Koordinatensystemen.

[2] engl.: Tool Center Point

[3] Denavit-Hartenberg Parameter

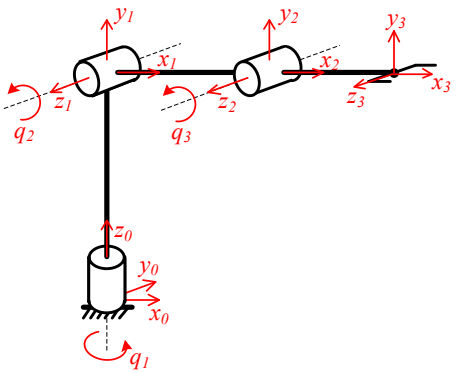


Abbildung 3.3: Schematische Zeichnung eines 3-DoF Roboterarms in Nullstellung

Link	a_i	d_i	α_i	θ_i
1	0	l_1	$\frac{\pi}{2}$	θ_1
2	l_2	0	0	θ_2
3	l_3	0	0	θ_3

Tabelle 3.1: DH-Tabelle des 3-DoF Roboters aus Abbildung 3.3

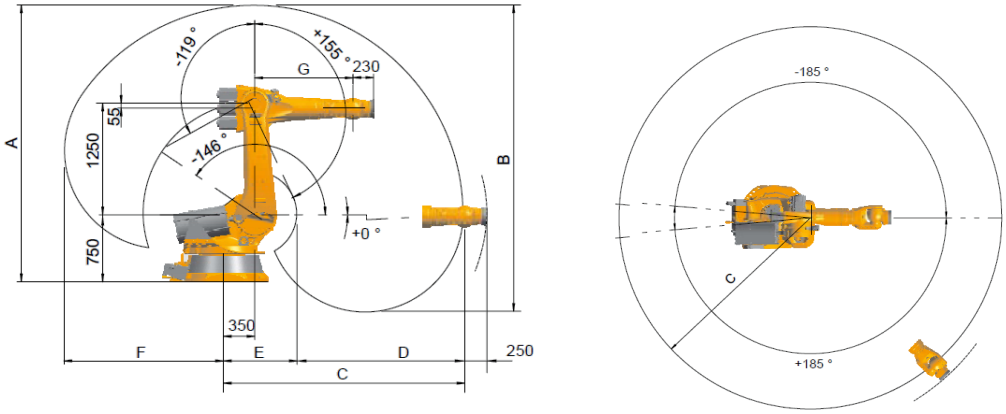
$$ {}^0T_1 = \begin{pmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & -\cos(\theta_1) & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.5} $$

$$ {}^1T_2 = \begin{pmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & l_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & l_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.6} $$

$$ {}^2T_3 = \begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & l_3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & l_3\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.7} $$

Inverse Kinematik

Die inverse Kinematik eines Roboters ist schwieriger zu lösen. Nicht zu allen Endeffektorposen liegen eindeutige Gelenkwinkellösungen vor. Punkte im Inneren des Arbeitsraums können meist über mehrere Gelenkwinkelstellungen erreicht werden. Liegt eine Singularität vor, kann es sogar unendlich viele Gelenkwinkelstellungen geben, die zu einer vorgegebenen TCP-Pose führen. Nur auf dem Rand des Arbeitsraums sind Gelenkwinkellösungen eindeutig. Liegt die gewünschte TCP-Pose außerhalb des Arbeitsraums existiert keine Lösung. Ist die Vorwärtskinematik (3.4) bekannt, kann das inverse Kinematik-Problem durch Lösen eines nichtlinearen Gleichungssystems berechnet werden. Das Lösen eines solchen Gleichungssystems (3.8) ist meist sehr komplex, weshalb numerische Algorithmen zum Einsatz kommen. Hierbei können z.B. auch Verfahren aus der statischen Optimierung (vgl. Abschnitt 4.1), insbesondere das Newton-Verfahren 4.1.1, verwendet werden, weil das numerische Lösen eines nichtlinearen Gleichungssystems als Optimierungsproblem aufgefasst werden kann. Die gewünschte Position und Orien-



(a) Arbeitsraum (front) (b) Arbeitsraum (oben)

Abbildung 3.4: Visualisierung des Arbeitsraums für den Industrieroboter KUKA KR-210 (aus [56])

tierung des Endeffektors ${}^0\tilde{T}_E$ wird durch eine homogene Matrix vorgegeben (rechte Seite der Gleichung (3.8)). Sie wird mit der Vorwärtskinematik-Transformationsmatrix (3.4) gleichgesetzt.

$$ {}^0T_E(\boldsymbol{\theta}) = {}^0\tilde{T}_E \tag{3.8} $$

$$ {}^0T_E(\boldsymbol{\theta}) - {}^0\tilde{T}_E = \boldsymbol{O}^{4\times4} \tag{3.9} $$

$$ {}^0T_E(\theta_1,\ldots,\theta_n) - \begin{pmatrix} R_{11} & R_{12} & R_{13} & r_1 \\ R_{21} & R_{22} & R_{23} & r_2 \\ R_{31} & R_{32} & R_{33} & r_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \boldsymbol{O}^{4\times4} \tag{3.10} $$

Gleichung (3.10) hat nun die Standardform $F(\boldsymbol{x}) = 0$ und kann mit Hilfe eines Lösungsalgorithmus für nichtlineare Gleichungssysteme und Startwerten $\boldsymbol{\theta}_0$ nach $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_n)^\top$ gelöst werden. Auf den ersten Blick ergibt sich also ein System mit 12 Gleichungen mit θ_i für $i = 1,\ldots,n$ als Unbekannte. Die vier Gleichungen, die sich aufgrund der untersten Matrixzeile ergeben, haben keinen Einfluss auf das Gleichungssystem (3× Gleichung: $0 = 0$ und 1× Gleichung: $1 = 1$). Des Weiteren beinhalten Rotationsmatrizen eine Redundanz, da ihre neun Einträge nur von drei Parametern (den Euler-Winkeln) abhängen. Insgesamt lässt sich das Problem somit auf ein System mit sechs unabhängigen, nicht redundanten Gleichungen zurückführen. Je nach Roboterdesign und Anzahl von Freiheitsgraden ergibt sich ein unterbestimmtes ($n > 6$), bestimmtes ($n = 6$) oder überbestimmtes ($n < 6$) Gleichungssystem mit einer unterschiedlichen Anzahl an Lösungen. Ist die Orientierung des TCP unwichtig, kann das Gleichungssystem weiter reduziert werden. Dann ist lediglich die letzte Spalte der Transformationsmatrix von Bedeutung. In manchen Fällen lassen sich die Lösungen auch analytisch bestimmen, sodass sich eine geschlossene Form für die Gelenkwinkelstellungen $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_n)^\top$ in Abhängigkeit zur Endeffektorpose finden lässt. Die Struktur vieler Industrieroboter ist deshalb extra so gewählt worden, damit eine analytische Formel aufgestellt werden kann. Ein hinreichendes Kriterium für die Existenz einer solchen Formel besagt, dass drei benachbarte Gelenkachsen vorhanden sein müssen, die einen gemeinsamen Schnittpunkt haben. Für viele Industrieroboter trifft dies für die Achsen 4, 5 und 6 zu. Die Lösung des inversen Kinematik-Problems für die TCP-Position (Gelenke q_1, q_2, q_3) und TCP-Orientierung (Gelenke q_4, q_5, q_6) lässt sich dann entkoppeln. Eine Beispielrechnung für den Industrieroboter Unimation PUMA 560 ist in [57] zu finden.

3.2 Newtonsche Dynamik

Die Dynamik ist ein Teilgebiet der Mechanik und bietet die nötigen Grundlagen um Bewegungen von Systemen unter dem Einfluss von Kräften beschreiben zu können. Dabei wird ein Zusammenhang zwischen den agierenden Kräften und der Geometrie der Bewegung - der Kinematik - geschaffen. Im Gegensatz zur Statik tritt zusätzlich der Begriff der Zeit auf. Im Allgemeinen reicht die klassische Newtonsche Mechanik aus Probleme in der Technik hinreichend genau zu beschreiben. Erst in speziellen Szenarien, wie z.B. bei extrem hohen Geschwindigkeiten (→ Zeitdilatation) oder sehr hohen Energiezuständen (→ Massenzunahme), ist die Annahme der Newtonschen Mechanik nicht mehr gerechtfertigt und es müssen relativistische Effekte berücksichtigt werden. Die drei Newtonschen Gesetze, welche die Basis zur Beschreibung von Bewegungen bilden, lauten [58]:

> ***1. Newtonsches Gesetz - Trägheitsgesetz:*** *Es existiert ein Bezugssystem (Intertialsystem), bezüglich dessen jeder Massepunkt in Ruhe oder in gleichförmiger Bewegung bleibt, wenn er nicht durch Kräfte gezwungen wird, diesen Zustand zu ändern.*
>
> ***2. Newtonsches Gesetz - Grundgesetz der Dynamik:*** *In diesem Inertialsystem gilt für einen Massepunkt der Zusammenhang*
>
> $$\boldsymbol{F} = m \cdot \ddot{\boldsymbol{x}} \tag{3.11}$$
>
> *d.h. der Kraftvektor ist proportional zum Beschleunigungsvektor; die Proportionalitätskonstante ist die Masse (träge Masse).*
>
> ***3. Newtonsches Gesetz - Gesetz von actio & reactio:*** *Die wechselseitigen Beeinflussungen zweier Körper sind immer gleich und entgegengerichtet: Übt ein Körper 1 eine Kraft auf einen Körper 2 aus, so ist diese gleich groß und entgegengerichtet zu der Kraft, die der Körper 2 auf Körper 1 ausübt.*

Ein weiteres wichtiges Gesetz ist der Eulersche Drehimpulssatz [4], der für rotierende Massen von Bedeutung ist. Bewegt sich z.B. ein Körper bezüglich seines eigenen Schwerpunktes oder ein Massepunkt um einen Momentanpol, gilt der folgende Zusammenhang [8], [55].

> ***Eulerscher Drehimpulssatz:***
> *Der Drall (Impulsmoment) eines Massepunktes m mit Position $\boldsymbol{x}$ bezüglich eines Koordinatensystems S_i ist als Moment des Impulses, d.h. als*
>
> $$^{i}\boldsymbol{L} = \boldsymbol{x} \times m\dot{\boldsymbol{x}} \tag{3.12}$$
>
> *definiert. Somit ist der Drallsatz:*
>
> $$^{i}\dot{\boldsymbol{L}} = {}^{i}\boldsymbol{N} \tag{3.13}$$
>
> *Dabei ist $^{i}\boldsymbol{N}$ die Summe der Momente aller äußeren Kräfte. Im Fall des Starrkörpers, der kontinuierlich aus Massepunkten zusammengesetzt ist, gilt:*
>
> $$\underbrace{{}^{c_i}I_i \cdot \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \left({}^{c_i}I_i \cdot \boldsymbol{\omega}_i\right)}_{^{i}\dot{\boldsymbol{L}}} = {}^{i}\boldsymbol{N} \tag{3.14}$$

[4] auch Drallsatz

Zunächst soll die Dynamik eines einfachen Massepunktes näher erläutert werden. Um die Bewegung eines Massepunktes berechnen zu können, muss seine Masse sowie seine Lage und Geschwindigkeit zu einem Anfangszeitpunkt t_0 bekannt sein. Ist der Gesamtkraftvektor $\boldsymbol{F}(t)$, der sich aus dem *Superpositionsprinzip* [59] aller einwirkenden Kräfte auf den Massepunkt ergibt, zu jedem Zeitpunkt t bekannt, so kann durch Integration des zweiten Newtonschen Gesetztes (3.11) die Geschwindigkeit und Position des Massepunktes berechnet werden. Liegt keine geschlossene formelmäßige Beschreibung für $\boldsymbol{F}(t)$ vor oder enthält diese Unstetigkeit und Sprünge aufgrund von Kraftstößen, Kollisionen oder abrupten Impulsänderungen, können numerische Verfahren zur Lösung von (3.11) eingesetzt werden. Darauf wird in Abschnitt 3.4 näher eingegangen.

Beispiel 3.2. Einmassenschwinger
In diesem einführenden Beispiel wird die Dynamik eines einzelnen Massepunktes mit Hilfe der Newtonschen Gesetze hergeleitet. Ein Punkt mit der Masse m ist an einer Feder befestigt und kann entlang eines reibungsfreien Bodens bewegt werden (s. Abbildung 3.5).

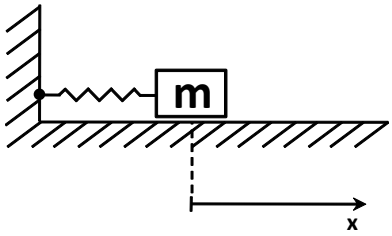


Abbildung 3.5: An einer Feder befestigte Punktmasse mit einem Freiheitsgrad

Es liegt also eine geführte Bewegung vor, welche die Freiheitsgrade des Massepunktes auf einen DoF in x-Richtung beschränken. Es ist somit ausreichend die Dynamik bezüglich dieses Freiheitsgrades zu beschreiben. Wird der Massepunkt aus der Ruhelage ausgelenkt, erfährt er aufgrund der Rückstellkraft der Feder eine Beschleunigung. Weist die verwendete Feder eine lineare Federkennlinie auf, ist die Rückstellkraft F_{spring} proportional zur Auslenkung x mit einem Faktor k. Weitere Kräfte wirken zunächst nicht auf den Massepunkt. Mit Hilfe des zweiten Newtonschen Gesetzes lässt sich folgende Bewegungsgleichung aufstellen.

$$m \cdot \ddot{x}(t) = F_{\text{spring}}(t) \tag{3.15}$$
$$m \cdot \ddot{x}(t) = -kx(t) \tag{3.16}$$
$$\ddot{x}(t) = -\frac{k}{m}x(t) \tag{3.17}$$

Ist das Objekt aus einem metallischen Stoff, kann es zusätzlich von außen durch Elektromagnete beeinflusst werden. Diese können gesteuert werden und üben eine weitere zeitabhängige Kraft, die Steuerungskraft $u(t)$, auf den Massepunkt aus.

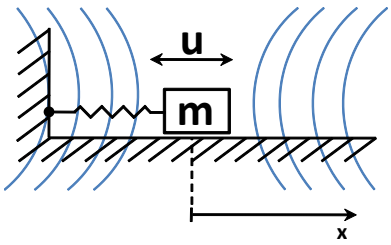


Abbildung 3.6: An einer Feder befestigte Punktmasse mit einem Freiheitsgrad und Steuerungskraft

$$m \cdot \ddot{x}(t) = F_{\text{spring}}(t) + u(t) \tag{3.18}$$
$$m \cdot \ddot{x}(t) = -kx(t) + u(t) \tag{3.19}$$
$$\ddot{x}(t) = -\frac{k}{m}x(t) + \frac{1}{m}u(t) \tag{3.20}$$

Um die Bewegung der Masse zu ermitteln, muss das Anfangswertproblem (IVP [5]) der DGL (3.20) (oder des äquivalenten DGL-Systems (3.97)) gelöst werden. Dazu sind die Anfangswerte für die Massenposition $x(t=0) = x_0$ und Massengeschwindigkeit $\dot{x}(t=0) = \dot{x}_0$ nötig. Abschnitt 3.4 beschäftigt sich mit der numerischen Lösung der IVP. Eine analytische Lösung kann durch zweifaches Integrieren der DGL (3.20) bestimmt werden. Da die Beschleunigung $\ddot{x}(t)$ als eine Funktion der Lage gegeben ist (abhängig von $x(t)$), kann die Lösung durch *Trennung der Veränderlichen* gefunden werden [58]. Es gilt $\ddot{x} = \frac{d\dot{x}}{dt} = \frac{d\dot{x}}{dx}\frac{dx}{dt} = \frac{d\dot{x}}{dx}\dot{x}$. Gleichung (3.20) kann somit geschrieben werden als:

$$\dot{x}\frac{d\dot{x}}{dx} = -\frac{k}{m}x + \frac{1}{m}u \tag{3.21}$$
$$\dot{x}d\dot{x} = (-\frac{k}{m}x + \frac{1}{m}u)dx \tag{3.22}$$
$$\int_{\dot{x}_0}^{\dot{x}} \dot{\bar{x}}\, d\dot{\bar{x}} = \int_{x_0}^{x} (-\frac{k}{m}\bar{x} + \frac{1}{m}u)\, d\bar{x} \tag{3.23}$$
$$\frac{1}{2}\dot{x}^2 - \frac{1}{2}\dot{x}_0^2 = \frac{1}{m}(-\frac{k}{2}x^2 + ux + \frac{k}{2}x_0^2 - ux_0) \tag{3.24}$$
$$\dot{x}^2 = \frac{1}{m}(-kx^2 + 2ux + kx_0^2 - 2ux_0 + m\dot{x}_0^2) \tag{3.25}$$
$$\dot{x}^2 = \frac{1}{m}(-k(x^2 - x_0^2) + 2u(x - x_0) + m\dot{x}_0^2) \tag{3.26}$$
$$\dot{x} = \pm\sqrt{\frac{1}{m}(-k(x^2 - x_0^2) + 2u(x - x_0) + m\dot{x}_0^2)} \tag{3.27}$$

Die resultierende Gleichung (3.27) ist wiederum eine Differentialgleichung, die von x abhängt. Sie kann umgeformt werden zu:

$$\frac{dx}{dt} = \sqrt{\frac{1}{m}(-k(x^2 - x_0^2) + 2u(x - x_0) + m\dot{x}_0^2)} \tag{3.28}$$
$$1\, dt = \frac{dx}{\sqrt{\frac{1}{m}(-k(x^2 - x_0^2) + 2u(x - x_0) + m\dot{x}_0^2)}} \tag{3.29}$$
$$\int_0^t d\bar{t} = \int_{x_0}^{x} \frac{d\bar{x}}{\sqrt{\frac{1}{m}(-k(\bar{x}^2 - x_0^2) + 2u(\bar{x} - x_0) + m\dot{x}_0^2)}} \tag{3.30}$$

Bereits dieses einfache System zeigt, wie schwer es sein kann eine analytische Lösung zu finden. Allerdings sind die Gleichungen für Einmassenschwinger aus der Mechanik bekannt. Falls der Spezialfall des

[5] engl.: Initial Value Problem

unbeeinflussten Systems herrscht (vgl. Gleichung (3.17)), gilt $u(t) \equiv 0$ und die Lösung von Gleichung (3.27) ist gegeben durch:

$$x(t) = x_0 \cos\left(\sqrt{\frac{k}{m}}\, t\right) + \frac{\dot{x}_0}{\sqrt{\frac{k}{m}}} \sin\left(\sqrt{\frac{k}{m}}\, t\right) \tag{3.31}$$

Ein alternativer Lösungsweg für Gleichung (3.20) (mit beliebigem $u(t) \not\equiv 0$) kann über die Berechnung des *Fundamentalsystems* und der *Partikulären Lösung* mittels *Variation der Konstanten* erfolgen. Siehe dazu Appendix B.

Bei der Simulation von Robotern spielen vor allem die Dynamik von Mehrkörpersystemen eine Rolle. Hier sind Starrkörper über Gelenke mit anderen Starrkörpern verbunden. Um das Verhalten eines einzelnen Starrkörpers simulieren zu können, ist zusätzlich zu dem translatorischen Verhalten seines Schwerpunktes die Orientierung des Körpers zu berücksichtigen. Ein fest mit dem Starrkörper verbundenes Koordinatensystem dient hierbei als Referenz. Üblicherweise liegt der Koordinatenursprung im Schwerpunkt.

Beispiel 3.3. Starrkörperbewegung
Ein Starrkörper mit der Masse m bewegt sich schwerelos durch den Raum. Er hat die Form eines Quaders mit der Länge l, der Breite b und der Höhe h. Sein körperfestes Koordinatensystem S_i liegt im Schwerpunkt. Die Masse des Starrkörpers ist homogen verteilt, sodass sein Trägheitstensor [58], [8] durch ${}^iI$ gegeben ist.

$${}^iI = \frac{m}{12} \begin{pmatrix} l^2+h^2 & 0 & 0 \\ 0 & b^2+h^2 & 0 \\ 0 & 0 & l^2+b^2 \end{pmatrix} \tag{3.32}$$

Die Orientierung des Starrkörpers lässt sich mittels der Rotationsmatrix R beschreiben, welche die aktuellen Basisvektoren des S_i Systems bzgl. des Weltkoordinatensystems S_0 enthält.

$$R(t) = \left({}^0\boldsymbol{e}_{x_i}(t) \mid {}^0\boldsymbol{e}_{y_i}(t) \mid {}^0\boldsymbol{e}_{z_i}(t)\right) \tag{3.33}$$

Wie in [60] gezeigt, können auch Quaternionen zur Beschreibung der Orientierung benutzt werden. Der Trägheitstensor lässt sich nun auch in Weltkoordinaten ausdrücken. Für den in den Bewegungsgleichungen benötigten inversen Trägheitstensor gilt:

$${}^0I^{-1} = R^\top\, {}^iI^{-1}\, R \tag{3.34}$$

Es wirkt nun eine konstante, sich mit dem Körper mitbewegende, externe Kraft $\boldsymbol{F}_{ext}$ auf den Starrkörper. Sie greift an den Koordinaten ${}^i\boldsymbol{p}$ des Körpers an. Dadurch ergibt sich am Schwerpunkt des Starrkörpers die Kraft $\boldsymbol{f}$ und das Drehmoment $\boldsymbol{\tau}$.

$$\boldsymbol{f} = \boldsymbol{F}_{ext} \tag{3.35}$$
$$\boldsymbol{\tau} = {}^i\boldsymbol{p} \times \boldsymbol{F}_{ext} \tag{3.36}$$

Die Bewegungsgleichungen des Starrkörpers bzgl. des Weltkoordinatensystems S_0 sind dann:

$$\ddot{\boldsymbol{r}}(t) = m^{-1} \cdot \boldsymbol{f} \tag{3.37}$$
$$\dot{\boldsymbol{\omega}}(t) = {}^0I^{-1}(t)\left(\boldsymbol{\tau} - (\boldsymbol{\omega}(t) \times ({}^0I(t)\boldsymbol{\omega}(t)))\right) \tag{3.38}$$

Bemerkung 3.1. Einmaliges Integrieren von (3.37) und (3.38) führt auf die lineare Geschwindigkeit des Schwerpunktes $\boldsymbol{v}(t) = \dot{\boldsymbol{r}}(t)$ und die Winkelgeschwindigkeit des Starrkörpers $\boldsymbol{\omega}(t)$. Nochmaliges Integrieren sollte demnach auf die Position $\boldsymbol{r}(t)$ und die Orientierung $R(t)$ des Starrkörpers führen. Letzteres ist nicht direkt durchführbar, denn die Beschreibung $\dot{R}(t)$ ist nicht kompatibel zu $\boldsymbol{\omega}(t)$. Mit Hilfe der Einbettung von $\boldsymbol{\omega}(t)$ in eine schiefsymmetrische Matrix kann eine passende Beschreibung von $\frac{d}{dt}R(t)$ trotzdem erfolgen [60].

$$\dot{R}(t) = B(\boldsymbol{\omega}(t)) \cdot R(t) \tag{3.39}$$

$$= \begin{pmatrix} 0 & -\omega_z & \omega_y \\ -\omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \cdot R \tag{3.40}$$

3.2.1 Dynamikmodell eines Industrieroboters mittels Newton-Euler Rekursion

Damit man die (inverse) Dynamik eines Roboters effizient herleiten kann, gibt es spezielle Algorithmen, die die Newtonschen Gesetze sowie den Drallsatz anwenden. Die Struktur vieler Mehrkörpersysteme in der Robotik ist baumartig, was explizit ausgenutzt werden kann. Die *Newton-Euler Rekursion* [54], [55] nutzt diesen Sachverhalt und berücksichtigt dementsprechend keine geschlossenen Schleifen in der Kinematikstruktur des Roboters. Der Algorithmus arbeitet rekursiv entlang der einzelnen Roboterglieder, die über Gelenke verbunden sind, und stellt dabei die einzelnen dynamischen Größen auf. Am Ende steht der Zusammenhang zwischen den aufzubringenden Motormomenten τ_i für eine gewünschte Roboterkonfiguration mit den Gelenkgrößen $q_i, \dot{q}_i, \ddot{q}_i$ fest. Der Algorithmus kann wie folgt zusammengefasst werden [55]:

- Die Bewegung der Roboterbasis (Glied 0) bzgl. eines Interialsystems ist über die Werte $\boldsymbol{\omega}_0$, $\dot{\boldsymbol{\omega}}_0$, $\boldsymbol{v}_0$,$\dot{\boldsymbol{v}}_0$ vorgegeben.

- Berechnungen für die Gelenke $i = 1, \ldots, n$:

$$\boldsymbol{\omega}_i = {}^iR_{i-1}\left(\boldsymbol{\omega}_{i-1} + \rho_i \, {}^{i-1}\boldsymbol{e}_{z_{i-1}}\dot{q}_i\right) \tag{3.41}$$

$$\dot{\boldsymbol{\omega}}_i = {}^iR_{i-1}\left(\boldsymbol{\omega}_{i-1} + \rho_i \left[{}^{i-1}\boldsymbol{e}_{z_{i-1}}\ddot{q}_i + \boldsymbol{\omega}_{i-1} \times ({}^{i-1}\boldsymbol{e}_{z_{i-1}}\dot{q}_i)\right]\right) \tag{3.42}$$

$$\boldsymbol{v}_i = {}^iR_{i-1}\boldsymbol{v}_{i-1} + \boldsymbol{\omega}_i \times ({}^iR_{i-1}\,{}^{i-1}\boldsymbol{r}_i) + (1-\rho_i)\,{}^i\boldsymbol{e}_{z_{i-1}}\dot{q}_i \tag{3.43}$$

$$\begin{aligned}\dot{\boldsymbol{v}}_i = {}^iR_{i-1}\dot{\boldsymbol{v}}_{i-1}\dot{\boldsymbol{\omega}}_i \times ({}^iR_{i-1}\,{}^{i-1}\boldsymbol{r}_i) + \boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times ({}^iR_{i-1}\,{}^{i-1}\boldsymbol{r}_i)\right) \\ + (1-\rho_i)\cdot\left[2\boldsymbol{\omega} \times {}^i\boldsymbol{e}_{z_{i-1}}\dot{q}_i + {}^i\boldsymbol{e}_{z_{i-1}}\ddot{q}_i\right]\end{aligned} \tag{3.44}$$

$$\dot{\boldsymbol{v}}_{c_i} = \dot{\boldsymbol{v}}_i + \dot{\boldsymbol{\omega}}_i \times {}^i\boldsymbol{r}_{c_i} + \boldsymbol{\omega} \times (\boldsymbol{\omega}_i \times {}^i\boldsymbol{r}_{c_i}) \tag{3.45}$$

$$\boldsymbol{F}_i = m_i \cdot \dot{\boldsymbol{v}}_{c_i} \tag{3.46}$$

$$\boldsymbol{N}_i = {}^{c_i}I_i \cdot \dot{\boldsymbol{\omega}}_i \times \boldsymbol{\omega}_i \times ({}^{c_i}I_i \cdot \boldsymbol{\omega}_i) \tag{3.47}$$

- Für den Endeffektor (Glied n) werden die Kraft $\boldsymbol{f}_{n+1}$ und das Moment $\boldsymbol{n}_{n+1}$ vorgegeben.

◦ Berechnungen für die Gelenke $i = n, \ldots, 1$:

$$\boldsymbol{f}_i = {}^{i}R_{i+1}\boldsymbol{f}_{i+1} + \boldsymbol{F}_i \tag{3.48}$$

$$\boldsymbol{n}_i = {}^{i}R_{i+1}\boldsymbol{n}_{i+1} + \left({}^{i}R_{i-1}{}^{i-1}\boldsymbol{r}_i + {}^{i}\boldsymbol{r}_{c_i}\right) \times \boldsymbol{F}_i + ({}^{i}R_{i-1}{}^{i-1}\boldsymbol{r}_i) \times ({}^{i}R_{i+1}\boldsymbol{f}_{i+1}) + \boldsymbol{N}_i \tag{3.49}$$

$$\tau_i = \begin{cases} ({}^{i}\boldsymbol{e}_{z_{i-1}})^{\top}\, \boldsymbol{n}_i, & \text{falls } \rho_i = 1 \text{ (Drehgelenk)} \\ ({}^{i}\boldsymbol{e}_{z_{i-1}})^{\top}\, \boldsymbol{f}_i, & \text{falls } \rho_i = 0 \text{ (Schubgelenk)} \end{cases} \tag{3.50}$$

Die dabei benutzten Bezeichner sind in der folgenden Tabelle 3.2 erläutert.

Bezeichner	Bedeutung
m_i	Masse des i.-ten Roboterglieds
τ_i	Drehmoment/Kraft des Motors im Koordinatensystem S_i
$\boldsymbol{\omega}_i$, $\dot{\boldsymbol{\omega}}_i$	Winkelgeschwindigkeit, Winkelbeschleunigung im Koordinatensystem S_i
$\boldsymbol{v}_i$, $\dot{\boldsymbol{v}}_i$	lineare Geschwindigkeit, Beschleunigung im Koordinatensystem S_i
$\boldsymbol{v}_{c_i}$, $\dot{\boldsymbol{v}}_{c_i}$	lineare Geschwindigkeit, Beschleunigung am Schwerpunkt des i.-ten Glieds
$\boldsymbol{F}_i$	wirkende Kraft am Schwerpunkt des i.-ten Glieds
$\boldsymbol{N}_i$	wirkendes Drehmoment am Schwerpunkt des i.-ten Glieds
$\boldsymbol{f}_i$	wirkende Kraft im Koordinatensystem S_i
$\boldsymbol{n}_i$	wirkendes Drehmoment im Koordinatensystem S_i
ρ	$\rho = 1$: Drehgelenk, $\rho = 0$: Schubgelenk
${}^{a}R_b$	Rotationsmatrix zwischen den Systemen S_a und S_b
${}^{a}\boldsymbol{r}_b$	Translationsvektor zwischen den Systemen S_a und S_b
${}^{a}\boldsymbol{e}_{z_b}$	z-Achse des Koordinatensystems S_b bzgl. Koordinaten in S_a
${}^{i}\boldsymbol{r}_{c_i}$	Position des Schwerpunktes des i.-ten Glieds bzgl. Koordinatensystem S_i
${}^{c_i}I_i$	Trägheitstensor des i.-ten Glieds bzgl. seines Schwerpunktes

Tabelle 3.2: Erläuterungen zu den in der Newton-Euler Rekursion verwendeten Variablen

Die Dynamik liegt noch nicht in geschlossener Form vor. Die Ergebnisse aus der Newton-Euler Rekursion können aber entsprechend umgeformt und zusammengefasst werden, dass die inverse Dynamikgleichung (3.51) entsteht [55].

$$\boldsymbol{\tau} = M(\boldsymbol{q}) \cdot \ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + G(\boldsymbol{q}) \tag{3.51}$$

mit

$$M(\boldsymbol{q}) \in \mathbb{R}^{n \times n} \;:\; \text{symmetrische, positiv definite Massenmatrix} \tag{3.52}$$

$$C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n} \;:\; \text{Vektor der Zentrifugal- und Coriolisanteile} \tag{3.53}$$

$$G(\boldsymbol{q}) \in \mathbb{R}^{n} \;:\; \text{Vektor der Gravitationsanteile} \tag{3.54}$$

Durch Auflösen nach $\ddot{\boldsymbol{q}}$ ergibt sich das direkte Dynamikmodell (3.55), womit sich die Bewegung des Roboters bei gegebenen Motormomentverläufen $\tau(t)$ berechnen lässt.

$$\ddot{\boldsymbol{q}} = M^{-1}(\boldsymbol{q}) \cdot \left(\boldsymbol{\tau} - C(\boldsymbol{q}, \dot{\boldsymbol{q}}) - G(\boldsymbol{q})\right) \tag{3.55}$$

Ein ausführliches Beispiel zur Berechnung der Dynamik eines 2-DoF Manipulatorarms mit der Newton-Euler Rekursion ist in [55] zu finden.

3.3 Lagrangesche Dynamik

Ein weiterer Weg Bewegungsgleichungen dynamischer Systeme aufzustellen, geht auf den Ansatz von Lagrange (*Lagrange-Formalismus*) zurück. Im Gegensatz zur Newtonschen Methode ist es hierbei nicht nötig die angreifenden Kraftvektoren zu bestimmen, sondern vielmehr werden die potentielle und kinetische Energie aller massebehafteter Körper bzw. Punkte betrachtet. Potentielle und kinetische Energie sind skalare Größen. Dem Lagrange-Formalismus liegt das *Prinzip der kleinsten Wirkung* [6] oder auch *Hamiltonsche Prinzip* zu Grunde. Dieses in der Natur vorkommende Phänomen bezieht sich auf konservative Systeme und sorgt dafür, dass Bewegungen immer in optimaler Weise (energieminimal) verlaufen. So ergeben sich z.B. nach Kepler Kreis- oder Ellipsenbahnen [8] für die Bewegung von Planeten oder der zeitlich kürzeste Weg eines Lichtstrahls nach dem Fermatschen Prinzip [61].
Das Funktional $\mathscr{S}$ beschreibt die *Wirkung* eines mechanischen Systems. Jedes mechanische System lässt sich vollständig über eine Lagrangefunktion $L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ definieren. Bewegt es sich in einer Zeit von 0 bis t_f, muss das Funktional $\mathscr{S}$ seinen minimalen Wert annehmen.

$$\mathscr{S} = \int_0^{t_f} L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\, dt \tag{3.56}$$

Die Lagrangefunktion wird über die Differenz aus kinetischer Energie T und potentieller Energie U aller Körper und Massepunkte bestimmt. Sie ist abhängig von den generalisierten Koordinaten q_i des mechanischen Systems. Diese Darstellung hat die Eigenschaft, dass sie die minimale Anzahl an Koordinaten besitzt, um das Verhalten des Systems beschreiben zu können. D.h., es existieren exakt gleich viele Koordinaten q_i, wie das System Freiheitsgrade besitzt. Der Ortsvektor $\boldsymbol{r}_i$ eines Massepunktes ist also zu jeder Zeit eindeutig über $\boldsymbol{r}_i = \boldsymbol{r}_i(q_1, q_2, \ldots, q_n, t)$ bestimmt. Seine Geschwindigkeit ist somit $\boldsymbol{v}_i = \frac{d}{dt}\boldsymbol{r}_i = \dot{\boldsymbol{r}}_i$. Der allgemeine Zusammenhang für die Lagrangefunktion lautet:

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = T - U \tag{3.57}$$

Die kinetische Energie eines Massepunktes mit der Masse m_i ist:

$$T_i = \frac{1}{2} m_i (\boldsymbol{v}_i^\top \cdot \boldsymbol{v}_i) \tag{3.58}$$

Für einen Starrkörper der Masse m_i und dem Trägheitstensor I_i muss zusätzlich noch die Winkelgeschwindigkeit $\boldsymbol{\omega}$ in die Berechnung der kinetischen Energie einfließen.

$$T_i = \frac{1}{2} m_i (\boldsymbol{v}_i^\top \cdot \boldsymbol{v}_i) + \frac{1}{2} \boldsymbol{\omega}_i^\top I_i \boldsymbol{\omega}_i \tag{3.59}$$

Die potentielle Energie ergibt sich, indem man die wirkende Kraft entlang des Weges in einem Potentialfeld integriert.

$$U_i = \int_a^b \boldsymbol{F}_i d\boldsymbol{x} \tag{3.60}$$

In den meisten Fällen bewegt sich das dynamische System im Gravitationsfeld der Erde. In guter Näherung kann angenommen werden, dass das Gravitationsfeld für geringe Höhenunterschiede homogen ist

[6] engl.: Principle of Least Action

und in Bodennähe die konstante Beschleunigung $\boldsymbol{g}$ wirkt. Somit hat ein Körper der Masse m_i, der sich in Position $\boldsymbol{r}$ über der Referenzhöhe $\boldsymbol{r}_{ref}$ befindet, die potentielle Energie

$$U_i = \int_{\boldsymbol{r}_{ref}}^{r} \boldsymbol{F}_i d\boldsymbol{x} = \int_{\boldsymbol{r}_{ref}}^{r} m_i \boldsymbol{g}^\top d\boldsymbol{x} = -m_i \boldsymbol{g}^\top \boldsymbol{r} + \boldsymbol{r}_{ref} \tag{3.61}$$

Die potentielle Energie einer Feder ist äquivalent zur Arbeit, die beim Spannen der Feder entsteht.

$$U_{\text{spring}} = -\int_0^x F(x)\, dx = \frac{1}{2} k x^2 \tag{3.62}$$

Insgesamt lautet die Lagrangefunktion eines Systems mit $i = 1 \dots n$ Körpern somit:

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = \sum_{i=1}^{n} T_i - \sum_{i=1}^{n} U_i \tag{3.63}$$

$$= \sum_{i=1}^{n} \frac{1}{2} m_i (\boldsymbol{v}_i^\top \cdot \boldsymbol{v}_i) + \frac{1}{2} (\boldsymbol{\omega}_i^\top I_i \boldsymbol{\omega}_i) - \sum_{i=1}^{n} -m_i \boldsymbol{g}^\top \boldsymbol{r_i} + \boldsymbol{r_{ref}} \tag{3.64}$$

3.3.1 Die Euler-Lagrangesche Differentialgleichung

Aus dem Prinzip der kleinsten Wirkung kann mit Hilfe der Variationsrechnung die Euler-Lagrangesche Gleichung hergeleitet werden. Sie spielt nicht nur in der Dynamik, sondern auch für die Optimierung (s. Abschnitt 4.2) eine zentrale Rolle. In der Natur stellen sich für die Bewegung von Systemen immer optimale Trajektorien ein. Die zeitabhängigen Funktionen der einzelnen Freiheitsgrade des Systems $q_i(t)$ führen also dazu, dass das Funktional $\mathscr{S}$ aus (3.56) minimal wird [61]. Als Funktional bezeichnet man eine Abbildung $J : D \to \mathbb{R}$, die innerhalb eines Vektorraums V jedem Vektor $\boldsymbol{v} \in D$ des Definitionsbereichs D eine reelle Zahl $J[\boldsymbol{v}]$ zuordnet. Handelt es sich bei den Vektoren um Funktionen $f : \mathbb{R}^n \to \mathbb{R}^m$, geschieht dies in einem Funktionenraum. Dieser Fall ist hier ausschlaggebend, da hier die einzelnen Freiheitsgrade $q_i(t)$ als Funktionen in Abhängigkeit zur Zeit zu Grunde liegen. Das Funktional $\mathscr{S}[\boldsymbol{q}(t)]$ beschreibt das Prinzip der kleinsten Wirkung und hängt von der Lagrangefunktion $L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ ab.

Die 1. Variation (Gâteaux-Variation)
Das Funktional $\mathscr{S}[\boldsymbol{q}(t)]$ soll minimal sein um dem Prinzip der kleinsten Wirkung zu genügen. Es muss also ein $\boldsymbol{q}^*(t) \in D$ im Definitionsbereich gefunden werden, sodass

$$\mathscr{S}[\boldsymbol{q}(t)] = \int_0^{t_f} L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\, dt \to \text{Extr! (Min!)} \tag{3.65}$$

Ausgehend vom Brachistochroneproblem (s. Abschnitt 2.1) entwickelte Euler eine Methode um Aufgaben diesen Typs zu lösen. Dabei wird zur angenommenen Lösung $\boldsymbol{q}^*(t)$ eine benachbarte (variierte) Funktion $\boldsymbol{q}_\epsilon(t) = \boldsymbol{q}^*(t) + \epsilon \cdot \delta\boldsymbol{q}(t)$ gebildet (s. Abbildung 3.7). Dieser addiert zur optimalen Funktion $\boldsymbol{q}^*(t)$ eine mit $\epsilon \in \mathbb{R}$ skalierte „Abweichung“ $\delta\boldsymbol{q}(t)$. Für diese muss $\delta\boldsymbol{q}(0) = \delta\boldsymbol{q}(t_f) = 0$ gelten um die Randbedingungen einzuhalten.

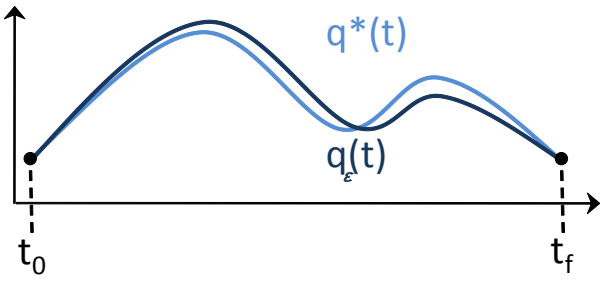


Abbildung 3.7: Variation der optimalen Funktion $q^*(t)$

Das Funktional $\mathscr{S}$ nimmt demnach den Wert

$$\mathscr{S}[\boldsymbol{q}^*(t)+\epsilon\cdot\delta\boldsymbol{q}(t)]=\int_0^{t_f} L(\boldsymbol{q}^*+\epsilon\cdot\delta\boldsymbol{q},\dot{\boldsymbol{q}}^*+\epsilon\cdot\delta\dot{\boldsymbol{q}},t)\,dt=\tilde{\mathscr{S}}[\epsilon] \tag{3.66}$$

an und ist nun von ϵ abhängig. Es ist offensichtlich, dass das Funktional $\tilde{\mathscr{S}}[\epsilon]$ seinen minimalen Wert für $\epsilon=0$ annimmt, da somit die Abbweichung zum optimalen Verlauf wegfällt. Ein Funktional $J[\epsilon]$ besitzt also einen Extremwert an der Stelle $J[\epsilon]|_{\epsilon=0}$. Damit gilt:

$$0=\frac{d}{d\epsilon}J[\epsilon]\bigg|_{\epsilon=0}=\lim_{\epsilon\to o}\frac{1}{\epsilon}(\mathscr{S}[\boldsymbol{q}^*(t)+\epsilon\cdot\delta\boldsymbol{q}(t)]-\mathscr{S}[\boldsymbol{q}^*(t)]) \tag{3.67}$$

$$=\lim_{\epsilon\to o}\frac{1}{\epsilon}(\mathscr{S}[\boldsymbol{q}_\epsilon(t)]-\mathscr{S}[\boldsymbol{q}^*(t)]) \tag{3.68}$$

Die erste Variation eines allgemeinen Funktionals $J:\mathbb{R}^n\to\mathbb{R}$ in Abhängigkeit zur Funktion $\boldsymbol{f}(x)$ lautet:

$$\delta J[\boldsymbol{f}(x)]:=\frac{d}{d\epsilon}J[\boldsymbol{f}^*(x)+\epsilon\cdot\delta\boldsymbol{f}(x)]\bigg|_{\epsilon=0} \tag{3.69}$$

$$:=\frac{d}{d\epsilon}J[\boldsymbol{f}_\epsilon(x)]\bigg|_{\epsilon=0} \tag{3.70}$$

Daraus ergibt sich die erste notwendige Bedingung an die Lösung $\boldsymbol{f}^*(x)$ für ein allgemeines Variationsproblem $J[\boldsymbol{f}(x)]\to\textsc{Extr}!$:

$$\delta J[\boldsymbol{f}^*(x)]=0 \tag{3.71}$$

Die 2. Variation

Die erste notwendige Bedingung (3.71) erfüllen alle Funktionen $\boldsymbol{f}(x)$, die für $\epsilon=0$ einen Extremwert besitzen, auch solche, die das Funktional J maximal werden lassen. Gesucht ist aber eine Funktion $\boldsymbol{f}^*(x)$, die das Funktional J minimiert. Als zweite Variation bezeichnet man

$$\delta^2 J[\boldsymbol{f}(x)]:=\frac{d^2}{d\epsilon^2}J[\boldsymbol{f}_\epsilon(x)]\bigg|_{\epsilon=0} \tag{3.72}$$

Daraus ergibt sich die zweite notwendige Bedingung an einen extremalen Lösungskandidaten

$$\delta^2 J[\boldsymbol{f}(x)]\geq 0 \tag{3.73}$$

Sie stellt sicher, dass es sich um ein Minimum handelt. Analog kann auch die zweite notwendige Bedingung umformuliert werden, falls ein Maximum erwünscht ist.

Bemerkung 3.2. Wie in Gleichung (3.71) und (3.73) gezeigt, gelten die aus der 1. und 2. Variation hervorgehenden notwendigen Bedingungen auch für allgemeinere Funktionale, die nicht notwendigerweise die Form $J[y(x)] = \int_a^b F(y(x), y'(x), x)\ dx$ besitzen. Allerdings gibt die *Euler-Lagrangesche Differentialgleichung* nur die Lösung für Variationsprobleme mit einem solchen Funktionaltyp an. Es gibt jedoch Formulierungen der Euler-Lagrangeschen Differentialgleichung, die höhere Ableitungen $y^{(n)}(x)$ im Funktional $J[y(x)]$ zulassen.

Die Euler-Lagrangesche Differentialgleichung lautet:

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}_i} L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) - \frac{\partial}{\partial q_i} L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = 0\ , \text{ für } i = 1, \ldots, n \tag{3.74}$$

Sie bilden die Lösung für das Variationsproblem (3.65) und ergeben somit die Bewegungsdifferentialgleichungen für ein dynamisches System, dass durch die Lagrangefunktion $L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)$ beschrieben wird. Der Beweis gelingt durch Einsetzen des Funktionals (3.56) in die erste notwendige Bedingung (3.71). Er wird nun für den eindimensionalen Fall gezeigt, kann aber ohne weiteres auf höherdimensionale Probleme übertragen werden.

$$0 = \delta \mathscr{S}[q(t)] \tag{3.75}$$

$$\Leftrightarrow 0 = \frac{d}{d\epsilon} \int_0^{t_f} L(q^* + \epsilon \cdot \delta q, \dot{q}^* + \epsilon \cdot \delta \dot{q}, t)\,|_{\epsilon=0}\ dt \tag{3.76}$$

$$\Leftrightarrow 0 = \int_0^{t_f} \frac{dL(q^*, \dot{q}^*, t)}{dq} \cdot \delta q + \frac{dL(q^*, \dot{q}^*, t)}{d\dot{q}} \cdot \delta \dot{q}\ dt \tag{3.77}$$

$$\Leftrightarrow 0 = \int_0^{t_f} \frac{dL(q^*, \dot{q}^*, t)}{dq} \cdot \delta q\ dt - \int_0^{t_f} \frac{d}{dt}\frac{dL(q^*, \dot{q}^*, t)}{d\dot{q}} \cdot \delta q\ dt + \left[\frac{dL(q^*, \dot{q}^*, t)}{d\dot{q}_i} \cdot \delta q\right]_{t=0}^{t=t_f} \tag{3.78}$$

$$\Leftrightarrow 0 = \int_0^{t_f} \left(\frac{dL(q^*, \dot{q}^*, t)}{dq} - \frac{d}{dt}\frac{dL(q^*, \dot{q}^*, t)}{d\dot{q}}\right) \cdot \delta q\ dt \tag{3.79}$$

Dieser Term (3.79) muss für beliebige δq den Wert 0 annehmen. Also gilt die eindimensionale Euler-Lagrangesche Differentialgleichung:

$$0 = \frac{d}{dt}\frac{d}{d\dot{q}} L(q, \dot{q}, t) - \frac{d}{dq} L(q, \dot{q}, t) \tag{3.80}$$

Das Variationsproblem (3.65) (und Variationsprobleme in der gleichen Form, wie sie in Bermerkung 3.2 beschrieben sind) kann also in das Randwertproblem der Differentialgleichung (3.74) mit den Bedingungen $\boldsymbol{q}(0) = \boldsymbol{q}_0$ und $\boldsymbol{q}(t_f) = \boldsymbol{q}_{t_f}$ überführt werden. Falls nicht sichergestellt werden kann oder es aus physikalischen Gründen nicht ersichtlich ist, dass die über die Euler-Lagrangesche Differentialgleichung bestimmte Lösung minimal, sondern ggf. maximal ist, müssen zusätzliche Kriterien (darunter die aus der zweiten Variation hervorgehende *Legendre-Clebsch* Bedingung (3.81)) überprüft werden.

$$\frac{\partial^2}{\partial \dot{\boldsymbol{q}}^2} L(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \geq 0 \tag{3.81}$$

3.3.2 Dynamikmodell eines Industrieroboters mittels Lagrange-Formalismus

Analog zu Abschnitt 3.2.1 kann die Dynamik von Robotern auch mit dem soeben vorangegangenen Lagrange-Formalismus entwickelt werden. Generell ist die Berechnungseffizienz im Vergleich zur Newton-Euler Rekursion nicht so hoch. Allerdings können die entstehenden Formeln geschickt zusammengefasst und umgeformt werden, sodass auch damit effiziente Berechnungen möglich sind. Diese Prozedur ist aber meist sehr aufwändig und nur von Hand durchzuführen. Ein Vorteil des Lagrange-Formalismus ist die Anwendbarkeit auf allgemeine Dynamikmodelle, wobei die Newton-Euler Rekursion auf baumartige Strukturen beschränkt ist. Die Berechnung der Roboterdynamik umfasst die Herleitung der kinetischen und potentiellen Energie sowie das Aufstellen der Euler-Lagrangeschen Differentialgleichung. Eine Zusammenfassung der einzelnen Berechnungschritte folgt [55]:

- Berechnung der kinetischen Energie K_i für Roboterglied i:

$$K_i(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2}\, m_i\, {}^0\boldsymbol{v}_{c_i}^{\top} \cdot {}^0\boldsymbol{v}_{c_i} + \frac{1}{2}\, {}^0\boldsymbol{\omega}_i^{\top}\, {}^{c_i}I_i\, {}^0\boldsymbol{\omega}_i \tag{3.82}$$

$$= \frac{1}{2}\, m_i\, ({}^0J_{c_i,v}\, \dot{\boldsymbol{q}})^{\top} \cdot ({}^0J_{c_i,v}\, \dot{\boldsymbol{q}}) + \frac{1}{2}({}^0J_{i,\omega}\, \dot{\boldsymbol{q}})^{\top}\, {}^{c_i}I_i\, ({}^0J_{i,\omega}\, \dot{\boldsymbol{q}}) \tag{3.83}$$

$$= \frac{1}{2}\, \dot{\boldsymbol{q}}^{\top} \left(m_i ({}^0J_{c_i,v})^{\top}\, {}^0J_{c_i,v} + ({}^0J_{i_\omega})^{\top}\, {}^{c_i}I_i\, {}^0J_{i,\omega} \right) \dot{\boldsymbol{q}} \tag{3.84}$$

mit ${}^0v_{c_i} = {}^0J_{c_i,v}(\boldsymbol{q})\, \dot{\boldsymbol{q}}$ und ${}^0\omega_i = {}^0J_{i,\omega}(\boldsymbol{q})\, \dot{\boldsymbol{q}}$

- Berechnung der gesamten kinetischen Energie K des Manipulators:

$$K(\boldsymbol{q},\dot{\boldsymbol{q}}) = \sum_{i=0}^{n} K_i(\boldsymbol{q},\dot{\boldsymbol{q}}) \tag{3.85}$$

$$= \frac{1}{2}\dot{\boldsymbol{q}}^{\top} \underbrace{\left(\sum_{i=0}^{n} \left((m_i ({}^0J_{c_i,v})^{\top}\, {}^0J_{c_i,v} + ({}^0J_{i,\omega})^{\top}\, {}^{c_i}I_i\, ({}^0J_{i,\omega}) \right) \right)}_{:=M(\boldsymbol{q})} \dot{\boldsymbol{q}} \tag{3.86}$$

$$= \frac{1}{2}\, \dot{\boldsymbol{q}}^{\top} \cdot M(\boldsymbol{q}) \cdot \dot{\boldsymbol{q}} \tag{3.87}$$

- Berechnung der potentiellen Energie P_i für Roboterglied i:

$$P_i = -m_i \boldsymbol{q}^{\top}\, {}^0r_{c_i} + P_{Ref,i} \tag{3.88}$$

- Berechnung der gesamten potentiellen Energie K des Manipulators:

$$P(\boldsymbol{q}) = \sum_{i=0}^{n} P_i \tag{3.89}$$

◦ Die Lagrangefunktion lautet somit:

$$L(\boldsymbol{q},\dot{\boldsymbol{q}}) = K(\boldsymbol{q},\dot{\boldsymbol{q}}) - P(\boldsymbol{q}) \tag{3.90}$$

$$= \frac{1}{2}\dot{\boldsymbol{q}}^{\top}\left(\sum_{i=0}^{n}\left(m_i({}^0J_{c_i,v})^{\top}\,{}^0J_{c_i,v} + ({}^0J_{i,\omega})^{\top}\,{}^{c_i}I_i\,({}^0J_{i,\omega})\right)\right)\dot{\boldsymbol{q}} - \sum_{i=0}^{n}\left(-m_i\boldsymbol{q}^{\top}\,{}^0r_{c_i} + P_{Ref,i}\right) \tag{3.91}$$

◦ Einsetzen in die Euler-Lagrangesche Gleichung (3.74) mit τ_i als nicht konservative Motormomente auf der rechten Seite ergibt schließlich:

$$\frac{d}{dt}\,\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \tag{3.92}$$

$$\Leftrightarrow \frac{d}{dt}\,\frac{\partial K(\boldsymbol{q}(t),\dot{\boldsymbol{q}})}{\partial \dot{q}_i} - \frac{\partial K(\boldsymbol{q}(t),\dot{\boldsymbol{q}})}{\partial q_i} + \frac{\partial P(\boldsymbol{q})}{\partial q_i} = \tau_i(t) \tag{3.93}$$

Symbolisches Ausrechnen führt auf die aus Abschnitt 3.2.1 bekannte inverse Dynamikgleichung (3.51).

$$\boldsymbol{\tau} = M(\boldsymbol{q})\cdot\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}}) + G(\boldsymbol{q})$$

Beide Verfahren, Newton-Euler Rekursion und Langrange-Formalismus, führen also auf ein äquivalentes Dynamikmodell. Die benutzten Bezeichner sind analog zur Tabelle 3.2. Die speziell für diesen Algorithmus verwendeten Variablen sind in der folgenden Tabelle erläutert 3.3.

Bezeichner	Bedeutung
$\boldsymbol{g}$	Gravitationsvektor
K_i	kinetische Energie des i.-ten Glieds
K	kinetische Energie des Roboters
P_i	potentielle Energie des i.-ten Glieds
P	potentielle Energie des Roboters
$P_{Ref,i}$	potentielle Energie des i-ten Glieds aufgrund der Referenzhöhe (=const.)
${}^aJ_{b,v}$	Jacobimatrix des Manipulators: Abbildung von $\dot{\boldsymbol{q}}$ nach ${}^a\boldsymbol{v}_b$.
${}^aJ_{b,\omega}$	Jacobimatrix des Manipulators: Abbildung von $\dot{\boldsymbol{q}}$ nach ${}^a\boldsymbol{\omega}_b$.
L	Lagrangefunktion

Tabelle 3.3: Erläuterungen zu den im Lagrange-Formalismus verwendeten Variablen

Ein ausführliches Beispiel zur Berechnung der Dynamik eines 2-DoF Manipulatorarms mit dem Lagrange-Formalismus ist in [55] zu finden.

3.4 Numerische Lösung von Bewegungsdifferentialgleichungen

Bei der Aufstellung der Bewegungsgleichungen von Systemen mit Hilfe der Newtonschen Gesetze oder dem Lagrange-Formalismus entstehen gewöhnliche Differentialgleichungen (ODE [7]) zweiter Ordnung.

[7] engl.: Ordinary Differential Equation

Jede Differentialgleichung höherer Ordnung kann über die Einführung von Hilfsvariablen in ein System von Differentialgleichungen erster Ordnung umgeformt werden.

$$\ddot{x} = f(x, \dot{x}, t) \tag{3.94}$$

$$x_1 := x,\ x_2 := \dot{x} \ \rightarrow\ \tilde{\boldsymbol{x}} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3.95}$$

$$\Leftrightarrow \dot{\tilde{\boldsymbol{x}}} = \boldsymbol{f}(\tilde{\boldsymbol{x}}) = \begin{pmatrix} x_2 \\ f(x_1, x_2, t) \end{pmatrix} \tag{3.96}$$

Beispiel 3.4. Einmassenschwinger Differentialgleichungssystem 1. Ordnung
Mit der vorangegangenen Umformungsvorschrift kann Gleichung (3.20) zu einem Differentialgleichungssystem 1. Ordnung umgewandelt werden.

$$\dot{\tilde{x}} = \begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v \\ \frac{k}{m} x(t) + \frac{1}{m} u(t) \end{pmatrix} \tag{3.97}$$

Des Weiteren kann jede nicht autonome Differentialgleichung, die eine explizite Zeitabhängigkeit der rechten Seite $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, t)$ besitzt, in ein autonomes System $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x})$ transformiert werden [1]. Dies geschieht über die Einführung einer so genannten „Uhrzeitvariablen“ $x_{n+1} = t$ mit der Anfangsbedingung $x_{n+1}(0) = 0$. Die Ableitung dieser Variable ist offensichtlich $\dot{x}_{n+1} = 1$. Das entstehende autonome Differentialgleichungssystem lautet:

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \\ \dot{x}_{n+1} \end{pmatrix} = \begin{pmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \\ 1 \end{pmatrix} \tag{3.98}$$

Somit kann jede gewöhnliche Differentialgleichung auf ein System in Standardform (3.99) gebracht werden.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{3.99}$$

Ist die DGL zusätzlich steuerbar über eine Funktion $\boldsymbol{u}(t)$, erhält man die Standardform (3.100).

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{3.100}$$

Für das Anfangswertproblem (IVP) (3.101)-(3.104) können verschiedene numerische Lösungsalgorithmen eingesetzt werden, die jeweils einen Kompromiss zwischen Berechnungseffizienz und Genauigkeit darstellen. Es folgt eine Übersicht der gebräuchlichsten Integrationsmethoden, die sich in explizite und implizite Methoden einteilen lassen. Weiterführende Techniken, wie die adaptive Zeitschrittweitenbestimmung für ein optimales Δt oder Mehrschrittverfahren, sind z.B. in [62] zu finden.

$$\dot{\tilde{x}}(t) = \boldsymbol{f}(\tilde{\boldsymbol{x}}(t)) \tag{3.101}$$

$$\Leftrightarrow \begin{pmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{v}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}(t) \\ \boldsymbol{a}(t) \end{pmatrix} \tag{3.102}$$

$$\Leftrightarrow \begin{pmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{v}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}(t) \\ \frac{1}{m} \boldsymbol{F}(\tilde{\boldsymbol{x}}) \end{pmatrix} \tag{3.103}$$

$$\text{mit } \boldsymbol{x}(0) = \boldsymbol{x}_0,\ \boldsymbol{v}(0) = \boldsymbol{v}_0 \tag{3.104}$$

3.4.1 Explizite Integrationsmethoden

Explizite Methoden berechnen den neuen Systemzustand zur Zeit $t_{i+1} = t_i + \Delta t$ ausschließlich aus vorangegangenen Zeitpunkten. Dazu müssen die Terme auf den rechten Seiten der Integrationsvorschriften nur ausgewertet werden. Das Auflösen nach einer Variablen bzw. das Lösen eines (ggf. nichtlinearen) Gleichungssystems ist nicht erforderlich, da bereits alle Terme der rechten Seite bekannt sind.

Explizites Euler-Verfahren
Das explizite Euler-Verfahren ist eine sehr schnelle Methode und besitzt einen Positions- und Geschwindigkeitsfehler der Größenordnung $\mathcal{O}(\Delta t^2)$. Allerdings eignet sie sich nicht dazu steife Differentialgleichungen zu berechnen, da sie nicht notwendigerweise stabil arbeitet. Hierfür wäre eine sehr kleine Zeitschrittweite Δt nötig, was sich wiederum negativ auf die eigentlich hohe Berechnungseffizienz auswirkt.

$$\boldsymbol{v}(t_0 + \Delta t) = \boldsymbol{v}_0 + \Delta t \ \boldsymbol{a}(t_0) \tag{3.105}$$
$$\boldsymbol{x}(t_0 + \Delta t) = \boldsymbol{x}_0 + \Delta t \ \boldsymbol{v}(t_0) \tag{3.106}$$

Verlet-Verfahren
Das Verlet-Verfahren benötigt ebenso wie das Euler-Verfahren nur eine Auswertung der Beschleunigung $\boldsymbol{a}$. Liegen keine geschwindigkeitsabhängigen Dämpfungs- oder Reibungseffekte in der Systemdynamik vor, kann auf die Berechnung von Gleichung (3.107) verzichtet werden, da in diesem Fall beide Integrationsvorschriften entkoppelt sind. Die neue Position (3.108) ist in diesem Fall nur noch abhängig von den beiden vorausgegangenen Positionen und der Beschleunigung. Die Geschwindigkeit lässt sich mit Fehlerordnung $\mathcal{O}(\Delta t)$ und die Position mit Fehlerordnung $\mathcal{O}(\Delta t^4)$ bestimmen.

$$\boldsymbol{v}(t_0 + \Delta t) = \frac{\boldsymbol{x}(t_0 + \Delta t) - \boldsymbol{x}(t_0)}{\Delta t} \tag{3.107}$$
$$\boldsymbol{x}(t_0 + \Delta t) = 2\boldsymbol{x}(t_0) - \boldsymbol{x}(t_0 - \Delta t) + \Delta t^2 \boldsymbol{a}(t_0) \tag{3.108}$$

Runge-Kutta-Verfahren 4. Ordnung
Das Runge-Kutta-Verfahren arbeitet sehr exakt und besitzt die Gesamtfehlerordnung $\mathcal{O}(\Delta t^5)$. Bei dieser Methode ist jedoch eine viermalige Berechnung der rechten Seite der Dynamik nötig, was sehr aufwändig ist. Runge-Kutta-Verfahren höherer Ordnung können mit dem Butcher-Schema ermittelt werden [63].

$$\boldsymbol{k}_1 = \Delta t \ \boldsymbol{f}(\tilde{\boldsymbol{x}}_0, t_0) \tag{3.109}$$
$$\boldsymbol{k}_2 = \Delta t \ \boldsymbol{f}(\tilde{\boldsymbol{x}}_0 + \frac{\boldsymbol{k}_1}{2}, t_0 + \frac{\Delta t}{2}) \tag{3.110}$$
$$\boldsymbol{k}_3 = \Delta t \ \boldsymbol{f}(\tilde{\boldsymbol{x}}_0 + \frac{\boldsymbol{k}_2}{2}, t_0 + \frac{\Delta t}{2}) \tag{3.111}$$
$$\boldsymbol{k}_4 = \Delta t \ \boldsymbol{f}(\tilde{\boldsymbol{x}}_0 + \boldsymbol{k}_3, t_0 + \Delta t) \tag{3.112}$$
$$\tilde{\boldsymbol{x}}(t_0 + \Delta t) = \tilde{\boldsymbol{x}}(t_0) + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{3.113}$$

3.4.2 Implizite Integrationsmethoden

Implizite Verfahren haben auf ihrer rechten Gleichungsseite auch eine Abhängigkeit zu Systemzuständen des aktuellen Zeitschritts t_{i+1}. Terme auf der rechten Seite enthalten nun Unbekannte, die erst durch

Umformungen und Lösen eines meist nichtlinearen Gleichungssystems bestimmt werden müssen. Sie arbeiten stabiler als explizite Verfahren und können mit größeren Zeitschrittweiten umgehen. Exemplarisch wird das implizite Euler-Verfahren vorgestellt, das sich nur in der Berechnungsvorschrift für $\boldsymbol{v}(t_i + \Delta t)$ unterscheidet (vgl. Gleichung (3.105) mit (3.114)).

Implizites Euler-Verfahren

Das implizite Euler-Verfahren ist uneingeschränkt stabil. Die Berechnung von Gleichung (3.114) erfordert das Lösen eines Gleichungssystems. Falls $\boldsymbol{v}$ nichtlinear in $\boldsymbol{a}(t)$ vorkommt, ist auch das Gleichungssystem nichtlinear und muss ggf. iterativ gelöst werden, was die Berechnungsdauer des Verfahrens maßgeblich beeinflusst. Das Ergebnis kann dann in die Positionsgleichung (3.115) eingesetzt werden, welche wie im expliziten Fall ausgewertet werden kann.

$$\boldsymbol{v}(t_0 + \Delta t) = \boldsymbol{v}_0 + \Delta t \; \boldsymbol{a}(t_0 + \Delta t) \tag{3.114}$$

$$\boldsymbol{x}(t_0 + \Delta t) = \boldsymbol{x}_0 + \Delta t \; \boldsymbol{v}(t_0 + \Delta t) \tag{3.115}$$

4 Optimierung

Mit der computergestützten Simulation ist ein Werkzeug vorhanden, das viele Probleme von realen experimentellen Versuchen umgeht. In der Forschung und Entwicklung ist es oft nötig natürliche Phänomene und Vorgänge sowie das Verhalten von Systemen und Konstruktionen zu analysieren um sie gegenüber einer Theorie zu verifizieren. Die dazu benötigten Experimente zur empirischen Datenerhebung können sich aber in vielen Bereichen als zu teuer, zu gefährlich oder als unpraktikabel erweisen. Simulationen bieten die Möglichkeit solche Versuche am Computer durchzuführen. Sie bilden neben Theorie und Experiment die so genannte dritte Säule der Wissenschaft. Dabei muss man jedoch stets beachten, dass nur Modelle simuliert werden, die Einschränkungen gegenüber der Realität aufweisen. Die Ergebnisse einer Simulation müssen deshalb stets auf ihre Plausibilität hin überprüft werden. Schlägt diese Verifikation (der Resultate) fehl, ist es nötig ein genaueres Modell zu entwickeln. Kann das Verhalten von Systemen ausreichend genau beschrieben werden, ist es ein weiteres Bestreben sie so zu beeinflussen und zu steuern, dass sie in hohem Maße effizient einem vom Anwender gewünschten Zweck dienen. Das Kriterium, nachdem die Qualität des Systemverhaltens bewertet wird, kann dabei frei gewählt werden. Die Aufgabe der Optimierung ist sicherzustellen dieses Zielkriterium unter Einhaltung eventueller Nebenbedingungen möglichst gut zu erreichen.

4.1 Statische nichtlineare Optimierung

Bei der statischen Optimierung werden Parameter eines Systems gesucht, sodass eine oder mehrere bestimmte Eigenschaften dieses Systems möglichst gut erfüllt werden. Dieser Abschnitt behandelt den Typ von kontinuierlich nichtlinearen Systemen, die nur nach einem Kriterium hin optimiert werden. In der Literatur werden diese Probleme als NLP [1] Probleme bezeichnet. Kontinuierlich bedeutet hierbei, dass die Parameter des Systems jeweils in $\mathbb{R}$ liegen. Zusätzlich können Nebenbedingungen die Wahl der Parameter einschränken, wobei dann von einem restringierten Optimierungsproblem gesprochen wird. Die Grundlagen und im Folgenden verwendeten Begriffe sollen mit Hilfe eines einfachen Beispiels eingeführt werden.

Beispiel 4.1. Bei der Herstellung von Konservendosen, wie sie in der Lebensmittelindustrie üblich sind, sollen Materialkosten minimiert werden. Diese ergeben sich aus dem Verbrauch von Blech, aus dem ein Zylinder geformt wird. Die Oberfläche des Zylinders soll somit minimiert werden. Die mathematische Formel für die Zylinderoberfläche bildet die *Zielfunktion* $\phi(\boldsymbol{p})$ des Optimierungsproblems. Sie ist abhängig von *Parametern* $\boldsymbol{p}$, die bei der Herstellung des Zylinders gewählt werden (s. Abbildung 4.1).

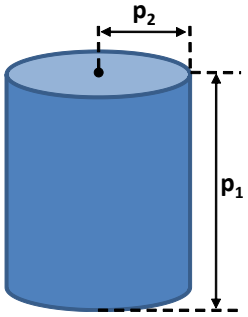


Abbildung 4.1: Höhen- und Radiusparameter des Blechzylinders

[1] engl: Nonlinear Programming

Dies ist der Radius der Grundfläche p_1 und die Höhe des Zylinders p_2. Gesucht sind die optimalen Parameter $\boldsymbol{p}^* = ({p_1}^*, {p_2}^*)^\top \in \mathbb{R}^2$ der Zielfunktion $\phi(\boldsymbol{p})$. An die Herstellung sind Bedingungen geknüpft. Die Produktionsmaschinen können nur ein gewisses Spektrum an Formen produzieren, d.h. die Wahl der Parameter wird durch *explizite Restriktionen* eingeschränkt. Der Radius und die Zylinderhöhe werden also durch obere und untere Schranken begrenzt (upper bounds / lower bounds) $p_i^L \leq p_i \leq p_i^U$ für $i = 1, 2$. Zusätzlich können weitere nichtlineare *Restriktionen* hinzukommen. Die Füllmenge (das Volumen des Zylinders) muss in der Realität häufig einen bestimmten Wert V_0 annehmen. Dies kann über die *Gleichheitsrestriktion* $a(\boldsymbol{p}) = 0$ ausgedrückt werden. Außerdem sind weitere nichtlineare *Ungleichheitsrestriktionen* möglich $b(\boldsymbol{p}) \leq 0$. Die *expliziten Restriktionen* können o.B.d.A. durch *Ungleichheitsrestiktionen* ausgedrückt werden. Das Optimierungsproblem lautet somit:

$$\min_{\boldsymbol{p}\in\mathbb{R}^2} \phi(\boldsymbol{p}) = 2\pi p_1 p_2 + 2\pi p_1^2 \ , \ s.t. \tag{4.1}$$

$$a_1(\boldsymbol{p}) = \pi p_1^2 p_2 - V_0 = 0 \tag{4.2}$$

$$b_1(\boldsymbol{p}) = p_1 - p_1^U \leq 0 \tag{4.3}$$

$$b_2(\boldsymbol{p}) = -p_1 + p_1^L \leq 0 \tag{4.4}$$

$$b_3(\boldsymbol{p}) = p_2 - p_2^U \leq 0 \tag{4.5}$$

$$b_4(\boldsymbol{p}) = -p_2 + p_2^L \leq 0 \tag{4.6}$$

Funktion (4.1) ist die Zielfunktion, die die Zylinderoberfläche beschreibt. Funktion (4.2) ist die Gleichheitsrestriktion zur Einhaltung des gewünschten Zylindervolumens V_0. Die Ungleichheitsrestriktionen (4.3)-(4.5) sind die umgeformten Parameterbeschränkungen. Allgemein hat ein NLP Optimierungsproblem die folgende Form:

$$\min_{\boldsymbol{p}\in\mathbb{R}^{n_p}} \phi(\boldsymbol{p}) \ , s.t. \tag{4.7}$$

$$\boldsymbol{a}(\boldsymbol{p}) = 0 \ , \boldsymbol{a} : \mathbb{R}^{n_p} \mapsto \mathbb{R}^{n_a} \tag{4.8}$$

$$\boldsymbol{b}(\boldsymbol{p}) \leq 0 \ , \boldsymbol{b} : \mathbb{R}^{n_p} \mapsto \mathbb{R}^{n_b} \tag{4.9}$$

Üblicherweise werden Optimierungsprobleme als Minimierungsprobleme formuliert. Sollte ein Maximierungsproblem vorliegen, kann dieses durch ein Vorzeichenwechsel der Zielfunktion $\tilde{\phi}(\boldsymbol{p}) = -\phi(\boldsymbol{p})$ auf ein Minimierungsproblem transformiert werden. Dasselbe gilt für Ungleichheitsrestriktionen, die alle auf eine kleiner-gleich Beziehung gebracht werden können. Außerdem werden die Restriktionen so umgeformt, dass sie auf der rechten Seite den Wert 0 annehmen.
Es gibt eine Vielzahl an verschiedenen Lösungsstrategien um Probleme der Form (4.7)-(4.9) zu lösen. Die Anzahl der Auswertung der Zielfunktion und der nichtlinearen Nebenbedingung verursachen dabei den Großteil des Rechenaufwands. Die Zielfunktion liegt meist nicht in algebraischer Form vor, sondern wird durch eine Simulation beschrieben. Ein prinzipielles Problem, das alle Algorithmen gemein haben, ist die Bestimmung des globalen Optimums. Wurde ein Minimum durch den Algorithmus gefunden, ist nicht bekannt, ob dieses auch das globale Minimum der Zielfunktion ist. Es existiert jedoch eine spezielle Klasse von Problemen, bei denen das einzige Optimum gleichzeitig das globale Optimum ist. Sind Optimierungsprobleme konvex, ist sichergestellt, dass ein gefundenes Minimum das globale Minimum

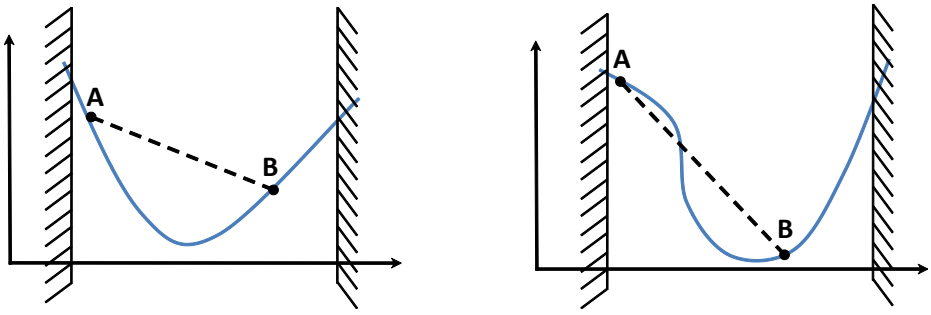
darstellt. Für die Konvexität eines Optimierungsproblems sind zwei Bedingungen zu erfüllen [30]. Eine Zielfunktion heißt konvex, wenn gilt:

$$\phi : [p_1^L, p_1^U] \times \ldots \times [p_{n_p}^L, p_{n_p}^U] \mapsto \mathbb{R} \tag{4.10}$$

$$\phi(\theta \cdot \boldsymbol{p}_A + (1-\theta) \cdot \boldsymbol{p}_B) \leq \theta \cdot \phi(\boldsymbol{p}_A) + (1-\theta) \cdot \phi(\boldsymbol{p}_B) \tag{4.11}$$

$$\forall \boldsymbol{p}_A, \boldsymbol{p}_B \in [p_1^L, p_1^U] \times \ldots \times [p_{n_p}^L, p_{n_p}^U]; \quad \forall \theta \in [0,1] \tag{4.12}$$

Der Term auf der rechten Seite von Ungleichung (4.11) stellt die Geradengleichung zwischen zwei zulässigen Funktionswerten $\phi(\boldsymbol{p}_A)$ und $\phi(\boldsymbol{p}_B)$ dar. Alle anderen Punkte, die von $\phi(\boldsymbol{p}_A)$ und $\phi(\boldsymbol{p}_B)$ begrenzt werden, müssen einen kleineren oder den gleichen Funktionswert der Geraden annehmen (linke Seite von Ungleichung (4.11)). Dieser Sachverhalt ist in Abbildung 4.2 illustriert.



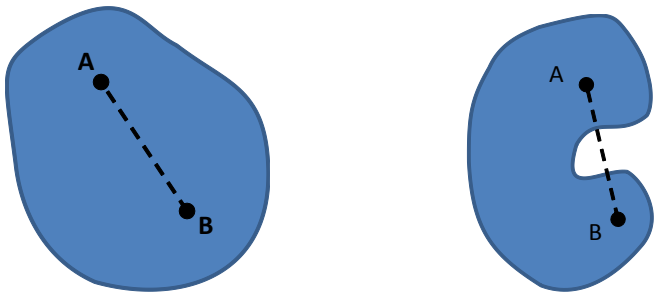
(a) Konvexe Funktion (b) Nicht konvexe Funktion

Abbildung 4.2: Darstellung der Konvexitätsbedingung an die Zielfunktion

Die Menge M aller zulässigen Punkte, also aller Punkte $\boldsymbol{p}$, die die Restriktionen (4.8)-(4.9) des Optimierungsproblems erfüllen, heißt konvex wenn gilt:

$$\boldsymbol{q} = \theta \cdot \boldsymbol{p}_A + (1-\theta) \cdot \boldsymbol{p}_B \in M; \quad \forall \boldsymbol{p}_A, \boldsymbol{p}_B \in M; \quad \forall \theta \in [0,1] \tag{4.13}$$

Dies bedeutet, dass alle Punkte $\boldsymbol{q}$ auf einer Verbindungsstrecke zwischen zwei beliebigen und zulässigen Punkten $\boldsymbol{p}_A$ und $\boldsymbol{p}_B$ wiederum zulässig sein müssen. In Abbildung 4.3 ist dies verdeutlicht.



(a) Konvexe Menge (b) Nicht konvexe Menge

Abbildung 4.3: Darstellung der Konvexitätsbedingung an die Parametermenge

Die genannten Bedingungen an die Zielfunktion ϕ und die zulässige Menge M sind hinreichend, jedoch nicht notwendig für eine eindeutige globale Lösung des Optimierungsproblems. Das bedeutet, es kann durchaus Optimierungsprobleme geben, die diese Bedingungen nicht erfüllen, aber trotzdem nur ein globales Minimum besitzen.

4.1.1 Gradientenbasierte Optimierungsverfahren

Gradientenbasierte Algorithmen [64], [1] nutzen die aus der Analysis bekannte Tatsache, dass Funktionen in einem Minimum stationär werden $\nabla\phi(\boldsymbol{p}) = 0$. Sie analysieren Gradienteninformationen $\nabla\phi$ der Zielfunktion (sowie der Nebenbedingungen ∇a_i, ∇b_j) und bestimmen damit eine Suchrichtung, die zu einer Verringerung des Zielfunktionswerts führt. Allerdings reicht die Bedingung $\nabla\phi(\boldsymbol{p}) = 0$ allein für einen stationären Punkt innerhalb des zulässigen Bereichs nicht aus. Es muss außerdem überprüft werden, ob es sich nicht um ein Maximum oder einen Sattelpunkt handelt. Dazu wird die Krümmungsinformation, die Hessematrix $\nabla^2\phi = H_\phi$, im stationären Punkt untersucht. Ist diese positiv definit, ist die Krümmung um den stationären Punkt konvex und es liegt ein Minimum vor. Es kann allerdings auch der Fall sein, dass das Minimum auf dem Rand des zulässigen Bereichs liegt. Enthält das NLP Problem Gleichheitsrestriktionen $\boldsymbol{a}(\boldsymbol{p})$, gilt dies sogar immer. Für Ungleichheitsrestriktionen $\boldsymbol{b}(\boldsymbol{p})$ kann dies zutreffen oder aber das Minimum liegt komplett im Inneren des zulässigen Bereichs. Um auch Minima am Rand des zulässigen Bereichs finden zu können, wird die *Lagrange-Funktion* (4.14) eingeführt, die neben dem Zielfunktionswert ϕ auch die Werte der Nebenbedingungen $\boldsymbol{a}$ und $\boldsymbol{b}$ berücksichtigt.

$$L(\boldsymbol{p}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \phi(\boldsymbol{p}) + \boldsymbol{\mu} \cdot \boldsymbol{a}(\boldsymbol{p}) + \boldsymbol{\lambda} \cdot \boldsymbol{b}(\boldsymbol{p}) \tag{4.14}$$

$\boldsymbol{\mu}$ und $\boldsymbol{\lambda}$ sind die so genannten *Lagrangemultiplikatoren* für die Gleichheits- und Ungleichheitsrestriktionen. Ist eine Nebenbedingung i für ein Optimum $\boldsymbol{p}^*$ auf dem Rand des zulässigen Bereichs *aktiv*, dann gilt $a_i(\boldsymbol{p}^*) = 0$ bzw. $b_i(\boldsymbol{p}^*) = 0$. Die vollständigen Bedingungen für ein restringiertes Optimierungsproblem bilden die so genannten *Karusch-Kuhn-Tucker* (KKT) Bedingungen [64], [1]. Sie sind im Appendix C zu finden. Sie geben Bedingungen für ein (lokales) Minimum vor, unabhängig davon, ob es sich auf dem Rand oder komplett innerhalb des zulässigen Bereichs befindet. Allerdings gibt es auch Techniken um restringierte Probleme auf unrestringierte Probleme zu transformieren. Damit tritt die Unterteilung in einen zulässigen und unzulässigen Bereich in den Hintergrund. Dabei wird der Funktionswert der Zielfunktion im ehemals unzulässigen Bereich durch Straffunktionen so erhöht, dass er für die Optimierungsalgorithmen möglichst uninteressant als Suchraum und somit vermieden wird. Für Straffunktionsverfahren ist wiederum die alleinige Betrachtung der Bedingungen an die modifizierte Zielfunktion $\nabla\phi(\boldsymbol{p}) = 0$ und $H_\phi \rightarrow$ positiv definit ausreichend. Allerdings kann nicht für alle Straffunktionsverfahren garantiert werden, dass am Ende das gefundene Minimum auch tatsächlich im zulässigen Bereich liegt (vgl. externe und interne Straffunktionsverfahren [64]).

Suchrichtungsverfahren 1. Ordnung:
Suchrichtungsverfahren 1. Ordnung benutzen die erste Ableitung der Zielfunktion, um eine Abstiegsrichtung zu finden, die möglichst effizient und robust den Funktionswert reduziert. Entlang dieser Suchrichtung wird dann eine so genannte Liniensuche [2] durchgeführt um die Schrittweite α zu ermitteln. Das Minimum der Liniensuche bildet den neuen Startpunkt zur Suchrichtungsbestimmung. Somit wird ein mehrdimensionales Optimierungsproblem auf eine Folge von eindimensionalen Liniensuchen reduziert. Die *Methode des Steilsten Abstiegs* [65] nutzt aus, dass der Gradient der Zielfunktion in Richtung des steilsten Aufstiegs zeigt. In Richtung des Gradienten steigt der Funktionswert also am schnellsten. Die Suchrichtung $\boldsymbol{d}^{(k)}$ der Methode liegt dementsprechend genau entgegengesetzt $\boldsymbol{d}^{(k)} = -\nabla\phi(\boldsymbol{p}^{(k)})$. Der Index (k) deutet hierbei den aktuellen Schritt an. Der Nachteil dieser Methode ist, dass alle aufeinander folgenden Suchrichtungen immer senkrecht zueinander stehen. Dadurch kann das Verfahren in der

[2] Für die Liniensuche gibt es spezielle effiziente Verfahren, wie z.B. Intervallschachtelung, die Methode des Goldenen Schnitts, Interpolationsverfahren sowie unvollständige Liniensuche [64].

Nähe eines Minimums (langgestrecktes Tal) sehr langsam werden. Die *Methode der konjugierten Gradienten* [65] kann für quadratische Zielfunktionen das Minimum in n_p Schritten finden. Selbst für nicht quadratische Zielfunktionen ist die Annahme einer quadratischen Funktion in der Nähe des Minimums meist eine gute Approximation, was den Einsatz dieses Verfahrens in vielen Fällen rechtfertigt. Zusammenfassend lassen sich die Methoden 1. Ordnung über die Iterationsformel (4.15) charakterisieren, die sich alle mittels der Bestimmung einer Suchrichtung $\boldsymbol{d}^{(k)}$ und einer Schrittweite $\alpha^{(k)}$ sukzessive durch den Suchraum bewegen.

$$\boldsymbol{p}^{(k+1)} = \boldsymbol{p}^{(k)} + \alpha^{(k)} \cdot \boldsymbol{d}^{(k)} \tag{4.15}$$

Diese Iteration wird solange durchgeführt, bis ein Abbruchkriterium (z.B. die KKT-Bedingungen) erfüllt ist. Je nach Wahl des Kriteriums gilt dann im besten Fall $\boldsymbol{p}^{(k)} = \boldsymbol{p}^*$.

Suchrichtungsverfahren 2. Ordnung:
Bei Methoden 2. Ordnung [64],[1] wird neben dem Gradienten noch zusätzlich die Hessematrix H_ϕ benötigt. Das *Newton-Verfahren* [65] wird ursprünglich zur Nullstellenbestimmung einer Funktion eingesetzt. Wie bereits erwähnt, wird die Zielfunktion im Minimum stationär $\nabla\phi(\boldsymbol{p}^*) = 0$. Somit kann das Newton-Verfahren auch als Optimierungsalgorithmus eingesetzt werden, wobei die Nullstellen der 1. Ableitung der Zielfunktion gesucht werden. Insgesamt ist das Verfahren also von 2. Ordnung. Dabei wird die Ableitung der Funktion (der Gradient der Zielfunktion) durch eine lineare Funktion angenähert und die Nullstelle dieser Näherung bestimmt.

$$\nabla\phi(\boldsymbol{p}) \approx \nabla\tilde{\phi}(\boldsymbol{p}) = \nabla\phi(\boldsymbol{p}^{(k)}) + \nabla^2\phi(\boldsymbol{p}^{(k)})(\boldsymbol{p} - \boldsymbol{p}^{(k)}) \stackrel{!}{=} 0 \tag{4.16}$$

$$\nabla\tilde{\phi}(\boldsymbol{p}) = \nabla\phi(\boldsymbol{p}^{(k)}) + H_\phi(\boldsymbol{p} - \boldsymbol{p}^{(k)}) \stackrel{!}{=} 0 \tag{4.17}$$

$$\tag{4.18}$$

Daraus kann folgende Iterationsvorschrift bestimmt werden.

$$\boldsymbol{p}^{(k+1)} = -H_\phi^{-1} \cdot \nabla\phi(\boldsymbol{p}^{(k)}) + \boldsymbol{p}^{(k)} \tag{4.19}$$

Die Suchrichtung des Verfahrens ergibt sich aus der Differenz aus altem und aktuellem Iterationspunkt.

$$\boldsymbol{d}^{(k)} = \boldsymbol{p}^{(k)} - \boldsymbol{p}^{(k-1)} = -H_\phi^{-1} \cdot \nabla\phi(\boldsymbol{p}^{(k)}) \tag{4.20}$$

Damit kann analog zu den Suchrichtungsverfahren 1. Ordnung die Iterationsvorschrift (4.15) benutzt werden um ein Minimum zu finden. Bei der *Quasi-Newton-Methode* wird der rechenintensive Schritt der Bestimmung der inversen Hessematrix H^{-1} durch ein Näherungsverfahren sukzessive in jedem Iterationsschritt aufgebaut. Hierfür kann z.B. das BFGS-Verfahren [3] [65] eingesetzt werden. Die dabei entstehenden Suchrichtungen sind konjugiert.

SQP - Sequential Quadratic Programming:
SQP Verfahren [9], [1] eignen sich besonders zur effizienten Lösung hochdimensionaler NLP Probleme. Spezielle Implementierungen nutzen dabei dünnbesetzte [4] Gradienten und Jacobimatrizen aus. Es existieren zahlreiche Varianten und Erweiterungen, die die Berechnung des SQP Algorithmuses schneller

[3] Broyden-Fletcher-Goldfarb-Shanno
[4] engl.: sparse

machen. Ausgehend von einer Startschätzung $\boldsymbol{p}^{(k)}$ (mit $k = 0$ im ersten Schritt) werden alle aktiven Restriktionen $a_i(\boldsymbol{p}^{(k)}) = 0$ und $b_j(\boldsymbol{p}^{(k)}) = 0$ bestimmt sowie eine Näherung der Lagrangemultiplikatoren $\mu_i^{(k)}$ und $\lambda_j^{(k)}$. Da nur noch die aktiven Nebenbedingungen betrachtet werden, kann man alle Restriktionen als Gleichheitsnebenbedingung auffassen und einen einheitlichen Lagrangemultiplikator $\boldsymbol{\mu}^{(k)}$ benutzen. Nun wird ähnlich zur (Quasi-)Newton-Methode (s. vorigen Absatz in 4.1.1) verfahren, wobei nun aber die Lagrangefunktion (s. Gleichung (4.14)) verarbeitet wird. Sie wird mit Hilfe einer Taylor-Entwicklung approximiert, für welche die Nullstelle ermittelt werden soll.

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \stackrel{!}{=} \nabla L(\boldsymbol{p}, \boldsymbol{\mu}) \approx \nabla L(\boldsymbol{p}^{(k)}, \boldsymbol{\mu}^{(k)}) + H_{L(\boldsymbol{p},\boldsymbol{\mu})}(\boldsymbol{p}^{(k)}, \boldsymbol{\mu}^{(k)}) \cdot \begin{pmatrix} \boldsymbol{d}_p \\ \boldsymbol{d}_\mu \end{pmatrix} \tag{4.21}$$

Analog zu den Schritten in Gleichungen (4.19) und (4.20) kann daraus eine Suchrichtung bestimmt werden, was über die Lösung des folgenden Gleichungssystems nach $(\boldsymbol{d}_p, \boldsymbol{d}_\mu)^\top$ geschieht.

$$H_{L(\boldsymbol{p},\boldsymbol{\mu})}(\boldsymbol{p}^{(k)}, \boldsymbol{\mu}^{(k)}) \cdot \begin{pmatrix} \boldsymbol{d}_p \\ \boldsymbol{d}_\mu \end{pmatrix} = -\nabla L(\boldsymbol{p}^{(k)}, \boldsymbol{\mu}^{(k)}) \tag{4.22}$$

Statt dieses Gleichungssystem direkt zu lösen, kann daraus auch ein äquivalentes Optimierungsproblem formuliert werden, was die spezielle Gestalt eines quadratischen Problems (QP) besitzt. Die Bezeichnung SQP geht also auf das iterative Lösen quadratischer Optimierungsprobleme zurück. Das QP zur Bestimmung der Suchrichtung lautet:

$$\min_{\boldsymbol{d}_p \in \mathbb{R}^{n_p}} \phi(\boldsymbol{p}^{(k)}) + \left(\nabla \phi(\boldsymbol{p}^{(k)})\right)^\top \cdot \boldsymbol{d}_p + \frac{1}{2} \boldsymbol{d}_p^\top \cdot H_L(\boldsymbol{p}^{(k)}, \boldsymbol{\mu}^{(k)}) \cdot \boldsymbol{d}_p \quad s.t., \tag{4.23}$$

$$\boldsymbol{a}(\boldsymbol{p}^{(k)}) + J_a^\top(\boldsymbol{p}^{(k)}) \cdot \boldsymbol{d}_p = 0 \tag{4.24}$$

Die Ermittlung der Suchrichtung über das QP ist wesentlich robuster als das Lösen des Gleichungssystems (4.22), weshalb auf diese Methode für das SQP Verfahren zurückgegriffen wird. Ist die Suchrichtung bestimmt, wird als nächstes die Schrittweite $\alpha^{(k)}$ mit Hilfe einer Testfunktion (merit function) ermittelt. Auch hier wird nun wieder solange die Iterationsvorschrift (4.15) angewendet, bis ein Abbruchkriterium erfüllt ist.

Innere-Punkt-Verfahren

Innere-Punkt-Verfahren benutzen eine spezielle Klasse von Straffunktionen. Sie sorgen dafür, dass die einzelnen Optimierungschritte im Inneren des zulässigen Bereichs verlaufen, indem sie die Ungleichheitsrestriktionen durch einen logarithmischen Barriereterm [64] zur Zielfunktion hinzufügen. Sie wurden anfangs für konvexe Probleme eingesetzt. Aktuelle Varianten können aber auch für allgemeine NLP Probleme benutzt werden. Das NLP Problem (4.7)-(4.9) wird umformuliert auf [66]:

$$\min_{\boldsymbol{p} \in \mathbb{R}^n} \phi_\mu(\boldsymbol{p}) = \phi(\boldsymbol{p}) - \mu \sum_{i=1}^{n_p} \ln(p_i) \; , s.t. \tag{4.25}$$

$$c(\boldsymbol{p}) = 0 \tag{4.26}$$

Die modifizierte Zielfunktion (4.25) hängt nun von einem Parameter μ ab, der den Verlauf der Barrierefunktion steuert. Offensichtlich stimmen das ursprüngliche Optimum $\boldsymbol{p}^*$ mit dem Optimum aus (4.25) um so besser überein, je mehr $\mu \to 0$. Es wird deshalb eine Serie von Optimierungen durchgeführt,

bei der μ stetig reduziert wird und das Ergebnis des vorangegangenen Laufs den Startwert des aktuellen bildet. Dies wird solange durchgeführt, bis die notwendigen Bedingungen für das Barriereproblem (4.25)-(4.26) erfüllt sind [66].

$$\nabla f(\boldsymbol{p}) + \nabla c(\boldsymbol{p})\boldsymbol{y} - \boldsymbol{z} = 0 \tag{4.27}$$
$$c(\boldsymbol{p}) = 0 \tag{4.28}$$
$$XZe - \mu e = 0 \tag{4.29}$$
$$\boldsymbol{p}, \boldsymbol{z} \geq 0 \tag{4.30}$$
$$\tag{4.31}$$

Hier sind $y \in \mathbb{R}^m$ und $z \in \mathbb{R}^n$ die Langrangemultiplikatoren für die Gleichheits- und expliziten Restriktionen. Außerdem gilt $X = Diag(\boldsymbol{p})$, $Z = Diag(\boldsymbol{z})$ und $e = (1, \ldots, 1)^\top$. Die einzelnen Optimierungsläufe mit festem μ lassen sich wiederum mit einem Newton-Typ-Verfahren durchführen.

4.1.2 Gradientenfreie Optimierungsverfahren

Gradientenfreie Optimierungsverfahren [1] eignen sich, falls die Berechnung von Ableitungen nicht möglich, ihre numerische Approximation zu aufwendig, ungenau oder rechenintensiv ist. Verfahren der gradientenfreien Optimierung arbeiten meist langsamer und ungenauer, da sie die zusätzlichen Informationen, die den gradientenbasierenden Verfahren vorliegen, nicht benutzen können. Deshalb werden sie meist nur in speziellen Szenarien eingesetzt.

Methoden 0. Ordnung
Methoden dieser Ordnung benutzen nur die Funktionswerte von ϕ und nicht deren Ableitungen. Auch die Restriktionen $\boldsymbol{a}$ und $\boldsymbol{b}$ werden nur direkt ausgewertet und nicht differenziert. Das Nelder-Mead-Simplex-Verfahren [65] generiert eine geometrische Struktur, einen Simplex, der sich nach bestimmten Regeln aufgrund der Funkionswerte an seinen Eckpunkten durch den Optimierungsraum bewegt. Auch Mustersuchverfahren wie das Asynchronous-Parallel-Pattern-Search (APPS) [1] tasten den Suchraum nach gewissen Richtlinien ab und bilden dabei spezielle Formen. Andere Verfahren, z.B. Dividing Rectangulars (DIRECT) [65], zerteilen den Suchraum und verfeinern eine bestimmte Auswahl der dabei entstehenden Zellen, die günstige Eigenschaften besitzen. Allgemein kann die Konvergenz in ein Minimum dieser Verfahrensklasse nicht sichergestellt werden, was ein entscheidender Nachteil ist.

Nichtdeterministische Suchverfahren
Eine spezielle Klasse dieser Optimierungsmethoden sind die nichtdeterministischen Suchverfahren [64], [1]. Sie besitzen alle eine Zufallskomponente in ihrem Algorithmus. Das bedeutet, dass der Verlauf der Optimierung nicht vorhersehbar ist und mehrmaliges Anwenden auf dieselbe Problemstellung unterschiedliche Optimierungsverläufe ergeben kann. Somit sind auch unterschiedliche gefundene Minima als Ergebnis der Optimierung möglich. Ihr großer Vorteil liegt in der Eigenschaft globale Optima aufspüren zu können. Klassische Optimierungsverfahren verfolgen eine „greedy“-Strategie und akzeptieren in jedem Schritt nur gleichbleibende oder bessere Funktionswerte. So konvergieren sie auf direktem Weg in das nächstgelegene lokale Minimum. Die Vorgabe guter Startwerte nahe am globalen Optimum ist also für diese Verfahrenstypen essentiell. Dies ist in der Praxis aber ein schwieriges Problem. Nichtdeterministische Verfahren können aufgrund ihrer Zufallskomponente in manchen Schritten auch schlechtere Funktionswerte akzeptieren. Befinden sie sich in einem lokalen Minimum, besitzen sie die Chance aus diesem herauszukommen und daraufhin in ein anderes Minimum zu konvergieren. Am Ende wird der

bisher beste Wert ausgegeben, der möglicherweise das globale Optimum ist. Evolutionäre und genetische Algorithmen [64] sind bekannte Verfahren dieser Klasse. Sie codieren mehrere Punkte $\boldsymbol{p}_i$ des Parameterraums, die sich rekombinieren und Nachkommen erzeugen können. Die Zufallskomponente wird entweder über die Auswahl der Individuen für den nächsten Optimierungsschritt (genetisch) oder über Rekombinations- und Mutationsmethoden (evolutionär) implementiert. Das Particle-Swarm-Verfahren [64] basiert ebenfalls auf in der Natur vorkommenden Prinzipien. Einzelne Partikel durchsuchen den Optimierungsraum und merken sich ihren eigenen bisher besten Parameter $\boldsymbol{p}$. Sie tauschen aber auch Informationen untereinander aus und kennen somit den besten Parameter $\boldsymbol{p}^g$ des ganzen Schwarms. Sie passen ihre Suchrichtung gemäß dieser Informationen an und imitieren somit das Verhalten einer sozial kooperierenden Gruppe.

4.2 Dynamische nichtlineare Optimierung

Im Gegensatz zur statischen Optimierung wird bei der dynamischen Optimierung kein einzelner optimaler Parameter $\boldsymbol{p}^*$ einer Funktion $\phi(\boldsymbol{p})$ gesucht, sondern vielmehr eine optimale Funktion $\boldsymbol{x}^*(t)$. Die Unbekannte ist also eine Funktion, die von einer unabhängigen Variablen t abhängt. Die dabei zu minimierende Zielfunktion ist vom Typ eines *Gütefunktionals* $J[x] : D \subseteq \mathbb{R}^{n_x} \mapsto \mathbb{R}$ (s. Abschnitt 3.3.1). Häufig wird auch von einem unendlichdimensionalen Optimierungsproblem gesprochen, da $\boldsymbol{x}^*(t)$ für jeden beliebigen reellen Wert t innerhalb der Schranken $t = 0$ und $t = t_f$ optimal sein muss. Die notwendigen Bedingungen an eine Lösung können mit Hilfe der Variationsrechnung hergeleitet werden. Wie bereits in Abschnitt 3.3.1 gezeigt, kann die Minimierung des Funktionals $J[\boldsymbol{x}(t)]$ in ein Variationsproblem der Form

$$J[\boldsymbol{x}(t)] = \int_0^{t_f} L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\, dt \rightarrow \text{EXTR! (MIN!)} \tag{4.32}$$

überführt werden. Die Lösung wird mit Hilfe der ersten und zweiten Variation (s. Abschnitte in 3.3.1) auf das Randwertproblem der Euler-Lagrangeschen Differentialgleichung (3.74) umgeformt.

4.2.1 Variationsprobleme

Dieser Abschnitt zeigt die notwendigen Bedingungen 1. Ordnung für verallgemeinerte Variationsprobleme, die über das Basisproblem (4.32) hinausgehen und schließlich zu den Optimalsteuerungsproblemen führen.

Variationsproblem mit natürlicher Randbedingung

Ein Variationsproblem mit natürlichen Randbedingungen ist mindestens an einem Rand unbeschränkt, d.h. es stehen nun nicht mehr nur alle stetig differenzierbaren Kurven, die durch zwei fest vorgegebene Punkte $\boldsymbol{x}(0) = \boldsymbol{x}_0$ und $\boldsymbol{x}(t_f) = \boldsymbol{x}_{t_f}$ verlaufen, zur Auswahl, sondern auch solche, die sich frei auf dem jeweiligen Rand bewegen können (s. Abbildung 4.4). Die notwendigen Bedingungen für ein Variationsproblem der Art (4.32) mit einer zusätzlichen natürlichen Randbedingung bei $\boldsymbol{x}(t_f) = \text{FREI}$ sind [61]

$$\frac{d}{dt}\frac{d}{d\dot{x}_i}L(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) - \frac{d}{dx_i}L(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) = 0 \quad \text{(Euler-Lagrangesche Differentialgleichung)} \tag{4.33}$$

$$\frac{d}{d\dot{x}_i}L(\boldsymbol{x}(t_f), \dot{\boldsymbol{x}}(t_f), t_f) = 0 \quad \text{(natürliche Randbedingung)} \tag{4.34}$$

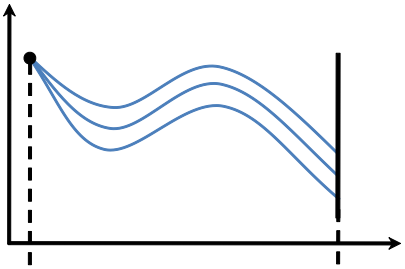


Abbildung 4.4: Bei der natürlichen Randbedingung ist der Funktionswert $\boldsymbol{x}(t_f)$ nicht vorgegeben.

Ist der andere Rand $\boldsymbol{x}(0)$ des Variationsproblems frei, dann wird die zweite Bedingung (4.34) für $t = 0$ statt für $t = t_f$ formuliert. Sind beide Ränder frei, müssen zwei Bedingungen vom Typ (4.34) erfüllt werden.

Variationsproblem mit Transversalitätsbedingung

Lässt sich z.B. der rechte Randpunkt auch in t-Richtung bewegen, entsteht das Variationsproblem (4.35).

$$J[\boldsymbol{x}(t)] = \int_0^{t_f} L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\, dt \rightarrow \text{EXTR!} \quad \boldsymbol{x}(0) = \boldsymbol{x}_0,\ \boldsymbol{x}(t_f) = \boldsymbol{f}(t) \tag{4.35}$$

Neben der Euler-Lagrangeschen Differentialgleichung (3.74) muss dann auch die so genannte Transversalitätsbedingung für den jeweiligen Rand erfüllt sein. Die notwendigen Bedingungen dafür sind [61]:

$$\frac{d}{dt}\frac{d}{d\dot{x}_i}L(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) - \frac{d}{dx_i}L(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) = 0 \quad \text{(Euler-Lagrangesche Differentialgleichung)} \tag{4.36}$$

$$\frac{d}{d\dot{x}_i}L + \frac{L}{\dot{f}_i - \dot{x}_i} = 0, \quad \text{in } t = t_f \quad \text{(Transversalitätsbedingung)} \tag{4.37}$$

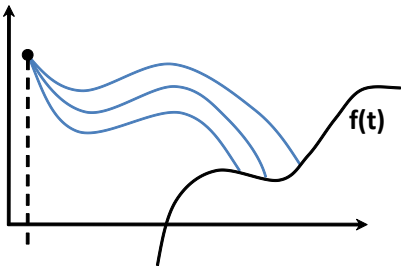


Abbildung 4.5: Der rechte Randpunkt hängt von der Transversalitätsbedingung $\boldsymbol{f}(t)$ ab.

Variationsproblem mit Integralnebenbedingung

Variationsprobleme mit Nebenbedingung in Integralform lassen sich wie folgt beschreiben:

$$J[\boldsymbol{x}(t)] = \int_0^{t_f} L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\, dt \rightarrow \text{EXTR!} \tag{4.38}$$

$$R(\boldsymbol{x}(t)) = \int_0^{t_f} G(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\, dt = c\ (= const.) \tag{4.39}$$

Probleme dieser Klasse sind als Isoperimetrische Probleme bekannt. Auch das in Abschnitt 2.1 beschriebene Problem der Dido fällt in diese Kategorie. Ähnlich wie in der statischen Optimierung lassen sich

Nebenbedingungen über Lagrangemultiplikatoren zur Zielfunktion dazukoppeln (s. Abschnitt 4.1.1). Es entsteht die Lagrange-Funktion $\widetilde{L}$, die im Funktional (4.38) ersetzt wird.

$$\widetilde{L} := L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) + \boldsymbol{\lambda} \cdot G(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) \tag{4.40}$$

Durch die 1. Variation der so modifizierten Zielfunktion ergibt sich die notwendige Bedingung an die Lösung [61].

$$\frac{d}{dt}\frac{d}{d\dot{x}_i}\widetilde{L}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) - \frac{d}{dx_i}\widetilde{L}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) = 0 \quad \text{(E.-L. DGL der Lagrange-Fkt. } \widetilde{L}) \tag{4.41}$$

Variationsproblem mit Gleichungsnebenbedingung

Variationsprobleme mit Nebenbedingung in Gleichungsform lassen sich wie folgt beschreiben:

$$J[\boldsymbol{x}(t)] = \int_0^{t_f} L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\, dt \rightarrow \text{Extr!} \tag{4.42}$$

$$G(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) = 0, \quad 0 \leq t \leq t_f \tag{4.43}$$

Die Lagrange-Funktion $\widehat{L}$ wird dann nicht mit einem konstanten Lagrange-Multiplikator $\boldsymbol{\lambda}$, sondern mit einem zeitabhängigen, der *adjungierten Variablen* $\boldsymbol{\lambda}(t)$, gebildet.

$$\widehat{L} := L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) + \boldsymbol{\lambda}(t) \cdot G(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) \tag{4.44}$$

Die notwendige Bedingung dieser Lagrange-Funktion entspricht der Bedingung (4.41) [61].

$$\frac{d}{dt}\frac{d}{d\dot{x}_i}\widehat{L}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) - \frac{d}{dx_i}\widehat{L}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) = 0 \quad \text{(E.-L. DGL der Lagrange-Fkt. } \widehat{L}) \tag{4.45}$$

Ist die Nebenbedingung $G(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) = 0$ in Differentialgleichungsform (oder ist ein DAE [5]), spricht man von einem Optimalsteuerungsproblem, welche im nächsten Abschnitt genauer betrachtet werden.

4.2.2 Optimale Steuerung

Optimalsteuerungsprobleme bestitzen eine über die Funktion $\boldsymbol{u}(t)$ beinflussbares Systemverhalten, das in Form von Differentialgleichungsnebenbedingungen $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ vorliegt. Es kann aufgrund der in Abschnitt 3.4 gezeigten Transformation immer von einem DGL-System 1. Ordnung ausgegangen werden. Ist ein dynamisches System $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t))$ nicht steuerbar, kann es nur zu Beginn durch seine Startwerte $\boldsymbol{x}(0) = \boldsymbol{x}_0$ und $\dot{\boldsymbol{x}}(0) = \dot{\boldsymbol{x}}_0$ beeinflusst werden, was es für die Behandlung mittels Methoden der Optimalen Steuerung uninteressant macht. Optimalsteuerungsprobleme können durch das folgende Schema dargestellt werden:

[5] engl.: Differential Algebraic Equation

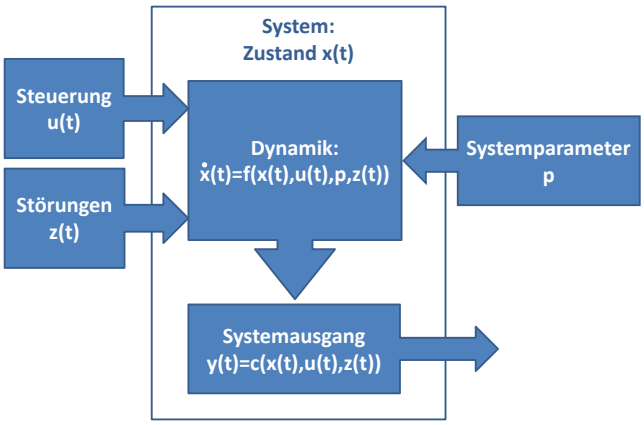


Abbildung 4.6: Schematischer Aufbau eines Optimalsteuerungsproblems

Gegeben sind die Anfangs- und Endbedingung $(\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0), (\boldsymbol{x}_{t_f}, \dot{\boldsymbol{x}}_{t_f})$ und ggf. die für den Simulationslauf konstanten Systemparameter $\boldsymbol{p}$. Gesucht sind die Verläufe der Steuergröße $\boldsymbol{u}(t)$ und des Systemzustands $\boldsymbol{x}(t)$ zur Simulationszeit $0 \leq t \leq t_f$. In einem allgemeineren Kontext muss davon ausgegangen werden, dass das System nicht modellierten Störgrößen $\boldsymbol{z}(t)$ unterliegt und sein Zustand nicht direkt beobachtbar ist, sondern nur über einen Systemausgang $\boldsymbol{y}(t)$ gemessen werden kann. Zunächst wird das Basisproblem der Optimalen Steuerung [1] behandelt.

Basisproblem der Optimalen Steuerung

$$\min_{\boldsymbol{u}(t)\in\mathbb{R}^{n_u}} J[\boldsymbol{u}(t)] = \phi(\boldsymbol{x}(t_f), t_f) + \int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt \qquad \text{(Zielfunktion)} \qquad (4.46)$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad 0 \leq t \leq t_f \qquad \text{(Systemdynamik)} \qquad (4.47)$$

$$x_i(0) = x_{i,0}\ (= const.), \quad i = 1, \ldots, n_x \qquad \text{(Anfangsbedingungen)} \qquad (4.48)$$

$$\boldsymbol{r}(\boldsymbol{x}(t_f), t_f) = 0 \quad \text{z.B. } x_j(t_f) = x_{j,t_f} \quad \boldsymbol{r} \in \mathbb{R}^{n_r} \qquad \text{(Endbedingungen)} \qquad (4.49)$$

$$t_f = \text{FEST} \quad \text{oder} \quad t_f = \text{FREI} \qquad \text{(Endzeit)} \qquad (4.50)$$

Das erweiterte Funktional des Basisproblems

Die zu minimierende Zielfunktion (4.46) hat die Form eines *Bolza-Funktionals*. Sie besteht aus dem *Mayer-Term* $\phi(\boldsymbol{x}(t_f), t_f)$, der die Kosten zum Endzeitpunkt angibt, und dem *Lagrange-Term*, der die Kosten $\int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt$ während des Verlaufs widerspiegelt. Analog zur Lagrangefunktion bei der statischen Optimierung können Nebenbedingungen mittels Multiplikatoren an die Zielfunktion gekoppelt werden, um ein Verletzen der Restriktion mit den Kosten zu verknüpfen. Im Fall der Dynamiknebenbedingung (4.47) ist der Multiplikator $\boldsymbol{\lambda}(t)$ nun nicht mehr konstant, sondern eine Funktion und wird *adjungierte Variable* genannt, was bereits im Abschnitt 4.2.1 für das Variationsproblem mit Gleichungsnebenbedingung angewandt wurde. Daraus resultiert die *Hamilton-Funktion*.

$$H(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) = L(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \sum_{i=1}^{n_x} \lambda(t)_i \cdot f_i(\boldsymbol{x}(t), \boldsymbol{u}(t)) = L(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{\lambda}(t)^\top \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad (4.51)$$

Nicht nur die Dynamiknebenbedingung, sondern auch die Endbedingung (4.49) kann verletzt werden. Da diese nur zu einem Zeitpunkt, dem Endzeitpunkt, auftritt und nicht kontinuierlich ist, kann sie über

den konstanten Multiplikator $\boldsymbol{\nu}$ an den Mayer-Term gekoppelt werden. Daraus resultiert die Hilfsfunktion.

$$\Phi(\boldsymbol{x}(t), t, \boldsymbol{\nu}) = \phi(\boldsymbol{x}(t), t) + \sum_{i=1}^{n_r} \nu_i \cdot r_i(\boldsymbol{x}(t), t) = \phi(\boldsymbol{x}(t), t) + \nu^\top \boldsymbol{r}(\boldsymbol{x}(t), t) \tag{4.52}$$

Mit den soeben definierten Funktionen (4.51) und (4.52) kann das erweiterte Funktional erstellt werden.

$$J[\boldsymbol{u}(t)] = \Phi(\boldsymbol{x}(t_f), t_f, \boldsymbol{\nu}) + \int_0^{t_f} H(\boldsymbol{x}(t), \boldsymbol{u}(t)) - \boldsymbol{\lambda}(t)^\top \dot{\boldsymbol{x}}(t)\, dt \tag{4.53}$$

$$= \phi(\boldsymbol{x}(t_f), t_f) + \boldsymbol{\nu}^\top \boldsymbol{r}(\boldsymbol{x}(t_f), t_f) + \int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{\lambda}(t)^\top (\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) - \dot{\boldsymbol{x}}(t))\, dt \tag{4.54}$$

Notwendige Bedingungen 1. Ordnung an das Basisproblem

Aus der 1. Variation des erweiterten Funktionals (4.54) ergeben sich die folgenden notwendigen Bedingungen an die Lösung $\boldsymbol{u}^*(t), \boldsymbol{x}^*(t), \boldsymbol{\lambda}^*(t)$ des Basisoptimalsteuerungsproblems [1].

$$\frac{\partial H}{\partial u_i} = 0 \text{ (Steuerungsbedingung)} \tag{4.55}$$

$$\dot{\boldsymbol{\lambda}}^*(t) = -\frac{\partial H}{\partial x_i} = -\frac{\partial L(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))}{\partial x_i} - \sum_{k=1}^{n_x} \lambda_k \frac{\partial f_k(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))}{\partial x_i} \text{ (E.-L. DGL i)} \tag{4.56}$$

$$\dot{\boldsymbol{x}}^*(t) = +\frac{\partial H}{\partial \lambda_i} = f_i(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)) \text{ (E.-L. DGL ii)} \tag{4.57}$$

$$\left.\begin{array}{rl} \boldsymbol{x}^*(t_0) = \boldsymbol{x}_0, & \text{fester Rand} \\ \boldsymbol{\lambda}^*(t_0) = 0, & \text{natürliche Randbedingung} \end{array}\right\} \text{(Anfangsbedingung)} \tag{4.58}$$

$$\left.\begin{array}{rl} \boldsymbol{x}^*(t_f) = \boldsymbol{x}_{t_f}, & \text{fester Rand} \\ \boldsymbol{\lambda}^*(t_f) = \frac{\partial \phi(\boldsymbol{x}^*(t_f), t_f)}{\partial x_i^*(t_f)}, & \text{natürliche Randbedingung} \end{array}\right\} \text{(Endbedingung)} \tag{4.59}$$

$$\left[\phi_{t_f} + H(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t))\right]_{t=t_f} = 0 \text{ (Randbedingung für } t_f = \text{FREI)} \tag{4.60}$$

Maximumsprinzip für das Basisproblem

Die Optimalitätsbedingung an die Steuerung (4.55) ist nicht allgemein genug und deckt nicht alle in der Realität auftretenden Fälle ab. Üblicherweise wird die Ableitung $\frac{\partial H}{\partial u_i}$ berechnet und das Ergebnis von $\frac{\partial H}{\partial u_i} = 0$ nach $\boldsymbol{u}(t)$ umgestellt. Die Steuerung ist dann in Abhängigkeit von $\boldsymbol{x}(t)$ und $\boldsymbol{\lambda}(t)$ gegeben und kann in den Bedingungen (4.56) und (4.57) (Euler-Lagrangesche Differentialgleichungen) eingesetzt werden. Das daraus resultierende Zweipunkt-Randwertproblem kann mit den Anfangs- und Endbedingungen gelöst werden. Tritt allerdings in der Hamiltonfunktion H die Steuerung $\boldsymbol{u}(t)$ nur linear auf, enthält die Ableitung $\frac{\partial H}{\partial u_i}$ die Variable $\boldsymbol{u}(t)$ nicht mehr. Die Steuerung kann somit nicht aus Bedingung (4.55) bestimmt werden. Das Maximumsprinzip [2],[3], das unter anderem auf Pontryagin zurückgeht, liefert eine allgemeinere Bedingung an die Steuerung $\boldsymbol{u}(t)$. Es wird hier als Minimumsprinzip definiert. Ist $\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t)$ die optimale Lösung des Basisproblems dann gilt:

$$\boldsymbol{u}^*(t) = \min_{\boldsymbol{u}(t)} H(\boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t)) \tag{4.61}$$

Tritt nun $\boldsymbol{u}(t)$ linear in H auf, kann die Hamiltonfunktion wie folgt formuliert werden:

$$H(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\lambda}) = \sum_{i=1}^{n_x} s_i(\boldsymbol{x},\boldsymbol{\lambda}) \cdot u_i + a(\boldsymbol{x},\boldsymbol{\lambda}) \tag{4.62}$$

Über die Schaltfunktion $s_i(\boldsymbol{x},\boldsymbol{\lambda})$ lassen sich nun die Steuerungswerte $u_i(t)$ bestimmen.

$$s_i(\boldsymbol{x}(t),\boldsymbol{\lambda}(t)) > 0 \Rightarrow u_i(t) = u_{i,min} \tag{4.63}$$
$$s_i(\boldsymbol{x}(t),\boldsymbol{\lambda}(t)) < 0 \Rightarrow u_i(t) = u_{i,max} \tag{4.64}$$
$$s_i(\boldsymbol{x}(t),\boldsymbol{\lambda}(t)) = 0 \Rightarrow u_i(t) = \text{SINGULÄR} \tag{4.65}$$

Im Basisproblem ist die Steuerung unbeschränkt, sodass $u_{i,min} = -\infty$ und $u_{i,max} = +\infty$ gilt. In der Realität ist allerdings die Steuerung beschränkt, was im allgemeinen Optimalsteuerungsproblem (4.68)-(4.75) berücksichtigt werden kann. Wechselt die Schaltfunktion nur zwischen $s_i > 0$ ((4.63)) und $s_i < 0$ ((4.64)), wird auch von einer *Bang-Bang Steuerung* gesprochen.

Singuläre Steuerung

Wird die Steuerung in einem Zeitabschnitt $t_a \leq t \leq t_b$ singulär, können Informationen über ihren Verlauf durch die totale Ableitung von $s_i(\boldsymbol{x}(t),\boldsymbol{\lambda}(t))$ bestimmt werden. Es gilt:

$$\frac{d}{dt} s_i(\boldsymbol{x}(t),\boldsymbol{\lambda}(t)) = \frac{\partial s_i}{\partial \boldsymbol{x}}^{\top} \cdot \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t)) - \frac{\partial s_i}{\partial \boldsymbol{\lambda}}^{\top} \cdot \frac{\partial H}{\partial x} \equiv 0. \tag{4.66}$$

Die Schaltfunktion wird nun so oft nach der Zeit abgeleitet, bis die Steuervariable u_i explizit auftritt.

$$\frac{d^k}{dt^k} s_i(\boldsymbol{x}(t),\boldsymbol{\lambda}(t)) = 0 \quad \rightarrow u_i(t) \tag{4.67}$$

Ist dies in Schritt k der Fall, so ist k eine gerade Zahl. Die Ordnung p der singulären Steuerung wird mit $k = 2p$ definiert. Über die Ordnung lässt sich bestimmen, wie die Steuerung u_i zwischen einem singulären Intervall $t_a \leq t \leq t_b$ und einem angrenzenden Bang-Bang Intervall $t \leq t_a$ oder $t \geq t_b$ verläuft [1]. Ist p ungerade, ergibt sich für u_i ein stetig differenzierbarer Übergang an der Intervallgrenze. Ist p gerade, ist der Übergang an dieser Stelle nur stetig in u_i.

Allgemeines Optimalsteuerungsproblem

$$\min_{\boldsymbol{u}(t)\in\mathbb{R}^{n_u}} J[\boldsymbol{u}(t)] = \phi(\boldsymbol{x}(t_f),t_f) + \int_0^{t_f} L(\boldsymbol{x}(t),\boldsymbol{u}(t))\,dt \quad \text{(Zielfunktion)} \tag{4.68}$$
$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t)), \qquad 0 \leq t \leq t_f \quad \text{(Systemdynamik)} \tag{4.69}$$
$$x_i(0) = x_{i,0}\ (= const.), \quad i = 1,\ldots,n_x \quad \text{(Anfangsbedingungen)} \tag{4.70}$$
$$\boldsymbol{r}(\boldsymbol{x}(t_f),t_f) = 0 \quad \text{z.B. } x_j(t_f) = x_{j,t_f} \quad \boldsymbol{r} \in \mathbb{R}^{n_r} \quad \text{(Endbedingungen)} \tag{4.71}$$
$$t_f = \text{FEST} \quad \text{oder} \quad t_f = \text{FREI} \quad \text{(Endzeit)} \tag{4.72}$$
$$\boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t)) \geq 0 \quad \text{(Steuerungsbeschränkungen)} \tag{4.73}$$
$$\boldsymbol{g}(\boldsymbol{x}(t)) \geq 0 \quad \text{(Zustandsbeschränkungen)} \tag{4.74}$$
$$\boldsymbol{r}^{(i)}(\boldsymbol{x}(t_s-0),\boldsymbol{x}(t_s+0),t_s) = 0 \quad \text{(Innere-Punkt-Bedingungen)} \tag{4.75}$$

Falls $\boldsymbol{u}(t)$ explizit in einer Beschränkung $\boldsymbol{g}$ auftritt, liegt eine Steuerungsbeschränkung (4.73) vor. Tritt $\boldsymbol{u}(t)$ nicht explizit in $\boldsymbol{g}$ auf, liegt eine reine Zustandsbeschränkung (4.74) vor. Diese sind wesentlich schwieriger zu behandeln. In beiden Fällen sind modifizierte und neue hinzukommende notwendige Bedingungen zu erfüllen. Hierbei sei auf [3], [2], [1] verwiesen. Analog zum Basisfall kann das erweiterte Funktional erstellt werden, mit dessen Hilfe diese Bedingungen hergeleitet werden können. Allerdings werden die neuen Beschränkungen $\boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t))$ über einen zusätzlichen Multiplikator $\boldsymbol{\mu}(t)$ an die *erweiterte Hamilton-Funktion* gekoppelt.

$$\tilde{H}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\lambda}(t),\boldsymbol{\mu}(t)) = L(\boldsymbol{x}(t),\boldsymbol{u}(t)) + \boldsymbol{\lambda}(t)^\top \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t)) + \boldsymbol{\mu}(t)^\top \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t)) \tag{4.76}$$

Der zeitliche Verlauf wird in Intervalle aktiver ($g_i = 0$) und nicht aktiver Restriktionen ($g_i > 0$) eingeteilt. Ist eine Restriktion in einem Bereich inaktiv, so gilt dort $\mu_i(t) = 0$ und die gezeigten notwendigen Bedingungen (4.55)-(4.60) bestehen weiterhin. Ist eine Restriktion in einem Bereich $t_a \leq t \leq t_b$ aktiv, wird die Steuerung bzw. der Systemzustand durch den Rand der Restriktion festgelegt. In diesem Fall werden die notwendigen Bedingungen angepasst und neue Schalt- und Sprungbedingungen kommen hinzu.

Beispiel 4.2. Engergieoptimale Dämpfung eines Einmassenschwingers
Der Einmassenschwinger, dessen Dynamik in Beispiel 3.2 gezeigt wurde, soll nun bezüglich einer gegebenen Anfangsauslenkung gedämpft werden. Dabei soll innerhalb eines definierten Zeitrahmens möglichst wenig Energie benutzt werden. Die zum Dämpfen verwendete Energie hängt nur von dem Betrag bzw. dem Quadrat der Steuervariablen ab, weshalb $J[u] = 0 + \int_0^{t_f} u(t)^2 \, dt$ ein passendes Kriterium des Optimalsteuerungsproblems ist. Weitere Kosten zum Endzeitpunkt t_f entstehen nicht, weswegen der Mayer-Term $= 0$ ist. Die Systemdynamik kann, wie in Beispiel 3.4 gezeigt, auf ein System 1. Ordnung transformiert werden. Die ausgelenkte Startkonfiguration der Masse wird durch die Randwerte zur Startzeit $t_0 = 0$ vorgegeben, die gewünschte Endkonfiguration durch die Bedingungen zur vorgegebenen Endzeit t_f. Das konkrete Optimalsteuerungsproblem, das im Folgenden behandelt werden soll, lautet:

$$\min_{u(t)\in\mathbb{R}} J[u(t)] = 0 + \frac{1}{2}\int_0^{t_f} u(t)^2 \, dt \tag{4.77}$$

$$\dot{x}_1(t) = x_2(t), \quad \dot{x}_2(t) = -\frac{k}{m}x_1 + \frac{1}{m}u(t)\,, \quad 0 \leq t \leq t_f \tag{4.78}$$

$$x_1(0) = 3, \quad x_2(0) = 0 \tag{4.79}$$

$$x_1(t_f) = 0, \quad x_2(t_f) = 0 \tag{4.80}$$

$$t_f = \text{FEST} \tag{4.81}$$

Um die Rechnungen später zu vereinfachen, wird die Masse auf $m := 1$ und die Federsteifigkeit auf $k := 1$ in der Dynamik (4.78) festgelegt. Zunächst wird die Hamiltonfunktion (4.51) gebildet.

$$H = \frac{1}{2}u^2 + \boldsymbol{\lambda} \cdot \begin{pmatrix} x_2 \\ -\frac{k}{m}x_1 + \frac{1}{m}u \end{pmatrix} \tag{4.82}$$

$$= \frac{1}{2}u^2 + \lambda_1 \cdot x_2 + \lambda_2 \cdot (-\frac{k}{m}x_1 + \frac{1}{m}u) \tag{4.83}$$

$u(t)$ tritt nicht linear in H auf. Deswegen kann mit der Optimalitätsbedingung an die Steuerung (4.55) eine Gleichung für $u(t)$ hergeleitet werden.

$$\frac{dH}{du} = 0 \tag{4.84}$$

$$u + \lambda_2 \cdot \frac{1}{m} = 0 \tag{4.85}$$

$$u = -\frac{1}{m}\lambda_2 \tag{4.86}$$

Die kanonische Differentialgleichung (4.56) kann in dieser Problemstellung direkt ausgewertet werden ohne $u(t)$ einsetzen zu müssen.

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} \tag{4.87}$$

$$\dot{\lambda}_1 = \frac{k}{m}\lambda_2 \tag{4.88}$$

$$\dot{\lambda}_2 = \lambda_1 \tag{4.89}$$

(4.88)-(4.89) ist ein vollständig definiertes DGL-System 1. Ordnung. Mit der Vereinfachung $k = m = 1$ und einem analytischen Lösungsverfahren ergibt sich daraus:

$$\lambda_1 = c_1 \cos(t) - c_2 \sin(t) \tag{4.90}$$

$$\lambda_2 = c_2 \cos(t) - c_1 \sin(t) \tag{4.91}$$

Damit ist die Steuerung (4.86) definiert über:

$$u = -c_2 \cos(t) - c_1 \sin(t) \tag{4.92}$$

Die Steuerung liegt somit bis auf die zwei Integrationskonstanten c_1 und c_2 fest. Man kann nun die Steuerung in der Systemdynamik ersetzen.

$$\dot{x}_1 = x_2 \tag{4.93}$$

$$\dot{x}_2 = -x_1 + u = -x_1 - c_2 \cos(t) - c_1 \sin(t) \tag{4.94}$$

Dies ist wiederum ein DGL-System 1. Ordnung. Auch hier kann ein analytisches Lösungsverfahren angewendet werden.

$$x_1 = \frac{1}{2}\left((-tc_1 + c_2 + 2c_3)\cos(t) + (tc_2 + 2c_4)\sin(t)\right) \tag{4.95}$$

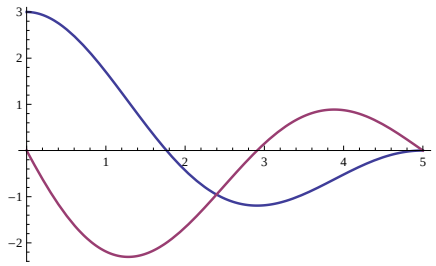
$$x_2 = \frac{1}{2}\left((-c_1 + tc_2 + c_4)\cos(t) + (tc_1 - 2c_3)\sin(t)\right) \tag{4.96}$$

Die vier verbliebenen Integrationskonstanten c_1, c_2, c_3 und c_4 können über die vier Randbedingungen (4.79) und (4.80) des Optimalsteuerungsproblems berechnet werden.

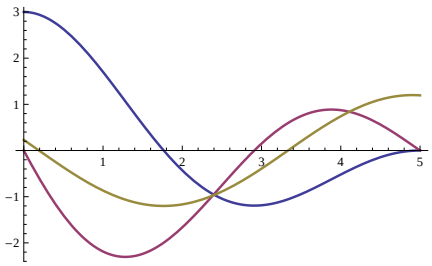
$$c_1 = \frac{6(20 + \sin(20))}{199 + \cos(20)} \approx 0.6292, \quad c_2 = \frac{12\sin(10)^2}{199 + \cos(20)} \approx -0.0178, \tag{4.97}$$

$$c_3 = \frac{600}{199 + \cos(20)} \approx 3.0089, \quad c_4 = \frac{3(20 + \sin(t))}{199 + \cos(20)} \approx 0.3146 \tag{4.98}$$

Mit einer festen Endzeit von $t_f = 5$ ergeben sich die Verläufe, die in Abbildung 4.7 zu sehen sind.



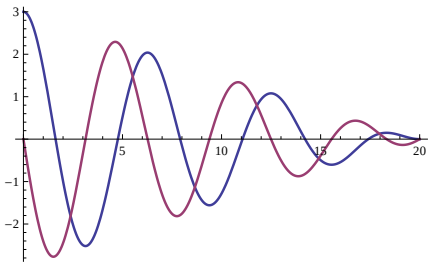
(a) blau: x_1 (Position); rot: x_2 (Geschwindigkeit)



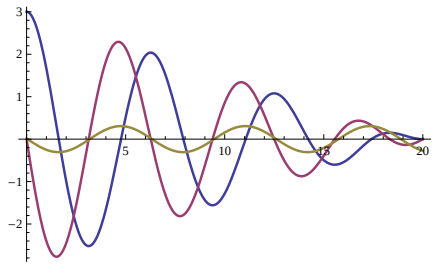
(b) gelb: u (Steuerung)

Abbildung 4.7: Die Lösungsverläufe für die energieminimal gedämpfte Punktmasse mit Endzeit $t_f = 5$

Mit einer festen Endzeit von $t_f = 20$ ergeben sich die Verläufe von Abbildung 4.8.



(a) blau: x_1 (Position); rot x_2 (Geschwindigkeit)



(b) gelb: u (Steuerung)

Abbildung 4.8: Die Lösungsverläufe für die energieminimal gedämpfte Punktmasse mit Endzeit $t_f = 20$

4.2.3 Lösungsmethoden für Optimalsteuerungsprobleme

Indirekte Verfahren: Einfach- und Mehrfachschießverfahren

Indirekte Verfahren benutzen explizit die Bedingungen aus der Optimalsteuerungstheorie und führen das Problem auf eine Randwertaufgabe zurück. Wie bereits für einfache Variationsprobleme gezeigt, führt dies zur Lösung der Euler-Lagrangeschen Differentialgleichung (3.74) mit Startwerten an t_0 und Endwerten an t_f. Werden zusätzliche Bedingungen an das Variationsproblem gestellt, erweitert sich das Randwertproblem, was in Abschnitt 4.2.1 für einige Standardfälle gezeigt wurde. Das Randwertproblem für Optimalsteuerungsprobleme besteht aus den kanonischen Differentialgleichungen (4.56)-(4.57) mit Startwerten an t_0 und Endwerten t_f und wird nach $\boldsymbol{\lambda}(t)$ und $\boldsymbol{x}(t)$ gelöst. Umfasst das Optimalsteuerungsproblem zusätzlich Innere-Punkt-Bedingungen (4.75), liegt nicht mehr ein Zweipunkt-Randwertproblem, sondern ein Mehrpunkt-Randwertproblem mit zusätzlichen Bedingungen an t_{s_i} vor. Dieses Randwertproblem enthält noch die Steuerung $\boldsymbol{u}(t)$. Sie kann aber mit einer der folgenden Bedingungen in Abhängigkeit von $\boldsymbol{\lambda}(t)$ und $\boldsymbol{x}(t)$ umgestellt werden und damit im Randwerproblem (4.56)-(4.57) ersetzt werden.

$$\frac{\partial H}{\partial \boldsymbol{u}} = 0 \text{ (Steuerungsbedingung (4.55))} \tag{4.99}$$

$$\boldsymbol{u}^*(t) = \min_{\boldsymbol{u}(t)} H(\boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t)) \text{ (Minimumsprinzip (4.61))} \tag{4.100}$$

$$\frac{d^k}{dt^k} s_i(\boldsymbol{x}(t), \boldsymbol{\lambda}(t)) = 0 \text{ (Bedingung an singuläre Steuerung (4.67))} \tag{4.101}$$

$$g_j^{(m_g)}(\boldsymbol{x}(t), \boldsymbol{u}(t)) = 0 \text{ (Bedingung bei aktiver Steuerbeschränkung [1])} \tag{4.102}$$

Ist das Randwertproblem für $\boldsymbol{\lambda}(t)$ und $\boldsymbol{x}(t)$ gelöst, können die Ergebnisse wiederum in $\boldsymbol{u}(t)$ eingesetzt werden, sodass auch die Steuerung des Problems vorliegt. Um diesen Ablauf durchzuführen, sind gute Kenntnisse der notwendigen Bedingungen nötig. Liegt das Randwertproblem in vollständig bestimmter Form vor, kann es mit Hilfe numerischer Verfahren gelöst werden.
Das Einfachschießverfahren [9], [63] löst allgemeine Randwertprobleme, indem es eine Startschätzung $s^{(0)}$ für die kompletten Startbedingungen vorgibt und damit nur das Anfangswertproblem mittels numerischer Integration (s. Abschnitt 3.4) bestimmt. Dies geschieht auf der kompletten Zeitspanne $t_0 \leq t \leq t_f$.

Dadurch ergibt sich ein Endzustand $\boldsymbol{F}(t_f)$. Der gewünschte Endwert ist aber B. Die Differenz der beiden ist eine Funktion in Abhängigkeit der Startwerte s.

$$c(s) = F(t_f) - B \tag{4.103}$$

Wird also ein Startwert s^* gefunden, sodass $F(t_f) = B$ gilt, ist das Randwertproblem gelöst. Somit muss lediglich die Nullstelle von Funktion (4.103) gefunden werden. Diese kann mit dem Newton-Verfahren bestimmt werden (s. Abschnitt in 4.1.1). Allerdings kann es aufgrund der hohen Nichtlinearität der DGL im Randwertproblem schwierig werden eine Lösung zu finden. Kleine Änderungen in s können große Änderungen am Randwert $F(t_f)$ bedeuten.
Mehrfachschießverfahren [9], [63] überwinden dieses Problem, indem sie nicht über die komplette Zeitspanne $t_0 \leq t \leq t_f$ integrieren, sondern den Zeitbereich in Intervalle einteilen. Dann wird für jeden Intervallanfang eine Startschätzung $s_i^{(0)}$ der Teilanfangswertprobleme vorgegeben, die wiederum numerisch integriert werden. Nun wird nicht mehr nur versucht die Differenz zwischen gewünschtem und tatsächlichem Endwert zu minimieren, sondern auch die Differenz zwischen dem End- und Anfangswert zweier benachbarter innerer Intervalle.

Direkte Verfahren: Schießverfahren und Kollokation

Direkte Verfahren basieren auf einer Diskretisierung der Steuerung (Schießverfahren) oder einer Diskretisierung der Steuerung und des Systemzustands (Kollokation). Die Herleitung der notwendigen Bedingungen (4.55)-(4.60) für das jeweilige Optimalsteuerungsproblem ist nicht nötig. Im Gegensatz zur Lösung des bei indirekten Verfahren entstehenden Randwertproblems wird das Optimalsteuerungsproblem in ein NLP Problem transformiert. Direkte Schießverfahren teilen die Zeit $[t_0, t_f]$ in n_s Intervalle auf.

$$t_i = \tau_i \cdot t_f, \quad 0 = \tau_1 < \tau_2 < \ldots < \tau_{n_s+1} = 1 \tag{4.104}$$

Es ist zu beachten, dass jedes Problem (auch mit freier Endzeit t_f) auf einen solchen normierten Zeitbereich von $[0,1]$ transformiert werden kann [3]. Die Steuerung kann nun auf diesem Gitter mittels Ansatzfunktionen approximiert werden. Mit linearen Ansätzen ergibt sich im Intervall $t_i \leq t < t_{i+1}$ [1]:

$$\tilde{\boldsymbol{u}}(t) = \boldsymbol{u}(t_i) + \frac{t - t_i}{t_{i+1} - t_i} \cdot \big(\boldsymbol{u}(t_{i+1}) - \boldsymbol{u}(t_i)\big) \tag{4.105}$$

Die Systemdynamiknebenbedingung (4.69) kann dann mit einer auf diese Weise diskretisierten Startschätzung $\tilde{\boldsymbol{u}}^{(0)}(t)$ und einem geeigneten Verfahren (s. Abschnitt 3.4) numerisch integriert werden, sodass die korrespondierenden Zustandswerte ebenfalls für die Gitterpunkte vorliegen. Je nach Anwendung eines Einfach- oder Mehrfachschießverfahrens geschieht die numerische Integration auf dem ganzen Intervall $[0,1]$ oder auf mehreren separaten Teilintervallen. Für die so ermittelten Werte für $\boldsymbol{u}$(t) und $\boldsymbol{x}$(t) wird die Zielfunktion (4.68) sowie die Verletzungen der Restriktionen ausgewertet. Diese Kosten und Fehlerwerte bilden das Optimierungskriterium und Nebenbedingungen eines NLP Problems, welches durch Anpassung der Parameter $\boldsymbol{u}_i$ auf dem Zeitgitter gelöst werden soll.
Direkte Kollokationsverfahren benutzen die vorgestellte Zeitgitterdiskretisierung (4.104). Allerdings wird nicht nur die Steuerung, sondern auch der Zustandsraum diskretisiert. Es ist dabei ratsam den Zustand mit einer höheren Ordnung als die Steuerung zu approximieren. Im einfachsten Fall geschieht dies mit einem abschnittsweisen konstanten Verlauf für die Steuerung und einem abschnittsweisen linearen Verlauf für den Zustand in den Intervallen $t_j \leq t < t_{j+1}$ [6].

$$\tilde{\boldsymbol{u}} = \boldsymbol{u}(t_{i+j/2}) \tag{4.106}$$

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}(t_j) + \frac{t - t_j}{t_{j+1} - t_j} \cdot \Big(\boldsymbol{x}(t_j) - \boldsymbol{x}(t_j)\Big) \tag{4.107}$$

Die diskreten Werte für den Zustand (auf den Zeitgitterpunkten) und für die Steuerung (in der Mitte der Zeitgitterpunkte) werden zusammen mit der Endzeit t_f in den Vektor $\boldsymbol{p}$ geschrieben.

$$\boldsymbol{p} = (\boldsymbol{x}(t_1), \boldsymbol{u}(t_{1+\frac{1}{2}}), \boldsymbol{x}(t_2), \boldsymbol{u}(t_{2+\frac{1}{2}}), \ldots, \boldsymbol{x}(t_{n_s}), \boldsymbol{u}(t_{n_s+\frac{1}{2}}), \boldsymbol{x}(t_{n_s+1}), t_f)^\top \tag{4.108}$$

Dieser Vektor dient als Parameter für das NLP Problem, das aus den Kosten und der Verletzung der Restriktionen für eine Startschätzung von $\tilde{\boldsymbol{u}}^{(0)}(t)$ und $\tilde{\boldsymbol{x}}^{(0)}(t)$ hervorgeht.

$$\min_{\boldsymbol{p}\in\mathbb{R}^{n_p}} \phi(\boldsymbol{p}) = \phi_{Mayer}(p_1, p_{n_p-1}, p_{n_p}) = \phi_{Mayer}(\boldsymbol{x}(t_1), \boldsymbol{x}(t_{n_s}), t_f) \tag{4.109}$$

$$\boldsymbol{f}\left(\tilde{\boldsymbol{x}}(t_{j+\frac{1}{2}}), \tilde{\boldsymbol{u}}(t_{j+\frac{1}{2}})\right) - \dot{\tilde{\boldsymbol{x}}}(t_{j+\frac{1}{2}}) = 0 \tag{4.110}$$

$$\tilde{x}_i(0) = x_{i,0} \tag{4.111}$$

$$\tilde{x}_j(t_f) = x_{j,t_f} \tag{4.112}$$

$$\tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}}(t_{j+\frac{1}{2}}), \tilde{\boldsymbol{u}}(t_{j+\frac{1}{2}})) \geq 0 \tag{4.113}$$

$$\tilde{\boldsymbol{g}}(\tilde{\boldsymbol{x}}(t_{j+\frac{1}{2}})) \geq 0 \tag{4.114}$$

Die Ableitung $\dot{\tilde{\boldsymbol{x}}}(t_{j+\frac{1}{2}})$ kann mit einer finiten Differenz $\dot{\tilde{\boldsymbol{x}}}(t_{j+\frac{1}{2}}) \approx \frac{\boldsymbol{x}(t_{j+1})-\boldsymbol{x}(t_j)}{t_{j+1}-t_j}$ angenähert werden. Die Werte für den Zustand in der Mitte eines Intervalls können über die Mittelung der beiden Werte am Rand $\tilde{\boldsymbol{x}}(t_{j+\frac{1}{2}}) \approx \frac{\boldsymbol{x}(t_{j+1})+\boldsymbol{x}(t_j)}{2}$ bestimmt werden. Die Zielfunktion des NLP Problems (4.109) hat die Form eines Mayer-Terms. Jedes Optimalsteuerungproblem mit Bolza-Funktional kann mittels einer Transformation auf ein Problem mit Zielfunktion in Mayer-Form gebracht werden [1]. Mit dieser vorangegangen Transformation des Optimalsteuerungsproblems werden im NLP Problem die kompletten Kosten korrekt abgedeckt. Es besitzt außerdem dünnbesetzte Strukturen, für die sich besonders die vorgestellten SQP 4.1.1 und Innere-Punkt-Methoden 4.1.1 eignen. Eine Konvergenzüberprüfung entscheidet, ob die geforderte Genauigkeit eingehalten wurde oder das Zeitgitter verfeinert werden muss um ausreichend genaue Resultate zu erhalten.
Pseudospektrale Kollokationsverfahren setzen spezielle Ansatzfunktionen ein um Steuerung und Zustand zu approximieren. Dabei werden die Funktionen nicht stückweise lokal auf einzelnen Teilintervallen zusammengesetzt, sondern komplett auf dem ganzen Intervall $t_0 \leq t \leq t_f$ definiert. Dies geschieht durch die Überlagerung gewichteter glatter Basisfunktionen, wie Legendre- (4.115) oder Tschebyschev-Polynomen (4.116).

$$L_N(t) = \frac{1}{2^N N!}\,\frac{d^N}{dt^N}(t^2-1)^N \tag{4.115}$$

$$C_N(t) = \cos\left(N\ \cos^{-1}(t)\right) \tag{4.116}$$

Die Gitterpunkte sind dabei die Nullstellen dieser Polynome oder die Nullstellen ihrer Ableitungen. Das Bolza-Funktional muss nicht erst auf Mayer-Term transformiert werden. Stattdessen wird es mittels numerischer Quadratur direkt ausgewertet.

5 Optimale Steuerung eines fräsenden Industrieroboters

In diesem Kapitel wird der Aufbau des Optimierungsansatzes beschrieben, welcher während dieser Arbeit entwickelt wurde und auf den Grundlagen basiert, die in Kapitel 3 und 4 erläutert wurden. Die verschiedenen Teilkomponenten werden vorgestellt und anschließend einzelne aus dem Verfahren resultierende Ergebnisse gezeigt.

5.1 Problemstellung

Die anfangs in dieser Arbeit beschriebenen Anforderungen an einen zum Fräsen eingesetzten Industrieroboter lassen einen Bereich an Möglichkeiten zu die Problemstellung zu definieren und zu lösen. In allen Fällen muss ein hinreichend genaues Modell des Gesamtvorganges entwickelt werden, das es sowohl ermöglicht detaillierte Aussagen zu treffen, die sich in der Realität verwerten lassen, als auch ein ausreichendes Maß an Abstraktion bietet, das dafür sorgt das Problem mit den zur Verfügung stehenden Werkzeugen effizient zu berechnen. Folgende Anforderungen bestehen an den Gesamtprozess, der mittels Optimaler Steuerung verbessert werden soll:

- Die durch das Fräsen entstehenden Pfadabweichungen des Roboterendeffektors bzw. des daran angebrachten Werkzeugs sollen minimiert werden (Optimierungsziel). Daraus resultiert ein präziser gefertigtes Bauteil.
- Der Manipulatorarm muss ausreichend genau modelliert werden, sodass sein Verhalten unter dem Einfluss äußerer Kräfte realistisch bestimmt werden kann (Dynamiknebenbedingung). Besonders das elastische Verhalten in den Gelenken, welches letztendlich zu einer Pfadabdrängung unter Last führt, muss genau abgebildet werden.
- Die Kräfte, die durch das Zerspanen des Werkstoffes auftreten, müssen in der Simulation möglichst exakt vorliegen um die Wechselwirkung zwischen Roboterarm und Fräsprozess ermitteln zu können (Dynamiknebenbedingung). Dazu ist es nötig einen Zusammenhang zwischen den Prozess- und Werkzeugparametern sowie den entstehenden Kräften zu finden.
- Die technischen Beschränkungen des Roboters müssen mit berücksichtigt werden (Zustandsbeschränkung, Steuerungsbeschränkungen).
- Um eine saubere Bearbeitung des Metalls zu gewährleisten, soll die Endeffektorgeschwindigkeit des Roboterarms während des Fräsvorgangs möglichst konstant sein (Zustandsbeschränkung).

5.2 Programmteile

Das Zusammenspiel der einzelnen verwendeten Komponenten wird in Abbildung 5.1 veranschaulicht. Zum Formulieren und Lösen des Optimalsteuerungsproblems wurde die Open-Source-Software PSOPT [7] eingesetzt. Sie bildet den zentralen Baustein des entwickelten Kompensationsansatzes. Hierbei handelt es sich um eine direkte Transkriptionsmethode, welche das dynamische Optimierungsproblem in ein NLP Problem übersetzt. Um dieses berechnen zu können, greift PSOPT auf einen NLP Solver zurück. Dazu kann der kommerzielle SQP Algorithmus (s. Abschnitt in 4.1.1) SNOPT [67] oder die frei erhältliche Implementierung der Inneren-Punkt-Methode (s. Abschnitt in 4.1.1) IPOPT [68] angebunden werden. Während dieser Arbeit wurde Letzteres durchgeführt. Wie in Kapitel 4 beschrieben, ist zur Lösung von Optimierungsproblemen (durch Verfahren mit Ordnung ≥ 1) die Berechnung von Ableitungen nötig. Da die Berechnung von Gradienten, Jacobi- oder Hessematrizen mit numerischen Approximationsmethoden sehr rechenintensiv ist, wird die Bibliothek ADOL-C (s. Abschnitt in 5.2.2) zur

automatischen Differentiation benutzt. Die Roboterdynamik ist für den realen 6-DoF Industrieroboter nicht mehr praktikabel per Hand aufstellbar. Deshalb wurde die am Fachgebiet „Simulation, Sytemoptimierung und Robotik“ entwickelte C++ Bibliothek namens MBSLIB verwendet um das Modell als Programmcode aufzubauen.

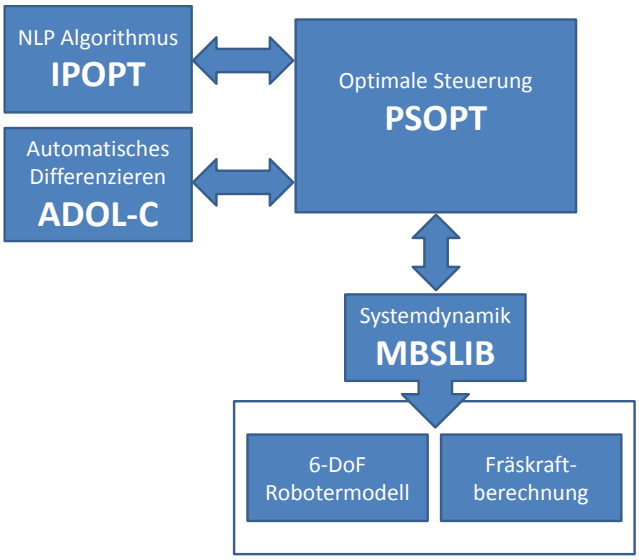


Abbildung 5.1: Aufbau der Kompensationsstrategie: Die Doppelpfeile repräsentieren Kopplungen zwischen PSOPT und Programmkomponenten, welche Werte und Ergebnisse an den Optimalsteuerungsalgorithmus liefern. Die Systemdynamik ist in Robotermodell und Fräskraftberechnung unterteilt.

5.2.1 Optimalsteuerung mit PSOPT

PSOPT bietet ein in C++ geschriebenes Framework zur Erstellung und Lösung von Optimalsteuerungsproblemen. Es verwendet einen effizienten direkten pseudospektralen Kollokationsalgorithmus (s. Abschnitt in 4.2.3). Da es in derselben Programmiersprache wie die Dynamikbibliothek MBSLIB geschrieben ist und bereits automatisches Differenzieren unterstüzt, eignet es sich sehr gut um die vorliegende Problemstellung zu bearbeiten. Mit PSOPT werden Optimalsteuerungsprobleme innerhalb einer Hauptdatei spezifiziert, die anschließend über ein Makefile kompiliert wird. Der Aufruf des dabei entstehenden Programms startet das Lösungsverfahren, bei dem iterativ die einzelnen NLP Probleme gelöst werden, bis das Verfahren erfolgreich konvergiert, keine optimale Lösung gefunden werden kann, oder aufgrund einer Abbruchbedingung vorzeitig beendet wird. Nach Terminierung des Programms, werden Informationen über den Optimierungslauf ausgegeben und drei Dateien erstellt, die bei Erfolg die optimale Lösung oder bei einem Fehlschlag die bisher beste Lösung enthalten. Die Datei `t.dat` beinhaltet das diskrete Zeitgitter, auf welchem die Lösungspunkte berechnet wurden, `x.dat` enthält die zu diesen Zeitpunkten passenden Zustandsinformationen des Systems und `u.dat` die korrespondierenden Steuerungsinformationen. Das Optimalsteuerungsproblem wird in mehreren Unterfunktionen definiert, die nun kurz den Bedingungen aus (4.68)-(4.75) gegenübergestellt werden. PSOPT erlaubt darüber hinaus noch allgemeinere Formulierungen von Optimalsteuerungsproblemen, auf die hier aber nicht näher eingegangen wird. Für eine ausführliche Beschreibung der Funktionalität von PSOPT sei auf [7] verwiesen.

◦ `endpoint_cost`: Hier wird der Mayer-Term $\phi(\boldsymbol{x}(t_f), t_f)$ der Zielfunktion (4.68) definiert.

◦ `integrand_cost`: Diese Methode enthält das Integral über den Lagrange-Term $\int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt$ der Zielfunktion (4.68).

◦ `dae`: Die rechte Seite $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ der Dynamiknebenbedingung (4.69) wird in dieser Methode implementiert.

◦ `events`: Hier werden die Zuweisungen der jeweiligen Randbedingung für Startwerte (4.70), Endwerte (4.71) und inneren Grenzen bei mehrphasigen Problemen festgelegt.

◦ `linkages`: Die Inneren-Punkt-Bedingungen (4.75) bei Mehrphasenproblemen werden in diesem Abschnitt aufgestellt.

◦ `main`: Hier werden die Anfangs- und Endwerte für die jeweiligen Phasen definiert und den `events` zugewiesen. Diese können als fest oder aber auch als natürliche Randbedingungen definiert werden (vgl. Abschnitt in 4.2.1). Zustandsbeschränkungen (4.74) und Steuerbeschränkungen (4.73) sowie Schranken an die Start- und Endzeit (4.72) können festgelegt werden. Außerdem müssen Einstellungen am Lösungsalgorithmus vorgenommen werden. Dabei kann die Anzahl der (initialen) Gitterpunkte, die Kollokationsmethode sowie das Verfahren zur Berechnung von Ableitungen gewählt werden. Das Vorgeben einer Startlösung ist ebenso möglich.

5.2.2 Automatisches Differenzieren mit ADOL-C

Automatisches Differenzieren ermöglicht es Ableitungen von Funktionen zu bilden, die als Programmcode vorliegen. Das zu Grunde liegende Prinzip beruht auf der Tatsache, dass jede komplexe Funktion sich aus einer Sequenz von elementaren Rechenoperationen zusammensetzt. Die analytischen Ableitungsregeln für diese unären (sin, cos, exp, ln) und binären ($+$, $-$, $\cdot$, $\div$) Operationen sind bekannt und können sequenziell angewandt werden. Der Programmcode muss dazu analysiert werden um die richtige Reihenfolge der Ableitungen bilden zu können. Die Software ADOL-C [5] benutzt dabei das Konzept des Überladens von Klassen, das aus dem objektorientierten Programmieren stammt. Es wird ein neuer Typ `adouble` zur Verfügung gestellt, der an die Stelle des gewöhnlichen C++ `double`-Typs tritt. Mit diesem speziellen Datentyp wird es der Software intern ermöglicht, die zur Laufzeit auf den `adouble`-Variablen ausgeführten Operationen mitzuverfolgen und damit Rückschlüsse auf benötigte Ableitungen ziehen zu können. Das Verfolgen der Rechenoperationen geschieht innerhalb eines Trace-Abschnitts im Programmcode. Es wird dabei eine Tape-Datei erzeugt, die einen Sequenzgraphen der im Trace-Abschnitts implementierten Funktion bezüglich vorher definierte unabhängiger Variablen enthält. Mit diesen Daten ist es möglich die Ableitungen zu erstellen. Das Verwendung wird im folgenden Codeabschnitt demonstriert.

Listing 5.1: Automatisches Differenzieren mit ADOL-C

```
1 double t, f
2 trace_on(1);
3         adouble at, ax, af
4         at << = t;
5         ax = 1.234;
6         af = ax*sin(at)*cos(at)+3*at-ax;    // Funktion f(t) -> (d/dt) f(t)
```

```
7          af >>= f;
8  trace_off;
```

Nach Ausführung liegt in der Tape-Datei mit der Kennung „1“ die Repräsentation der Ableitung $\frac{d}{dt}f$ der Funktion f vor.

5.2.3 Robotermodellierung und Dynamik mit MBSLIB

Die Dynamik von Robotern kann, wie in Kapitel 3 gezeigt, hergeleitet werden. Es stehen z.B. die Newton-Euler Rekursion 3.2.1 oder der Lagrange-Formalismus 3.3.2 zur Verfügung. Für reale Roboter mit sechs Freiheitsgraden ist die Erstellung der Bewegungsgleichungen per Hand nicht mehr praktikabel, da die entstehenden Gleichungen zu umfangreich und komplex werden. Die in C++ geschriebene MultiBodySystemLIBrary (MBSLIB) bietet eine Möglichkeit ein Robotermodell strukturiert und effizient aufzubauen. Außerdem verfügt sie über eine Sammlung an nützlichen Algorithmen für in der Robotik häufig auftretende Problemstellungen. Auch Erweiterungen des starren Mehrkörpermodells sind damit realisierbar. Die prinzipielle Verwendung der MBSLIB wird anhand eines Manipulators mit zwei Freiheitsgraden skizziert.

Listing 5.2: MBSLIB: Dynamikmodell für einen 2-DoF Manipulator

```
1  mbslib::MbsCompoundWithBuilder * createRobot(TScalar l1, TScalar m1,
2                                                TScalar l2, TScalar m2)
3  {
4    TMatrix3x3 ISP;
5    ISP << 1, 0, 0,
6           0, 1, 0,
7           0, 0, 1;
8
9    mbslib::MbsCompoundWithBuilder * rob =
10           new mbslib::MbsCompoundWithBuilder();
11
12   rob->addFixedBase("Base");
13   rob->addRevoluteJoint(Eigen::Vector3d::UnitZ().cast<TScalar>(),0, "J1");
14   rob->addRigidLink(TVector3(l1,0,0), TVector3(0,0,0), m1, ISP, "L1");
15   rob->addRevoluteJoint(Eigen::Vector3d::UnitZ().cast<TScalar>(),0, "J2");
16   rob->addRigidLink(TVector3(l2,0,0), TVector3(0,0,0), m2, ISP, "L2");
17   rob->addEndpoint("TCP");
18   rob->setGravitation(TVector3(0,-9.81,0));
19   return rob;
20 }
```

Zunächst wird ein leeres `MbsCompoundWithBuilder`-Objekt erstellt, dem nach und nach die einzelnen Roboterelemente hinzugefügt werden. Dann wird eine feste und unbewegliche Basis erstellt. Entsprechend der DH-Konventionen wird entlang der kinematischen Kette jeweils ein Drehgelenk und anschließend ein Glied hinzugefügt. Die Längen und Massen der Roboterglieder können über die Parameter `l1`, `l2` und `m1`, `m2` festgelegt werden. Der Trägheitstensor `ISP` der Glieder ist hier auf die Einheitsmatrix $I^{3\times3}$ gesetzt worden. Am Ende wird der TCP des Roboters definiert, der das Ende der Kinematikkette festlegt. Außerdem wird die Gravitationsbeschleunigung festgesetzt.
Die Auswertung der Dynamik für den 2-DoF Manipulator wird im nächsten Codeabschnitt dargestellt.

Listing 5.3: MBSLIB: Dynamikauswertung für einen 2-DoF Manipulator

```
1  mbslib::MbsCompound * robot = createRobot(l1, m1, l2, m2);
2
3  TVectorX q(robot->getNumberOfJoints());
4  q(0) = q1;
5  q(1) = q2;
6  robot->setJointPosition(q);
7
8  TVectorX dq(robot->getNumberOfJoints());
9  dq(0) = dq1;
10 dq(1) = dq2;
11 robot->setJointVelocity(dq);
12
13 TVectorX ddq(robot->getNumberOfJoints());
14 ddq.setZero();
15 robot->setJointAcceleration(ddq);
16
17 TVectorX tau(robot->getNumberOfJoints());
18 tau(0) = motor_control1;
19 tau(1) = motor_control2;
20 robot->setJointForceTorque(tau);
21
22 robot->doCrba();
23
24 ddq = robot->getJointAcceleration();
25 dq = robot->getJointVelocity();
26 q = robot->getJointPosition();
```

Zunächst wird eine Instanz `roboter` des definierten 2-DoF Manipulatorarms erstellt. Gemäß der Anzahl seiner Freiheitsgrade wird die aktuelle Konfiguration festgelegt. Es werden die aktuellen Gelenkwinkel `q`, die Gelenkwinkelgeschwindigkeiten `dq` sowie die Gelenkwinkelbeschleunigungen `ddq` festgesetzt. Dann wird das Antriebsmoment `tau` bestimmt und die Dynamik mittels des in der Methode `doCrba()` implementierten Composite-rigid-body-algorithm [54] (oder `doAba()` Articulated-body-algorithm [54]) ausgewertet. Es ergibt sich die neue Roboterkonfiguration mit aktualisierten Werten für `q`, `ddq` und `ddq`. Ähnlich zu diesem Beispiel wurde für das 6-DoF Modell des Industrieroboters verfahren, welches für diese Arbeit zur Verfügung gestellt wurde. Jedoch ist dieses Modell nicht nur aufgrund der höheren Anzahl an Freiheitsgraden komplexer. Es ist generisch aufgebaut und lässt sich sehr einfach über eine Tabelle mit DH-Werten modifizieren. Außerdem können Elastizitäten und zusätzliche Kippachsen sowie Getriebespiel mitberücksichtigt werden. Damit besitzt die vollständige Roboterdynamik diese Form:

$$M(\tilde{\boldsymbol{q}}(t))\,\ddot{\tilde{\boldsymbol{q}}}(t) + C(\tilde{\boldsymbol{q}}(t), \dot{\tilde{\boldsymbol{q}}}(t)) + G(\tilde{\boldsymbol{q}}(t)) = \boldsymbol{\tau} + J_c \cdot \boldsymbol{F}_{xyz,tool}(t) \tag{5.1}$$

Die Matrix M und die Vektoren C, G wurden bereits in Abschnitt 3.2.1 vorgestellt (vgl. (3.52), (3.53) und (3.54)) und stammen aus dem starren Dynamikmodell. Das Antriebsmoment $\boldsymbol{\tau}$ kann das gelenkelastische Verhalten abbilden.

$$\boldsymbol{\tau} = K \cdot (\tilde{\boldsymbol{q}}_u(t) - \tilde{\boldsymbol{q}}_{soll}(t)) \tag{5.2}$$

Sollen zusätzlich noch Dämpfungen modelliert werden, kommt in (5.2) ein weiterer Term $D \cdot (\dot{\tilde{\boldsymbol{q}}}_u(t) - \dot{\tilde{\boldsymbol{q}}}_{soll}(t))$ hinzu. Die erweiterten Gelenkwinkel $\tilde{\boldsymbol{q}}$ können neben den sechs realen durch Motoren angetriebenen Gelenken des Roboters außerdem virtuelle Freiheitsgrade enthalten. Diese gehen aus den

Elastizitäten in x- und y-Richtung der Gelenke hervor. Sollen diese Kippfreiheitsgrade im Modell mitberücksichtigt werden, besteht $\tilde{\boldsymbol{q}}$ aus den sechs Gelenkwinkeln q_i und den Kippwinkeln q_i^x und q_i^y des jeweiligen Gelenks. Die Steifigkeitsmatrix K benutzt die erweiterten Gelenkwinkel $\tilde{\boldsymbol{q}}$ um nicht nur das elastische Verhalten um die Drehachsen in z-Richtung widerzuspiegeln, sondern auch die Elastizität der identifizierten Kippachsen in x- und y-Richtung. Da diese passiv und nicht steuerbar sind, ist der erweiterte Steuerungsvektor $\tilde{\boldsymbol{q}}_u$ an diesen Stellen $= 0$. Experimente am realen Roboter haben gezeigt, dass vor allem in den Gelenken 1, 2, 3 und 6 virtuelle Kippachsen sowohl in x- als auch in y-Richtung benötigt werden. Eine dementsprechende Steifigkeitsmatrix K lag im Robotermodell bereits vor. $\tilde{\boldsymbol{q}}_{soll}$ enthält den gewünschten Fräspfad in Gelenkkoordinaten. Darin sind die Einträge für die virtuellen Achsen $= 0$. Der zweite Summand auf der rechten Seite von Gleichung (5.1) beschreibt die externe Kraft in kartesischen Koordinaten, die am Roboter angeschlossenen Werkzeug wirkt. Über die Jacobimatrix J_c des Manipulators [55] wird sie auf Momente in den Gelenken umgerechnet. Eine detaillierte Berechnung dieser Fräskräfte kann über eine in MATLAB entwickelte Simulationssoftware bestimmt werden. Wie in Abschnitt 2.3 vorgestellt, können dazu verschiedene Methoden zum Einsatz kommen. Das implementierte Simulationsverfahren basiert auf einem numerischen Nagelbrettmodell. Eine direkte Kopplung der Fräskraftberechnung an das Dynamikmodell und damit auch an den Optimierungsalgorithmus ist jedoch zu rechenaufwendig. Durchgeführte Frässimulationen mit einem ausgewählten Satz an Prozessparametern haben jedoch gezeigt, dass sich die Kräfte in erster Näherung gut durch eine geschwindigkeitsabhängige Funktion darstellen lassen. Da der verwendete Werkstoff homogen und isotrop ist, ist nur der Betrag der TCP-Geschwindigkeit $||\boldsymbol{v}_{tool}(t)||$ ausschlaggebend.

$$\boldsymbol{F}_{xyz,tool}(t) = \boldsymbol{F}_{xyz,tool}\left(||\boldsymbol{v}_{tool}(t)||\right) \tag{5.3}$$

Der Verlauf von $\boldsymbol{F}_{xyz,tool}\left(||\boldsymbol{v}_{tool}(t)||\right)$, der über mehrere Simulationen mit unterschiedlichen Geschwindigkeiten ermittelt wurde, ist in Abbildung 5.2 aufgetragen.

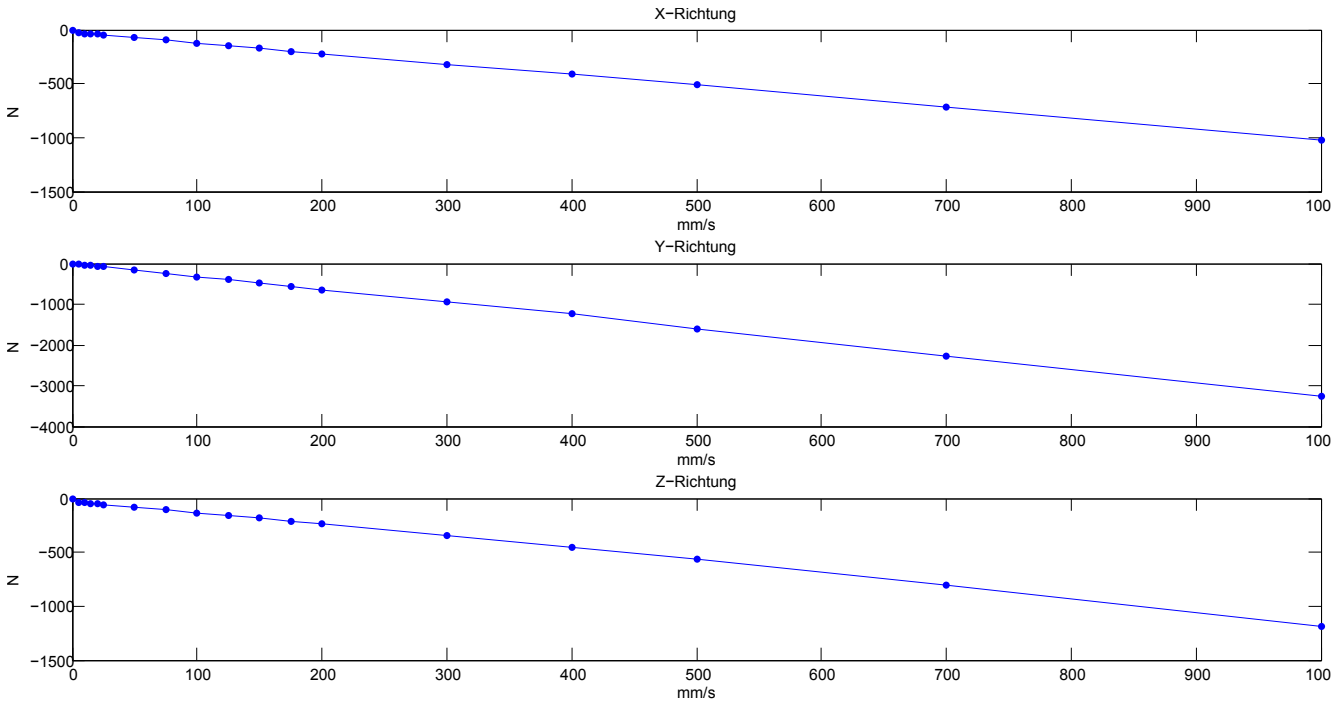


Abbildung 5.2: Fräskraftverlauf in Abhängigkeit von $||\boldsymbol{v}_{tool}||$

Die Fräskraftberechnung wird während der Optimierung mit diesen Funktionen bestimmt. Da die gezeigten Funktionen nur aus Messpunkten aufgebaut sind, muss während der Optimierung linear interpoliert werden, wenn Werte zwischen den Punkten benötigt werden. Ebenso kann für jede Richtungskomponente von $F_{e_i,tool}$ mittels Regression, der Steigungsparameter m und y-Achsenabschnitt b für eine Gradenfunktion $F_{\boldsymbol{e}_i}\left(||v_{tool}||\right) = m \cdot v + b$ berechnet werden. Wie in Abbildung 5.2 zu sehen ist, sind die Verläufe nahezu linear, was diese Alternative genauso rechtfertigt wie eine lineare Interpolation. Die MBSLIB stellt Methoden zur Verfügung um externe Kräfte auf die Roboterstruktur zu übertragen.

Listing 5.4: MBSLIB: Koppelung der Fräskraft an den Rooter

```
1  TMatrix3x3 R = robot->getEnd().getCoordinateFrame().R;
2  TVector3 v = R * robot->getEnd().getCoordinateFrame().v;
3
4  double absvel = sqrt(pow(v(0),2) + pow(v(1),2) + pow(v(2),2));
5
6  TVector3 millingforce;
7  millingforce(0) = millingforceX(absvel);
8  millingforce(1) = millingforceY(absvel);
9  millingforce(2) = millingforceZ(absvel);
10
11 TVector3 millingtorque;
12 millingtorque.setZero();
13
14 robot->getEnd().addExternalForceTorqueWCS(millingforce, millingtorque);
```

Die Funktionen `millingforceX`, `millingforceY` und `millingforceZ` enthalten die gezeigte Approximation der Fräskraftberechnung. Auf die Modellierung zusätzlicher Momente, die beim Fräsprozess entstehen, wurde verzichtet. Sie können aber einfach durch Anpassung der Werte in `millingtorque` mitberücksichtigt werden.

5.3 Formulierung des Optimalsteuerungsproblems: Basisvariante

5.3.1 Pfadvorgabe

Zu Beginn der Formulierung muss entschieden werden, auf welche Weise der Sollpfad vorgegeben wird. Während der Optimierung muss es zu jedem Zeitpunkt t möglich sein die Abweichung zwischen Ist- und Sollposition zu ermitteln. Der Sollpfad muss also als eine geschlossene Funktion vorliegen. Da z.B. Polynomfunktionen nur eingeschränkte Geometrien des Sollpfads erlauben, ist es notwendig einen allgemeinen Ansatz zu verwenden. Deshalb wurde in dieser Arbeit der Sollpfad dem Optimierungsalgorithmus mittels diskreter Punkte übergeben, die aus beliebigen Kurvenbeschreibungen generierbar sind. Um dennoch eine kontinuierliche Beschreibung des Pfads zu erhalten, kann während der Optimierung ein Interpolationsverfahren angewandt werden, sodass auch Werte vorhanden sind, falls t zwischen zwei Stützstellen liegt. Es kann z.B. eine Splineinterpolation gewählt werden, die PSOPT über eine Hilfsfunktion `spline_interpolation` unterstützt. Da der Sollpfad stets in sehr hoher Auflösung mit vielen Stützstellen übergeben wurde, kann man aus Gründen der Berechnungseffizienz auf eine lineare Interpolation zurückgreifen. Die Vorgabe des Sollpfads in kartesischen Koordinaten (bzgl. der Roboterbasis) bringt einige Probleme mit sich. Wie in Abschnitt 3.1.1 gezeigt, treten beim inversen Kinematikproblem Mehrdeutigkeiten auf. Für viele TCP-Punkte des Sollpfads sind mehrere Gelenkwinkelstellungen möglich,

wodurch der Suchraum sehr groß werden kann. Bereits Tests an einem 2-DoF Roboter haben gezeigt, dass der Optimierungsalgorithmus häufig Lösungen findet, die ein mehrfaches Wechseln zwischen äquivalenten Gelenkwinkelkonfigurationen beinhalten. Werden die Motormomente stark beschränkt, ist in vielen Fällen dieser abrupte Wechsel zwischen zwei Konfigurationen nicht mehr möglich. Dann ist es für den Optimierungsalgorithmus schwierig zu einer gültigen Lösung zu konvergieren und oftmals kann kein Optimum mehr gefunden werden. Dieses Problem lässt sich zwar durch zusätzliche Energieterme in der Zielfunktion, die einen sprunghaften Wechsel in den Gelenken bestrafen, vermindern, aber es kann dadurch nicht generell vermieden werden. Deshalb wurde in einem ersten Formulierungsansatz des Optimalsteuerungsproblems der Sollpfad in Gelenkwinkeln vorgegeben. Der kartesische Sollpfad wird dazu zunächst diskretisiert und dann mit einem numerischen Algorithmus zur Berechnung des inversen Kinematikproblems (s. Gleichungen (3.8)-(3.10)) in Gelenkwinkelkoordinaten umgewandelt. In dieser Arbeit wurde zu diesem Zweck auf die für MATLAB frei erhältliche Robotics Toolbox zurückgegriffen [69].

Listing 5.5: Inverse Kinematikberechnung zur Sollpfadvorgabe

```
% Aufbau des Robotermodells "kuka" mit DH-Werten
L1 = link([-pi/2  0.35  0.00  0.75  0.00], 'standard');
L2 = link([  0    1.25  0.00  0.00  0.00], 'standard');
L2.offset =-pi/2;
L3 = link([ pi/2  0.00  0.00  0.00  0.00], 'standard');
L4 = link([-pi/2  0.00  0.00 -1.10  0.00], 'standard');
L5 = link([ pi/2  0.00  0.00  0.00  0.00], 'standard');
L6 = link([ pi    0.00  0.00 -0.23  0.00], 'standard');
kuka = robot({L1 L2 L3 L4 L5 L6}, 'KR-210');

% Diskretisierung des Sollpfads an den Gitterpunkten "disgrid"
for t=1:size(disgrid,2)
    [rx,ry,rz] = tcppath(disgrid(t),tfinal);
    TCP(1:4,1:4,t) = [1,0,0,rx; 0,1,0,ry; 0,0,1,rz; 0,0,0,1];
end

% Inverse Kinematik für alle Posen in "TCP"
Qini = [0 -pi/2 pi/4 0 0 0];
Q = ikine(kuka, TCP, Qini);
```

Es muss anschließend überprüft werden, ob die Gelenkwinkelverläufe in `Q` Sprünge enthalten. Ist dies der Fall, muss versucht werden diese durch eine Anpassung der Startwerte `Qini` zu vermeiden. Außerdem sollte der Wertebereich der Gelenkwinkelverläufe auf ein passendes Intervall normiert werden, damit nicht unötig ein Vielfaches von π verwendet wird. Die so aufbereiteten Gelenkwinkelverläufe können dem Optimierungsalgorithmus übergeben werden.

5.3.2 Endeffektorgeschwindigkeit

Die Endeffektorgeschwindigkeit des Roboters soll konstant während des Fräsvorgangs sein, damit gute Resultate erzielt werden können. Solche Restriktionen sind grundsätzlich über Zustandsnebenbedingungen implementierbar. Eine passende Formulierung lautet:

$$g_1(\boldsymbol{q}(t)) = 0 \tag{5.4}$$

$$||\boldsymbol{v}_{tool}(t)|| - v_{soll} = 0 \tag{5.5}$$

Dies stellt eine für den Optimierungsalgorithmus sehr schwer zu erfüllende Restriktion dar. Der aktuelle Zustand hängt von den Gelenkwinkeln $\boldsymbol{q}(t)$ ab. Die Bedingung selbst ist aber in kartesischen Koordinaten definiert. Das bedeutet, dass zur Einhaltung dieser Bedingung das komplette Roboterdynamikmodell berücksichtigt werden muss. Zusätzlich stellt auch die nichtlineare Betragsfunktion eine große Herausforderung dar. Um eine robuste und effiziente Optimierung zu gewährleisten, muss also ein anderer Weg gefunden werden. In dieser Arbeit wurde eine spezielle Parametrisierung des Sollpfads vorgenommen, sodass die Ableitung nach der Zeit auf eine konstante Bahngeschwindigkeit führt. Mit diesen speziell aufbereiteten Rohdaten wird das Hinzufügen von Nebenbedingungen in das Optimalsteuerungsproblem vermieden, was Vorteile für die Berechnungseffizienz hat. Für eine parametrische Kurve $\boldsymbol{f}(t) \in \mathbb{R}^3$ mit dem Parameter $t \in [0, \overline{t_f}]$ ist die Geschwindigkeit entlang der Kurve durch die Ableitung nach t bestimmt.

$$\dot{\boldsymbol{f}}(t) = \begin{pmatrix} \frac{df_1}{dt} \\ \frac{df_2}{dt} \\ \frac{df_2}{dt} \end{pmatrix} \tag{5.6}$$

Der Betrag der Geschwindigkeit ist dann gegeben durch:

$$||\dot{\boldsymbol{f}}(t)|| = \sqrt{\left(\frac{df_1}{dt}\right)^2 + \left(\frac{df_2}{dt}\right)^2 + \left(\frac{df_3}{dt}\right)^2} \tag{5.7}$$

Ist einer der Einträge in $\boldsymbol{f}(t)$ nichtlinear, dann ist im Allgemeinen der Betrag der Geschwindigkeit nicht konstant [1]. Um dies zu vermeiden, kann eine Paremetrisierung mit der Bogenlänge durchgeführt werden. Wird der Parameter t (Zeit) äquidistant diskretisiert, so ist für nichtlineare Funktionen der Abstand in Bogenlänge zwischen jeweils zwei benachbarten Werten t_i und t_{i+1} nicht konstant. D.h., in gleichen Zeitspannen müssen unterschiedlich lange Wege zurückgelegt werden, was eine Änderung der Geschwindigkeit zur Folge hat. Wird ein neuer Parameter s eingeführt, der die Kurve in Abhängigkeit ihrer Bogenlänge beschreibt, können die Unterschiede in den Abständen vermieden werden. Die Bogenlänge eines Abschnitts der Kurve $\boldsymbol{f}(t)$ von 0 bis t ist gegeben durch:

$$s(t) = \int_0^t \sqrt{\left(\frac{df_1}{d\tau}\right)^2 + \left(\frac{df_2}{d\tau}\right)^2 + \left(\frac{df_3}{d\tau}\right)^2}\, d\tau \tag{5.8}$$

Kann das Integral in Gleichung (5.8) analytisch bestimmt werden, kann sie nach t umgeformt werden. Es liegt dann eine Beschreibung $t(s)$ vor. Mit dieser kann der Parameter t in der Funktion $\boldsymbol{f}(t)$ ersetzt werden. Damit entsteht eine äquivalente Funktion $\tilde{\boldsymbol{f}}(s)$ mit dem Parameter $s \in [0, L]$, wobei L die Gesamtbogenlänge der Kurve ist, wenn t seinen maximalen Wert $\overline{t_f}$ annimmt. Der Betrag der Geschwindigkeit für eine Kurve mit solcher Parametrisierung ist immer $= 1$, denn:

$$\begin{aligned} ||\frac{d}{ds}\tilde{\boldsymbol{f}}(s)|| &= ||\tilde{\boldsymbol{f}}'(t(s))|| \cdot \frac{dt}{ds} && (5.9)\\ &= ||\tilde{\boldsymbol{f}}'(t(s))|| \cdot \frac{1}{\frac{ds}{dt}} && (5.10)\\ &= ||\tilde{\boldsymbol{f}}'(t(s))|| \cdot \frac{1}{||\tilde{\boldsymbol{f}}'(t(s))||} && (5.11)\\ &= 1 && (5.12) \end{aligned}$$

[1] Die Kreisbahn $\boldsymbol{f}(t) = (\cos(t), \sin(t), 0)^\top$ ist beispielsweise eine Ausnahme.

Das Integral von Gleichung (5.8) kann für komplexe Kurven nicht mehr analytisch bestimmt werden. Aufgrund der Erläuterungen im vorangegangenen Unterabschnitt 5.3.1 muss der Pfad letztendlich nur diskretisiert vorliegen. Es kann vorher festgelegt werden, aus wie vielen Punkten die Diskretisierung bestehen soll. Daraus ergibt sich der Abstand zwischen den einzelnen Stützstellen, falls die Gesamtbogenlänge L der Kurve $\boldsymbol{f}(t)$ bekannt ist. Sie kann durch numerischer Integration des Integrals (5.13) berechnet werden.

$$L = \int_0^{\overline{t_f}} \sqrt{\left(\frac{df_1}{d\tau}\right)^2 + \left(\frac{df_2}{d\tau}\right)^2 + \left(\frac{df_3}{d\tau}\right)^2}\, d\tau \tag{5.13}$$

In MATLAB steht dazu der Befehl `quadgk` zur Verfügung. Die Intervallbogenlänge Δs zwischen den Diskretisierungspunkten ergibt sich aus dem Quotienten von L und der Anzahl der gewünschten Stützstellen. Für jeweils zwei benachbarte Stützstellen s_i und s_{i+1} muss dann gelten:

$$\Delta s = \int_{s_i}^{s_{i+1}} \sqrt{\left(\frac{df_1}{d\tau}\right)^2 + \left(\frac{df_2}{d\tau}\right)^2 + \left(\frac{df_3}{d\tau}\right)^2}\, d\tau \tag{5.14}$$

$$\Delta s - \int_{s_i}^{s_{i+1}} \sqrt{\left(\frac{df_1}{d\tau}\right)^2 + \left(\frac{df_2}{d\tau}\right)^2 + \left(\frac{df_3}{d\tau}\right)^2}\, d\tau = 0 \tag{5.15}$$

Auch in Gleichung (5.15) kann das Integral numerisch gelöst werden. Die linke Seite entspricht einer Funktion $n(s_{i+1})$. Zur Generierung der Diskretisierung wird eine Iteration durchgeführt. Man startet am Anfang der Kurve mit der Stützstelle $s_0 = 0$. Der Wert des nächsten Punktes s_{i+1}, der den Abstand Δs entlang der Kurve zum vorherigen Punkt s_i haben soll, ist noch unbekannt. Allerdings muss Gleichung (5.15) erfüllt sein, bzw. $n(s_{i+1})$ besitzt dort eine Nullstelle. Zur Nullstellensuche kann z.B. das Newtonverfahren eingesetzt werden (s. Abschnitt in 4.1.1). MATLAB verfügt mit `fzero` über ein hilfreiches Werkzeug zur numerischen Nullstellenbestimmung. Damit ist der Wert von s_{i+1} mit ausreichender Genauigkeit bekannt. Er wird nun zur neuen unteren Grenze für das Integral in (5.15) und die Suche der oberen Integralgrenze mit Hilfe der Nullstellenbestimmung beginnt von vorne. Diese Iteration wird so lange durchgeführt, bis die gewünschte Anzahl an Stützstellen erreicht wurde und $s_n \approx \overline{t_f}$ gilt. Das Diskretisierungsgitter `disgrid`, das in den Zeilen 12 und 13 von Programmausschnitt 5.5 benutzt wurde, lässt sich auf diese Weise erstellen. Ein anderes Verfahren zur Erstellung einer solchen Parametrisierung ist in [70] zu finden.

5.3.3 Zielfunktion

Die Zielfunktion soll in diesem Szenario die Abweichung zwischen der Ist- und Sollposition der Werkzeugspitze bzw. des Roboterendeffektors widerspiegeln. Da der Sollpfad in Gelenkwinkeln vorgegeben wird, kann die Abweichung auch in Gelenkwinkeln betrachtet werden. Da nur der Betrag der Abweichung wichtig ist und nicht das Vorzeichen, wird der quadratische Fehler zwischen Ist- und Sollwert benutzt. Zusätzliche Kosten zur Endzeit entstehen nicht, weswegen der Mayer-Term auf 0 gesetzt wird. Außerdem können Gewichtungsfaktoren ω_i eingeführt werden. Abweichungen im ersten Gelenk haben aufgrund des längeren Hebelarms einen größeren Einfluss auf die Endeffektorgenauigkeit als Abweichungen in den hinteren Gelenken und sollten deshalb höher gewichtet werden.

$$J[\boldsymbol{u}(t), \boldsymbol{q}(t)] = 0 + \int_0^{t_f} \sum_{i=1}^{6} \omega_i \cdot \left(q_{\text{soll},i}(t) - q_i(t)\right)^2 \tag{5.16}$$

Wird die Zielfunktion jedoch direkt über kartesische Abweichungen ausgedrückt, entfallen diese Faktoren. Die aktuelle Istposition des Endeffektors $\boldsymbol{p}$ hängt von den Gelenkwinkeln $\boldsymbol{q}(t)$ ab und muss zusätzlich vom Dynamikmodell zur Verfügung gestellt werden. Das Zielkriterium bezüglich der direkten TCP-Abweichung ist:

$$J[\boldsymbol{u}(t), \boldsymbol{q}(t)] = 0 + \int_0^{t_f} \sum_{i \in \{x,y,z\}} \left(p_{\text{soll},i}(t) - p_i(t) \right)^2 \tag{5.17}$$

5.3.4 Systemdynamik - Zustand

Die rechte Seite $\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ der Systemdymnamik wird, wie in Abschnitt 5.2.3 beschrieben, durch Programmcode, der mittels der MBSLIB entwickelt wurde, bereitgestellt. Er enthält die Gleichungen (5.1) und (5.2). Die Größen, von denen der aktuelle Systemzustand abhängt, sind die Gelenkwinkel $\boldsymbol{q}^{6\times1}$ und Gelenkwinkelgeschwindigkeiten $\dot{\boldsymbol{q}}^{6\times1}$. Werden im Modell virtuelle Achsen berücksichtigt, müssen die erweiterten Gelenkwinkel $\tilde{\boldsymbol{q}}^{n\times1}$ und Gelenkwinkelgeschwindigkeiten $\dot{\tilde{\boldsymbol{q}}}^{n\times1}$ benutzt werden.

$$\boldsymbol{x}(t)^{2n\times1} = \begin{pmatrix} \tilde{\boldsymbol{q}}(t) \\ \dot{\tilde{\boldsymbol{q}}}(t) \end{pmatrix}, \; n \geq 6 \tag{5.18}$$

5.3.5 Systemdynamik - Steuerung

Die Steuerung kann über sechs Motormomente τ_i erfolgen. Aufgrund der Elastizitäten im Robotermodell kann $\boldsymbol{\tau}$ durch die Gleichung (5.2) ausgedrückt werden. Statt den Roboter somit direkt über die Motormomente zu steuern, kann er über die Pfadgröße $\tilde{\boldsymbol{q}}_u$ (in Gelenkwinkelkoordinaten) kontrolliert werden. Das hat den Vorteil, dass nach der Optimierung die Lösung für die Steuerung direkt den kompensierten Pfad darstellt. Er repräsentiert den eigentlichen Pfad, den der Roboter aufgrund der externen Kräfte ansteuern muss, um den Sollpfad zu erhalten. Nur die sechs Einträge in $\tilde{\boldsymbol{q}}_u$, die den angetriebenen Gelenken des Roboters zugeordnet sind, bilden die Steuerung. Die restlichen virtuellen Achsen werden nicht angesteuert.

$$\boldsymbol{u}^{6\times1}(t) \begin{cases} = \boldsymbol{\tau}(t) & \text{Steuerung mit Motormomenten} \\ = \tilde{\boldsymbol{q}}_u(t) & \text{Steuerung mit Kompensationspfad} \end{cases} \tag{5.19}$$

5.3.6 Anfangs- und Endbedingungen

Die Anfangs- und Endwerte für $\boldsymbol{q}$ sind über den Sollpfad, der in Gelenkwinkeln vorliegt, vorgegeben. Die Anfangs- und Endwerte für $\dot{\boldsymbol{q}}$ können zunächst als natürliche Randbedingungen formuliert und automatisch vom Optimierungsalgorithmus bestimmt werden. PSOPT benötigt dazu einen Bereich, in dem gesucht werden soll. Falls keinerlei Informationen darüber vorhanden sind, kann `-INF` bis `INF` verwendet

werden, was aber sehr ineffizient ist. Im vorliegenden Fall kann $\dot{\boldsymbol{q}}(0)$ und $\dot{\boldsymbol{q}}(t_f)$ über die Beschränkungen des physikalischen Roboters eingegrenzt werden, die auch in den Nebenbedingungen stecken.

$$\boldsymbol{q}(0) = \boldsymbol{q}_{\text{soll}}(0) \tag{5.20}$$
$$\dot{\boldsymbol{q}}(0) = \text{FREI} \tag{5.21}$$
$$\boldsymbol{q}(t_f) = \boldsymbol{q}_{\text{soll}}(t_f) \tag{5.22}$$
$$\dot{\boldsymbol{q}}(t_f) = \text{FREI} \tag{5.23}$$

Werden die erweiterten Gelenkwinkel $\tilde{\boldsymbol{q}}$ mit virtuelle Achsen q_i^x, q_i^y berücksichtigt, sind deren Randwerte ebenfalls als „FREI“ zu wählen.

5.3.7 Endzeit

Die Endzeit ist fest vorgegeben. Da der Sollpfad mit konstanter Geschwindigkeit abgefahren wird, ergibt sich automatisch daraus t_f. Aufgrund der Parametrisierung des Pfades mit der Bogenlänge wurde eine konstante Geschwindigkeit erreicht. Damit hat die Endzeit denselben Wert wie die Bogenlänge der Kurve.

$$t_f = L \tag{5.24}$$

Der Endeffektor erreicht dann die Geschwindigkeit $||\boldsymbol{v}_{tool}|| = 1$.

5.3.8 Zustands- und Steuerungsnebenbedingungen

Die Nebenbedingungen enthalten die Steuer- und Zustandsrestriktionen. Da die Endeffektorgeschwindigkeit bereits über eine spezielle Wahl der Diskretisierungspunkte festgelegt wurde, sind alle verbliebenen Nebenbedingungen über einfache explizite Restriktionen (upper / lower bounds) formulierbar. Weitere nichtlineare Nebenbedingungen sind nicht vorhanden. Die Zustandsbeschränkungen ergeben sich aus der Geometrie und der Konstruktion des realen Roboters. In den Herstellerdaten sind die Grenzen $q_{i,min}$ und $q_{i,max}$ der (angetriebenen) Gelenke und die Grenzen $\dot{q}_{i,min}$ und $\dot{q}_{i,max}$ der Gelenkwinkelgeschwindigkeiten zu finden. Diese können allerdings schon in der Sollpfadvorgabe berücksichtigt werden. Dabei ist zu beachten, dass die Kurve im Arbeitsraum des Roboters liegt. Anschließend ist zu überprüfen, ob die Gelenkwinkelwerte, die bei der inversen Kinematikberechnung entstehen, die genannten physikalischen Grenzen nicht verletzen. Im Optimalsteuerungsproblem können dann als Schranken für $\tilde{q}_i$ die engeren minimalen und maximalen Gelenkwinkelwerte des Sollpfads $\tilde{q}_{\text{soll } i,min}$ und $\tilde{q}_{\text{soll } i,max}$ benutzt werden. Die Einträge für die passiven Kippfreiheitsgrade q_i^x, q_i^y sind darin jeweils $= 0$. Alle DoF müssen noch durch eine Toleranz ϵ_q erweitert werden, da die auftretenden Abweichungen zu berücksichtigen sind. Die Zustandsbeschränkgungen lauten:

$$\tilde{q}_{\text{soll } i,min} - \epsilon_q \leq \tilde{q}_i \leq \tilde{q}_{\text{soll } i,max} + \epsilon_q \tag{5.25}$$
$$\dot{\tilde{q}}_{i,min} \leq \dot{\tilde{q}}_i \leq \dot{\tilde{q}}_{i,max} \tag{5.26}$$

Repräsentiert die Steuerung direkt die Motormomente τ_i, sind die Grenzen $\tau_{i,min}$ und $\tau_{i,max}$ durch die Leistungsdaten der Servomotoren gegeben. Diese stehen in den Datenblättern des Roboterherstellers. Repräsentiert die Steuerung den kompensierenden Pfad, sind die Beschränkungen nicht so klar vordefiniert, sondern hängen vom Prozess selbst ab. In den Fräskraftsimulationen und bei realen Experimenten

hat sich gezeigt, dass die Abweichungen zum Sollpfad, die beim Abfahren der Trajektorien entstehen, relativ klein sind. Das bedeutet, auch die kompensierende Trajektorie liegt nahe an der Sollbahn. Somit kann die Steuerungsbeschränkung als eine Erweiterung der maximalen und minimalen Gelenkwinkelwerte der Sollbahn mit einer größeren Toleranz $\epsilon_u \geq \epsilon_q$ definiert werden.

$$\left.\begin{array}{rl} \tau_{i,min} \leq u_i \leq \tau_{i,max}, & \text{falls } u_i = \tau_i \\ \tilde{q}_{\text{soll } i,min} - \epsilon_u \leq u_i \leq \tilde{q}_{\text{soll } i,max} + \epsilon_u, & \text{falls } u_i = \tilde{q}_{u,i} \end{array}\right\} \tag{5.27}$$

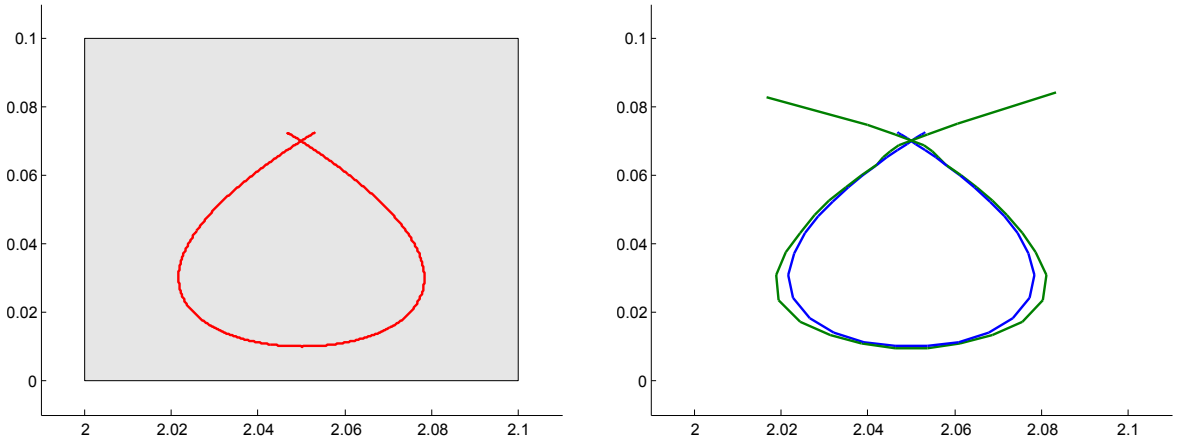
Nach jeder Optimierung sollte nochmals überprüft werden, ob die angenommenen Werte für ϵ_q und ϵ_u realistisch waren und die nicht verwendeten realen physikalischen Begrenzungen des Roboters ($q_{i,min}$, $q_{i,max}$, $\tau_{i,min}$, $\tau_{i,min}$) von der gefundenen Lösung verletzt werden.

5.3.9 Resultate

Dieser Abschnitt zeigt die Optimierungsergebnisse für den Beispielpfad $\boldsymbol{p}(t)$.

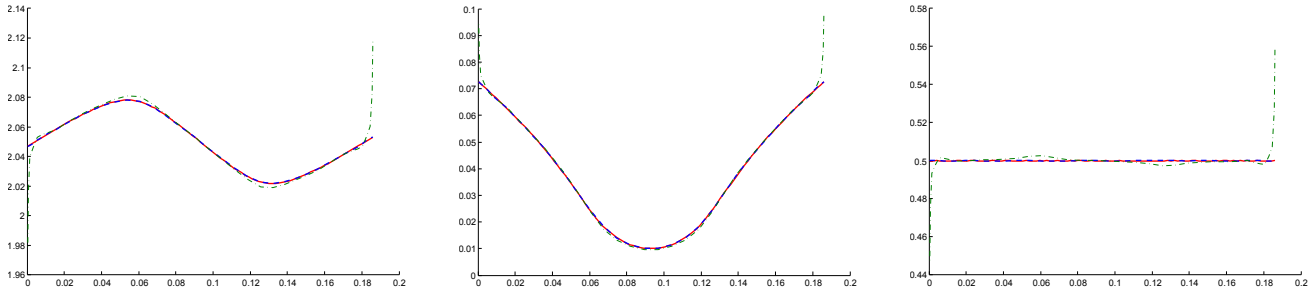
$$\boldsymbol{p}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} \left(\frac{1}{2} \cdot (t-2.5)^3 - 3 \cdot (t-2.5)\right) \cdot 0.01 + 2.05 \\ \left((t-2.5)^2 - 4\right) \cdot 0.01 + 0.05 \\ 0.5 \end{pmatrix}, \quad t \in [0,5] \tag{5.28}$$

Der Pfad liegt innerhalb einer Metallplatte mit 0.1 $[m] \times 0.1$ $[m]$ Kantenlänge. Die Bogenlänge des Pfads beträgt $L = 0.1858$ $[m]$. Er wird in 1000 Intervalle unterteilt (= 1001 Diskretisierungspunkte), deren Länge jeweils $\Delta s = 0.0001858$ $[m]$ beträgt. Für die Zielfunktion wird Formel (5.16) gewählt. Die Steuerung wird über die kompensierende Pfadangabe (5.19) realisiert. Der betrachtete Manipulator ist in allen sechs angetriebenen Gelenken elastisch, hat aber aufgrund der hohen Komplexität noch keine zusätzlichen Kippachsen. Die Randwerte stammen aus (5.20)-(5.23). Es zeigt sich, dass sich die Rechenzeit reduzieren lässt, wenn auch $\dot{\boldsymbol{q}}(0)$ und $\dot{\boldsymbol{q}}(t_f)$ fest vorgegeben werden. Deren Werte lassen sich über die Jacobimatrix des Manipulators oder in guter Näherung über finite Differenzen der beiden ersten bzw. beiden letzten Gelenkwinkelwerte bestimmen. Die Endzeit lautet wie in (5.24) $t_f = L$. Der Zustand wird über Bedingungen (5.25)-(5.26), die Steuerung über Bedingung (5.27) beschränkt. Es werden keine weiteren nichtlinearen Nebenbedingungen verwendet.



(a) Sollpfad auf der Werkstückoberfläche (b) Zustand und Steuerung (40 Punkte)

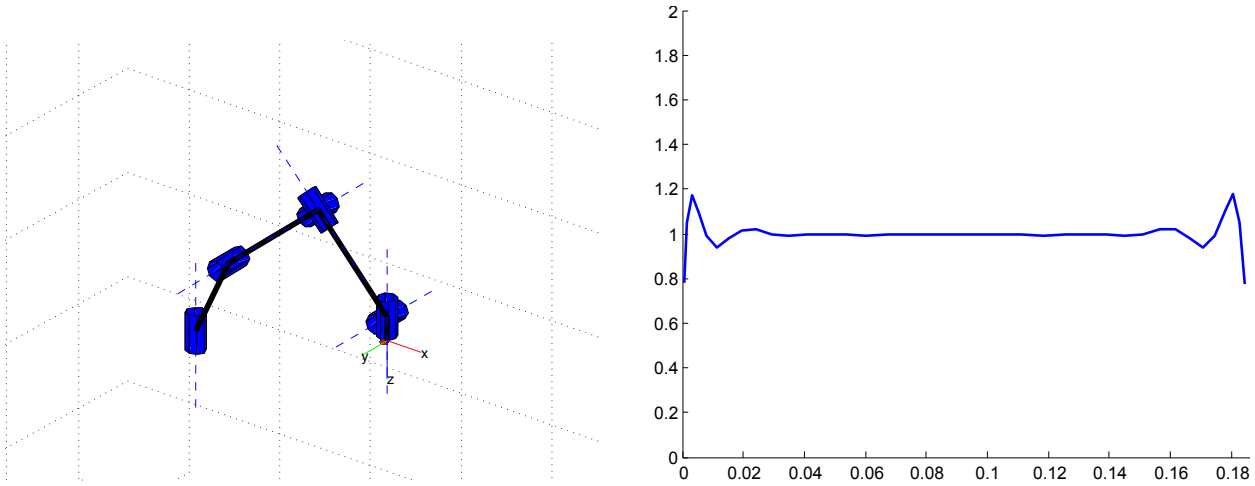
Abbildung 5.3: Fräspfad des Roboters in der x-y-Ebene;
rot: Sollpfad, grün: Kompensationspfad, blau: Istpfad



(a) x-Achse (b) y-Achse (c) z-Achse

Abbildung 5.4: Fräspfad des Roboters; rot: Sollpfad, grün: Kompensationspfad, blau: Istpfad

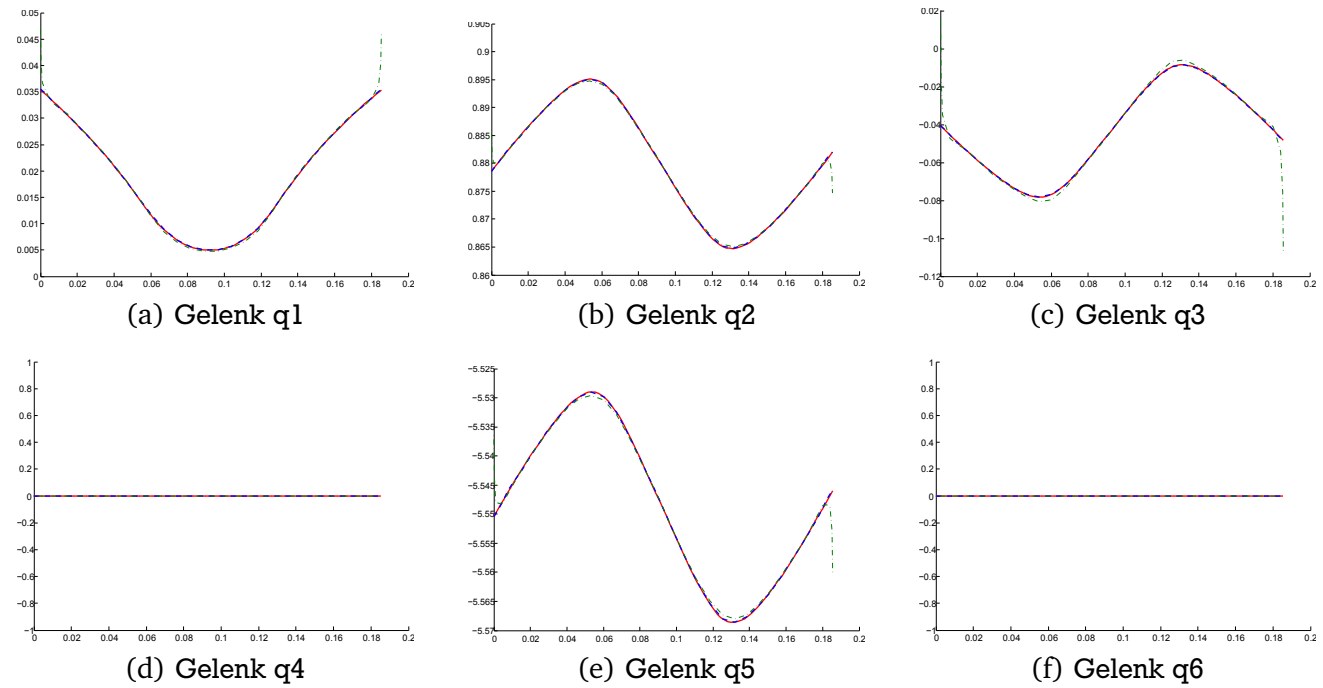
Abbildung 5.3 (a) zeigt den Sollpfad von Formel (5.28), welcher über die inverse Kinematik in Gelenkwinkelkoordinaten umgewandelt und dem Optimalsteuerungsalgorithmus vorgegeben wird. Dieser berechnet die Lösung von Abbildung 5.3 (b). Der grüne Verlauf ist der kompensierende Steuerungspfad (vgl. Abschnitt 5.3.5). Wird dieser Pfad abgefahren, erhält man den eigentlichen Pfadverlauf des Roboters, der blau eingezeichnet ist. Er soll möglichst genau den Sollpfad wiedergeben. In Abbildung 5.4 (a)-(c) ist zu sehen, dass der rote Sollpfad in allen drei Raumrichtungen gut mit dem blauen Istpfad übereinstimmt. Es ist auch zu erkennen, dass sich das Robotermodell trotz seiner Elastizitäten recht steif verhält, weswegen der grüne Steuerpfad sehr nah am roten Sollpfad liegt.



(a) Roboterkonfiguration (b) Endeffektorgeschwindigkeit

Abbildung 5.5: links: Darstellung der Roboterkonfiguration zum Endzeitpunkt t_f,
rechts: Endeffektorgeschwindigkeit während des Fräsens

Die Abbildungen 5.6 (a)-(f) zeigen den Soll-, Ist- und Steuerungsverlauf in Gelenkwinkelkoordinaten. Die Farbkodierung wurde analog zu den Abbildungen 5.4 (a)-(c) beibehalten. Auch hier lassen sich Unterschiede erst bei einer höheren Vergrößerung feststellen, was in Kapitel 6 vorgenommen wurde.



(a) Gelenk q1 (b) Gelenk q2 (c) Gelenk q3

(d) Gelenk q4 (e) Gelenk q5 (f) Gelenk q6

Abbildung 5.6: Gelenkwinkelverläufe des Roboters; rot: Sollpfad, grün: Kompensationspfad, blau: Istpfad

NLP Variablen	560
NLP Nebenbedingungen	505
Auswertungen der Zielfunktion	141
Auswertungen des Gradienten der Z.f.	75
Kollokationsmethode	Legendre
Gitterpunkte (fest)	40
CPU-Zeit	348.54 [sec]
Iterationen	74
Kostenwert	2.17764-e09

Tabelle 5.1: Ausgabe von PSOPT

5.4 Änderungen des Optimalsteuerungsproblems

Neben der genannten Forumulierung des Optimalsteuerungsproblems wurden noch weitere Möglichkeiten untersucht. Aus allen anderen bisher durchgeführten Optimierungen resultierten nur schlechtere oder vergleichbare Ergebnisse bei längerer Rechenzeit. Es folgen nun einige ausgewählte Beispiele. In diesen wurde stets die beschriebene Diskretisierung beibehalten. Nur die Unterschiede zur vorangegangenen Formulierung werden aufgezeigt.

Variante 1:
Die Zielfunktion misst die Abweichung zur kartesischen TCP-Sollposition (s. Formel (5.17)). Sprünge in den Gelenkkonfigurationen aufgrund von Mehrdeutigkeiten traten nicht auf. Die Beschränkungen der Gelenkwinkel und Gelenkwinkelgeschwindigkeit wurden immer noch auf den Sollpfad in Gelenkwin-

kelkoordinaten angepasst, was eine Erklärung dafür ist. Bei dieser Optimierung wurde der kartesische Sollpfad ähnlich gut wie in den Ergebnissen des letzten Abschnitts 5.3.9 eingehalten. Es trat allerdings ein zeitlicher Versatz zwischen Ist- und Sollwerten in Gelenkwinkelkoordinaten auf. Das hat zur Folge, dass die geforderte TCP-Geschwindigkeit größere Schwankungen bzgl. der Referenzgeschwindigkeit aufweist. Außerdem hat sich die Rechenzeit vervielfacht.

NLP Variablen	560
NLP Nebenbedingungen	505
Auswertungen der Zielfunktion	836
Auswertungen des Gradienten der Z.f.	267
Kollokationsmethode	Legendre
Gitterpunkte (fest)	40
CPU-Zeit	862.52 [sec]
Iterationen	266
Kostenwert	6.711005-e09

Tabelle 5.2: Ausgabe von PSOPT für Variante 1

Variante 2:

Um den Versatz in den resultierenden Gelenkwinkelverläufen der letzten Formulierung zu reduzieren, besteht die Zielfunktion hier sowohl aus der Abweichung der Gelenkwinkel als auch aus der Abweichung der kartesischen TCP-Position. Die Anteile können über den Faktor γ gewichtet werden. Die Resultate aus diesen Optimierungen haben gezeigt, dass erst für einen γ-Wert nahe bei 1 ($\gamma \approx 0.9$) der Versatz gut vermindert wird. Diese Formulierung bietet damit keine Vorteile gegenüber der Basisvariante (für die $\gamma = 1$ gilt), da die TCP-Abweichung kaum noch miteinfließt. Trotzdem wird eine deutlich höhere Rechenzeit benötigt.

$$J[\boldsymbol{u}(t), \boldsymbol{q}(t)] = 0 + \int_0^{t_f} \gamma \cdot \sum_{i=1}^{6} \omega_i \cdot \left(q_{\text{soll},i}(t) - q_i(t)\right)^2 + (1-\gamma) \cdot \sum_{i \in \{x,y,z\}} \left(p_{\text{soll},i}(t) - p_i(t)\right)^2 \, dt \quad (5.29)$$

NLP Variablen	560
NLP Nebenbedingungen	505
Auswertungen der Zielfunktion	979
Auswertungen des Gradienten der Z.f.	232
Kollokationsmethode	Legendre
Gitterpunkte (fest)	40
CPU-Zeit	812.7800 [sec]
Iterationen	231
Kostenwert	5.288319-e07

Tabelle 5.3: Ausgabe von PSOPT für Variante 2

Variante 3

Die Zielfunktion hat wieder die Form (5.17). Zusätzlich wurden nichtlineare Zustandsnebenbedingungen eingeführt. Sie beschränken den Sollpfad in Gelenkwinkeln nach oben und unten, sodass eine

ϵ-Umgebung um den Gelenkwinkelverlauf entsteht, die der Optimierungsalgorithmus bei seiner Suche nicht verlassen darf.

$$g_1(\boldsymbol{q}(t)) = (\boldsymbol{q}(t) - \boldsymbol{q}_{\text{soll}}(t) - \epsilon_g) \geq 0 \tag{5.30}$$
$$g_2(\boldsymbol{q}(t)) = (\boldsymbol{q}(t) - \boldsymbol{q}_{\text{soll}}(t) + \epsilon_g) \leq 0 \tag{5.31}$$
$$= -(\boldsymbol{q}(t) - \boldsymbol{q}_{\text{soll}}(t) + \epsilon_g) \geq 0 \tag{5.32}$$

Die damit generierten Ergebnisse zeigen nur eine leichte Veringerung des zeitlichen Versatzes zur Sollbahn. Aber für eine bessere Genauigkeit muss ϵ_g kleiner gewählt werden. Das kann zur Folge haben, dass der Optimierungsalgorithmus zur sehr eingeschränkt wird und keine Lösung mehr finden kann. Bei einem $\epsilon_g = 0.0001$ ist die Rechenzeit bereits enorm.

NLP Variablen	560
NLP Nebenbedingungen	745
Auswertungen der Zielfunktion	1652
Auswertungen des Gradienten der Z.f.	510
Kollokationsmethode	Legendre
Gitterpunkte (fest)	40
CPU-Zeit	1488.480 [sec]
Iterationen	509
Kostenwert	6.016006-e09

Tabelle 5.4: Ausgabe von PSOPT für Variante 3

Bemerkung 5.1. Die Kostenwerte der einzelnen Varianten des Optimalsteuerungsproblems sind nicht unmittelbar vergleichbar, da jeweils eine andere Zielfunktion verwendet wurde. Hierzu sei auf Kapitel 6 verwiesen.

5.5 Mehrphasenprobleme

Mit Mehrphasenproblemen lassen sich Aufgabenstellungen beschreiben, die nicht stetige Übergänge oder diskrete Zustandsänderungen beinhalten. Auch die Pfadverfolgung von Industrierobotern beim Fräsen können solche Phänomene enthalten. Das Eintauchen des Fräskopfes von außerhalb in das Werkstück hinein oder auch in umgekehrter Richtung lässt sich damit beschreiben. Dabei werden innerhalb eines Zeitschritts nahezu sprunghaft die Kräfte auf den Endeffektor ein- oder ausgeschaltet. Es kann nötig sein Parameter des Robotermodells während des Prozesses zu ändern. Auch der Sollpfad kann zu Mehrphasenformulierungen führen. Enthält er Kanten (nicht stetig differenzierbare Übergänge), kann der Roboter diesen nicht mit konstanter Geschwindigkeit abfahren. In einzelnen Motoren wären dafür unendlich große Drehmomente notwendig. Mit den üblichen Formulierungen von Optimalsteuerungsproblemen könnten somit aufgrund von Restriktionsverletzungen keine Lösungen oder nur Lösungen mit großen Abweichungen an den Phasenübergängen gefunden werden. Mit Inneren-Punkt-Bedingungen (4.75) an diesen Stellen kann dies vermieden werden. Hier ist beispielhaft das Fräsen einer Kante gezeigt.

$$\boldsymbol{p}(t) = \big(x(t), y(t), z(t)\big)^\top = \begin{cases} ((t+2.05), (t-0.2), 0.5)^\top, & \text{falls } t \leq 0.2 \\ ((-t+0.4)+2.05, (t-0.2), 0.5)^\top, & \text{falls } t > 0.2 \end{cases}, \quad t \in [0, 0.4] \tag{5.33}$$

Beim Übergang zum Zeitpunkt $t_s = 0.2$ kann die Geschwindigkeit nicht aufrecht gehalten werden. Deshalb wird eine Übergangsbedingung aufgestellt, die nur einen stetigen Verlauf der Gelenkwinkel $\boldsymbol{q}$ fordert.

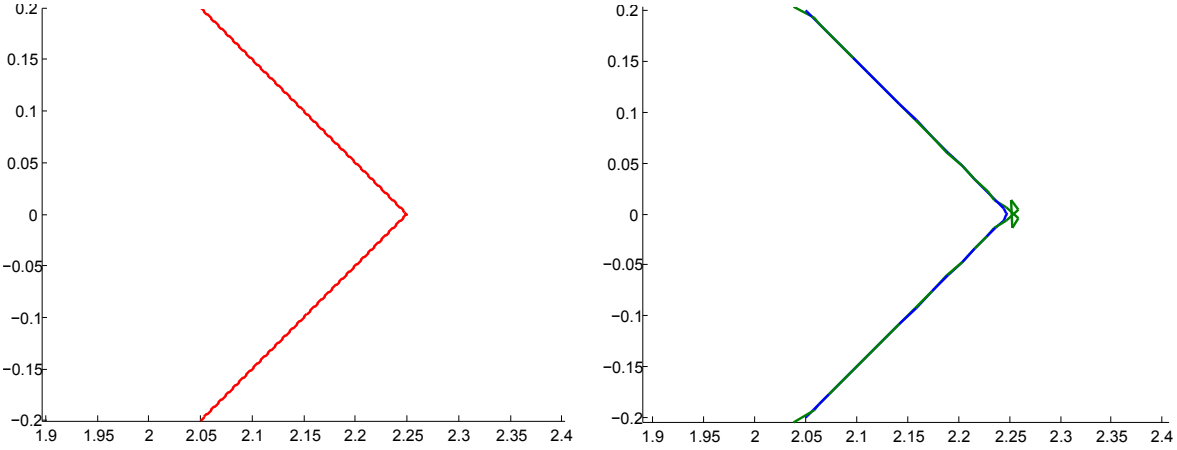
$$\boldsymbol{r}(\boldsymbol{q}^-(t_s), \boldsymbol{q}^+(t_s), t_s) = \boldsymbol{q}^-(t_s) - \boldsymbol{q}^+(t_s) = 0 \tag{5.34}$$

Ein stetiger Übergang für $\dot{\boldsymbol{q}}$ ist nicht möglich und wird deshalb nicht verlangt. In PSOPT muss zusätzlich gefordert werden, dass die Zeit t ebenfalls stetig an diesem Übergang ist. Für die Endzeit $t_f^{(1)} = t_s^- = 0.2$ der ersten Phase und der Startzeit $t_0^{(2)} = t_s^+ = 0.2$ der zweiten Phase muss gelten:

$$r(t_f^{(1)}, t_0^{(2)}) = t_f^{(1)} - t_0^{(2)} = 0 \tag{5.35}$$

Der Zeitpunkt t_s des Übergangs ist hier bereits bekannt, was aber nicht immer der Fall sein muss.

5.5.1 Resultate



(a) Sollpfad (b) Istpfad und Steuerung

Abbildung 5.7: Fräspfad des Roboters; rot: Sollpfad, grün: Kompensationspfad, blau: Istpfad

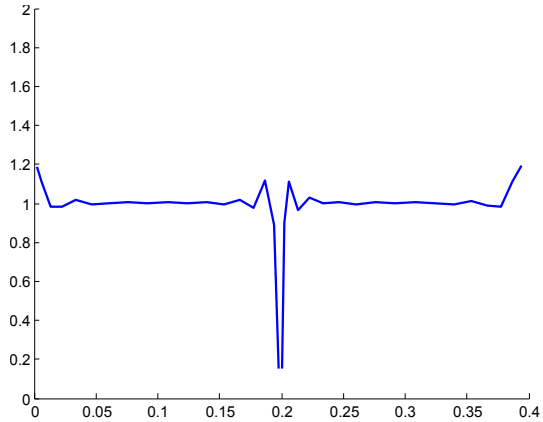


Abbildung 5.8: Endeffektorgeschwindigkeit im Mehrphasenproblem

In Abbildung 5.8 sieht man, wie die Endeffektorgeschwindigkeit an der Kante des Pfades einbricht. Dies geschieht aufgrund von Bedingung (5.34), die nur einen stetigen Übergang der Gelenkwinkel und somit der TCP-Position fordert. Effizientere Bedingungen, die einen stetigen TCP-Verlauf gewährleisten und gleichzeitig den Geschwindigkeitseinbruch möglichst gering halten, sind für zukünftige Weiterentwicklungen interessant. In Abbildung 5.7 (b) ist außerdem eine Schleife im Steuerungspfad (grün) zu erkennen. Ein besserer Steuerungsverlauf ist in diesem Fall vielleicht über eine lokale Kollokationsmethode mit Gitterverfeinerung zu erreichen. Mit zusätzlichen Gitterpunkten in diesen kritischen Übergangsbereichen können an solchen Stellen genauere Verläufe berechnet werden.

NLP Variablen	560
NLP Nebenbedingungen	513
Auswertungen der Zielfunktion	200
Auswertungen des Gradienten der Z.f.	89
Kollokationsmethode	Legendre
CPU-Zeit	329.3000 [sec]
Iterationen	89

	Phase 1	Phase 2
Gitterpunkte (fest)	20	20
Endzeit	0.2	0.4
Kostenwert	3.876056-e08	3.888513-e08

Tabelle 5.5: Ausgabe von PSOPT

6 Bewertung

6.1 Beurteilung der erzielten Ergebnisse

Es bleibt nun zu klären, in wie weit die im letzten Kapitel erreichten Ergebnisse Einfluss auf die Qualität der Pfadverfolgung genommen haben. In Abbildung 5.4 ist bereits zu sehen, dass für die einzelnen Richtungskomponenten der Soll- und Istpfad gut übereinstimmen. Für die Darstellung in Gelenkkoordinaten ist dies am Beispiel von q_3 nochmals genauer im folgenden Bild illustiert. Auch hier ist die gute Übereinstimmung des roten eingezeichneten Sollpfads mit dem blauen Istverlauf sehr deutlich. In den beiden vergrößerten Bereichen ist die Auswirkung der kompensierenden Steuerungstrajektorie nun besser zu erkennen. Der grün eingezeichnete Steuerungsverlauf liegt deutlich neben dem roten Sollpfad. Wird nun die Steuerungstrajektorie vom Roboter abgefahren, entsteht aufgrund der externen Kräfte der blaue Verlauf. Dieser wurde durch die versetzt liegende Steuerung auf den Sollpfad „gezogen“. Bei einer noch höheren Vergrößerung kann man sehen, dass an manchen Punkten der Istpfad leicht über oder unter dem Sollpfad liegt. An diesen Stellen überkompensiert die Steuerung die externe Kraft oder ist vom Optimierungsverfahren zu gering ausgelegt worden. In welchem Maße dies geschieht, soll in diesem Kapitel betrachtet werden. Außerdem ist zu beachten, dass die berechnete Lösung nur an diskreten Punkten vorliegt. Die Steuerung (grün) und der Zustand (blau) wurden nur für 40 Gitterpunkte berechnet. Eine höhere Auflösung ist momentan wegen technischer Beschränkungen nicht möglich. Der große Steuerungsausschlag, der jeweils zu Beginn und am Ende auftritt, konnte bisher nicht effektiv reduziert werden. Da die Zustände an den Rändern fest vorgegeben sind und dort der Sollpfad mit dem Istpfad bereits übereinstimmt, ist der Wert der Steuerung an diesen Punkten unerheblich. Vermutlich setzt der Optimierungsalgorithmus die Steuerwerte dort beliebig fest oder läuft in die Beschränkungen. Für eine spätere Weiterverarbeitung der Steuertrajektorie ist es empfehlenswert jeweils den ersten und letzten Punkt durch passendere Werte zu ersetzen.

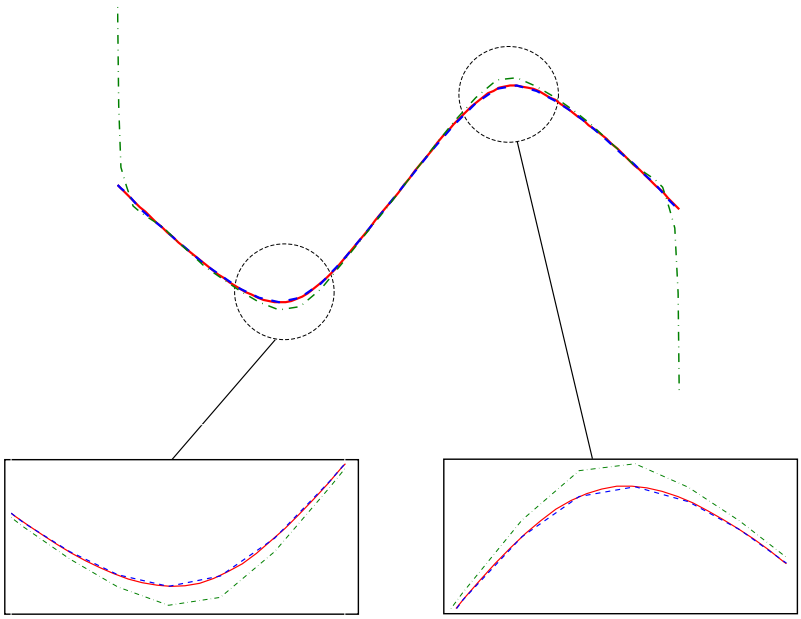


Abbildung 6.1: Exemplarische Betrachtung des Gelenkwinkelverlaufs q_3;
rot: Sollpfad, grün: Kompensationspfad, blau: Istpfad

Die genaue Messung des absoluten Fehlers auf dem von PSOPT generierten Zeitgitter erfordert einige Zwischenschritte. Dabei muss wiederum auf numerische Verfahren zurückgegriffen werden. Die somit auftretenden Ungenauigkeiten verfälschen die Messung der Pfadabweichung. Der durchgeführte Ablauf wird in Abbildung 6.2 gezeigt.

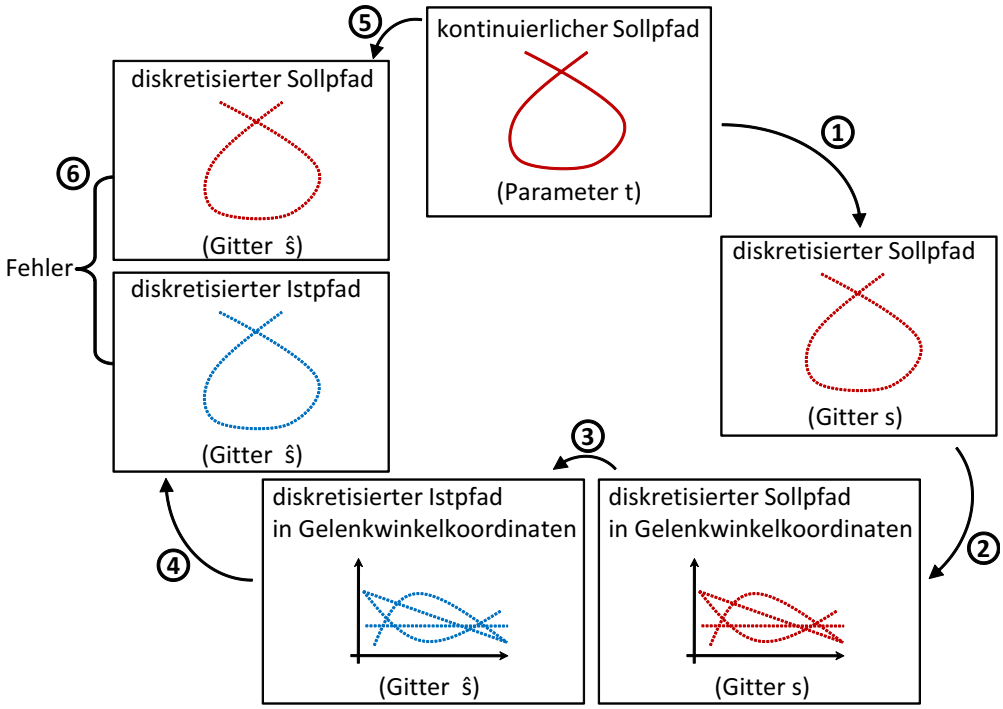


Abbildung 6.2: Ablauf zur Messung der Pfadabweichung

1. Der kontinuierliche kartesische Sollpfad $\boldsymbol{p}(t),\ t \in [0, \overline{t_f}]$ wird diskretisiert, wobei eine Parametrisierung mit der Bogenlänge durchgeführt wird, sodass der Abstand der einzelnen Diskretisierungspunkte jeweils gleich groß ist. Dazu ist in jedem Schritt ein Integral und eine Nullstelle numerisch zu bestimmen. Am Ende liegt eine diskrete Beschreibung des kartesischen Sollpfads $\boldsymbol{p}_i(s),\ s \in [0, L]$ an den Stellen $s_0, \ldots, s_{n_d}$ vor.

2. Für das in Abschnitt 5.3 beschriebene Basisoptimalsteuerungsproblem müssen die TCP-Punkte $\boldsymbol{p}_i$ in Gelenkwinkelkoordinaten umgewandelt werden. Dies geschieht über das numerische Lösen eines nichtlinearen Gleichungssystems. Der Sollpfad liegt als diskreter Gelenkwinkelverlauf in $\boldsymbol{q}_i$ vor.

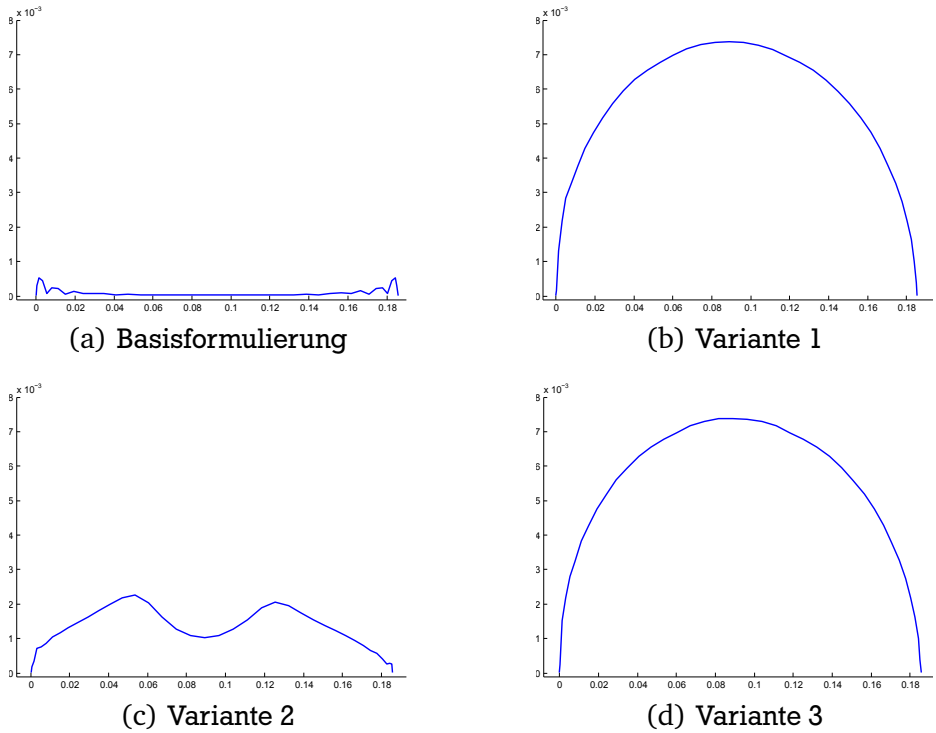
3. Der numerische Optimalsteuerungalgorithmus berechnet den Istpfad in Gelenkwinkeln $\boldsymbol{q}_j(\hat{s}),\ \hat{s} \in [0, L]$ auf einem eigenen Zeitgitter mit $\hat{s_0}, \ldots, \hat{s_{39}}$.

4. Die Gelenkwinkel werden mit einer Vorwärtskinematiktransformation in TCP-Koordinaten umgerechnet. Daraus resultiert der Istpfad $\boldsymbol{p}_j(\hat{s}),\ \hat{s} \in [0, L]$.

5. Um die Differenz zwischen Sollpfad und Istpfad messen zu können, muss der Sollpfad nun auf dem PSOPT Zeitgitter ausgewertet werden. Eine kontinuierliche und an beliebigen Punkten auswertbare Funktion des Sollpfads liegt aber nur für den Parameter t vor. Das von PSOPT generierte Zeitgitter muss nun erst von $\hat{s}$ nach t transformiert werden. Hierfür ist nochmals das numerische Auswerten eines Integrals und einer Nullstelle für jeden Gitterpunkt nötig.

6. Am Ende liegen die Werte des Sollpfades und des Istpfades an korrespondierenden Zeitpunkten vor, an denen die Abweichung gemessen werden kann.

6.1.1 Resultierende Pfadabweichungen



(a) Basisformulierung (b) Variante 1

(c) Variante 2 (d) Variante 3

Abbildung 6.3: Pfadabweichungen auf dem von PSOPT generierten Diskretisierungsgitter $\hat{s}$

Wie in Abbildung 6.3 zu sehen ist, schneidet die Basisformulierung am besten ab. Ihre Zielfunktion ist direkt von den Gelenkwinkeln des Systemzustands abhängig und besitzt somit eine recht einfache Darstellung. Variante 1 und Variante 3 haben jeweils die Zielfunktion (5.17), welche die TCP-Abweichung misst. Da ihr Verlauf somit von der Systemdynamik abhängt, ist sie wesentlich komplexer. Der hohe Grad an Nichtlinearität führt dazu, dass der Optimierungsalgorithmus sie nur schlecht minimieren kann. Die zusätzliche Nebenbedingung (5.32) von Variante 3 zeigt kaum eine Auswirkung. Die ϵ-Umgebung ist immer noch zu grob um den Algorithmus in ein besseres Minimum zu lenken. Der resultierende Fehlerverlauf von Variante 3 ist deswegen nahezu identisch zu dem aus Variante 1. Das kombinierte Zielfunktionskriterium von Variante (5.29) verhält sich etwas besser. Dafür scheint jedoch nur die hohe Gewichtung des Anteils aus der Basisformulierung verantwortlich zu sein. Der zweite Term wirkt eher störend, sodass der Fehlerverlauf immer noch wesentlich schlechter gegenüber dem Verlauf aus der Basisformulierung ist.

6.2 Andere Bewertungskriterien

Eine objektive und exakte Bewertung des Verfahrens ist gegenwärtig aufgrund der zur Verfügung stehenden Werkzeuge nur schwer durchzuführen. Es existiert ein MATLAB Programm zur Frässimulation basierend auf einem numerischen Nagelbrettmodell, welches ein genaueres Verhalten als die in Gleichung (5.3) beschriebene Approximation bietet. Um die Frage zu klären, welche Verbesserung eine Pfadkompensation mit sich bringt, können damit Simulationen durchgeführt werden. Zum einen wird das Fräsen mit der kompensierenden Trajektorie simuliert, zum anderen das Fräsen mit der reinen Solltrajektorie. Dadurch kann der Vorteil einer Pfadkompensation deutlich gemacht werden. Allerdings sind die Rechenzeiten dafür momentan noch sehr hoch und ein spezielles Aufbereiten der Daten des Fräspfads notwendig. Beispielsweise müssen Überschleifradien und die Taktfrequenz der Robotersteuerung eingehalten werden. Darüber hinaus ist ein direkter Vergleich der entstehenden Nagelbrettmuster denkbar. Die Unterschiede zwischen gewünschtem Sollnagelbrett und erreichtem Istnagelbrett eignen sich ebenfalls als Maß zur Bewertung des Verfahrens. Dabei kann der Fehler noch weiter in die Kategorien „falsch getroffener Nagel“ (false positive) und „nicht getroffener Nagel“ (false negative) unterteilt werden. Das MATLAB Skript verfügt allerdings derzeitig nicht über eine solche Funktionalität. Letztendlich ist entscheidend, wie sich das Verfahren in der Realität am echten Roboter verhält. Zwar sind in Experimenten bereits Fräsversuche unternommen worden, jedoch wurden dazu noch nicht die durch Optimalsteuerung ermittelten Kompensationstrajektorien verwendet. Die Durchführung solcher Fräsversuche in der Realität ist zur Zeit noch sehr aufwendig. Dies könnte unter anderem durch eine effizientere Gestaltung der Schnittstellen und Abläufe verbessert werden. Eine genauere Untersuchung und Bewertung der real erzielten Fräsergebnisse bietet sich somit als Gegenstand zukünftiger Arbeiten an.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung des Lösungsansatzes

In dieser Masterthesis wurde eine Methode vorgestellt um die Pfadabweichung von Industrierobotern während des Fräsprozesses zu minimieren. Der Vorteil von Industrierobotern zu diesem Zweck liegt hauptsächlich in den niedrigen Anschaffungskosten und dem großen Arbeitsraum. Herkömmliche Werkzeugmaschinen können nur kleine Bauteile bearbeiten. Roboter können neben dem Fräsen noch weitere Aufgaben übernehmen, was Unternehmen in ihrer Produktionsplanung flexibler macht.
Optimale Steuerung bietet die Möglichkeit ein allgemein gültiges Verfahren zu entwickeln, das nicht auf einem bestimmten Robotertyp basiert oder andere Annahmen oder ein zusätzliche technische Ausrüstung vorraussetzt. Die Entwicklung und der aktuelle Stand der Optimalen Steuerung wurden erläutert, die häufigsten Einsatzgebiete beschrieben und eine Übersicht der Lösungsverfahren gegeben. Außerdem sind weitere Arbeiten, die sich mit der Pfadverfolgung durch Manipulatorarme beschäftigen, vorgestellt und die Gemeinsamkeiten und Unterschiede zum hier verfolgten Ansatz herausgestellt worden. Dabei traten vor allem die in dieser Arbeit zu berücksichtigenden hohen externen Kräfte als spezielle Problemcharakteristik hervor. Diese entstehen beim Zerspanen von Metallen und lassen sich mit analytischen, empirischen oder mechanistischen Beschreibungen ermitteln. Für genaue Berechnungen sind numerische Verfahren wie Nagelbrettmodelle nötig, die aber sehr rechenintensiv sind.
Wenn die externen Kräfte bekannt sind, kann man ihren Einfluss auf die Roboterstruktur bestimmen. Roboter arbeiten aufgrund von Elastizitäten in den Gelenken noch zu ungenau. Für eine Kompensationsstrategie ist es deshalb wichtig bei der Modellierung des Roboters dieses elastische Verhalten mitzuberücksichtigen. Das vorliegende Robotermodell bildet die Elastizität in allen sechs Gelenken ab und verfügt über die Möglichkeit weitere orthogonal zu den Drehachsen liegende virtuelle Kippachsen hinzuzufügen. Die zugrunde liegende Herleitung der starren Roboterdynamik wurde auf zwei unterschiedliche Weisen in Kapitel 3 gezeigt.
Die Formulierung und Berechnung eines Optimalsteuerungsproblems erlaubt es nun eine Beschreibung der Steuerung zu finden, welche die Abweichung zwischen gewünschtem Sollpfad und tatsächlich entstehendem Istpfad minimiert. Diese Beschreibung enthält den Kompensationspfad. Wird dieser während des Fräsens anstatt des eigentlichen Sollpfads angesteuert, kann nahezu der gewünschte Verlauf ohne Abweichungen erreicht werden. Das zur Lösung eingesetzte direkte Verfahren wandelt das dynamische Optimierungsproblem in ein hochdimensionales NLP Problem um. Die mathematischen Grundlagen der Optimalen Steuerung, die als spezielle Variationsprobleme angesehen werden können, und die Grundlagen zur Berechnung von NLP Problemen wurden in Kapitel 4 gegeben.
Kapitel 5 beinhaltet schließlich, wie mit Hilfe von PSOPT das Optimalsteuerungsproblem aufgestellt und gelöst, das Dynamikmodell mit Hilfe der MBSLIB erstellt und Ableitungen über automatisches Differenzieren mit ADOL-C ermittelt wurden. Ausgewählte Formulierungsmöglichkeiten des Optimalsteuerungsproblems wurde erläutert. Resultate für einen vorgegebenen Beispielpfad wurden gezeigt und in Kapitel 6 bewertet.

7.2 Bewertung der entwickelten Methodik

Die Vorteile des auf Optimalsteuerung basierenden Ansatzes liegen hauptsächlich in der Flexibilität und der Adaptionsmöglichkeit auf unterschiedliche Problemfälle. Verschiedene Robotermodelle, Beschränkungen und Nebenbedingungen, Pfadvorgaben sowie externe Krafteinflüsse können leicht an das Opti-

malsteuerungsproblem angekoppelt und ausgetauscht werden. So ist es möglich eine unabhängige fehlerminimierende Steuerungsstrategie zu finden. Optimalsteuerungsprobleme sind auf herkömmlicher Computerhardware effizient berechenbar und mit speziellen (teils frei verfügbaren) Softwaretools elegant zu implementieren. Auch für umfangreiche Problemstellungen ist diese Lösungsmethode prinzipiell geeignet. Da alle Berechnungen vor dem eigentlichen Fräsen durchgeführt werden und die Steuerungsstrategie offline bestimmt wird, sind keine Echtzeitbedingungen an den Prozess geknüpft.
Der Nachteil des beschriebenen Ansatzes liegt in der Notwendigkeit, dass ein sehr genaues Dynamikmodell des Roboters vorliegen muss. Da die Berechnung der Kompensation vor dem Beginn der Pfadverfolgung erfolgt, kann auf Abweichungen während des Prozesses, die aufgrund nicht berücksichtigter Störeinflüsse oder Modellungenauigkeiten entstehen, nicht mehr reagiert werden. Mit Hilfe von Softwarebibliotheken kann allerdings der Implementierungsaufwand eines realistischen und komplexen Dynamikmodells erheblich reduziert werden. Jedoch müssen die Parameter dieses Modells, wie z.B. Schwerpunktlagen, Trägheiten und Steifigkeiten, mit hoher Genauigkeit bestimmt werden. Darüber hinaus führt das Ankoppeln einer exakten Fräskraftberechnung an das Optimalsteuerungsproblem zu sehr hohen Rechenzeiten. Wird stattdessen eine Approximation dieser Kräfte verwendet, muss erst überprüft werden, ob die im Zusammenspiel mit dem gesamten Simulationsmodell berechneten Ergebnisse realistisch sind.

7.3 Praktische Aspekte der Implementierung

Die Verwendung und gemeinsame Benutzung der in Abschnitt 5.2 vorgestellten Programmkomponenten stellte zu Beginn dieser Arbeit eine Herausforderung dar, weil meist keine vordefinierten Schnittstellen zur Verfügung standen. Dies führte zunächst zu einigen technischen Hindernissen, welche jedoch bereits durch eigene Modifikationen angemessen behoben werden konnten. Derzeit existieren jedoch noch kleinere Limitationen, welche den Lösungsprozess in seiner Effizienz einschränken. Sowohl die Dynamikbibliothek MBSLIB als auch die Software ADOL-C befinden sich noch in der Entwicklung. Es ist anzunehmen, dass eine Verbesserung dieser Softwarekomponenten insgesamt eine Verbesserung und Vereinfachung der Implementierung des Lösungsansatzes nach sich ziehen. Die verwendete Dynamikbibliothek MBSLIB besitzt den Vorteil bereits ADOL-C zu unterstützen. Der nötige Datentyp `adoublе` ist somit schon vorhanden. Auch die Optimalsteuerungssoftware PSOPT unterstützt ADOL-C. Das doppelte Einbinden von ADOL-C (anfangs unterschiedlicher Versionen) verursachte Kompatibilitätsprobleme. Durch eine Anpassung der ADOL-C Headerdateien wurde dies aber erfolgreich behoben. Des Weiteren scheint es Einschränkungen bei der Verwendung von ADOL-C im Zusammenhang mit der MBSLIB zu geben. Somit konnten bisher nur Probleme auf einem relativ groben Zeitgitter ($\approx$ 40 Knoten) gelöst werden. Außerdem wird in jeder Iteration des Optimalsteuerungsproblems das komplette Robotermodell neu aufgebaut. Das Zwischenspeichern des Modells mit aktueller Konfiguration mittels einer globalen Variablen schlug fehl. Dies wirkt sich auf die Rechenzeit aus und hat einen großen Einfluss auf die zusammengestellten CPU-Zeiten von Tabellen 5.1, 5.2, 5.3 und 5.4. Es ist also zu erwarten, dass bei einer verbesserten Dynamikmodellanbindung diese Rechenzeiten noch weiter reduziert werden können.

7.4 Ausblick

Diese Thesis kann als Grundlage für Erweiterungen und Verbesserungen dienen. Der zunächst wichtigste Punkt, der in einer zukünftigen Arbeit untersucht werden sollte, ist die ausführliche Validierung der Methode am realen Roboter. Erst dadurch können Schwachstellen des Lösungsansatzes zuverlässig

identifiziert und daraufhin einzelne Komponenten überarbeitet und angepasst werden. Das am besten geeignete Gütekriterium zur Bewertung des Verfahrens ist der direkte Vergleich zwischen der gewünschten Werkstückgeometrie und der durch das Fräsen hergestellten Geometrie. Diese Gegenüberstellung gestaltet sich wiederum aufwendig. Das gefertigte Bauteil muss dabei möglichst genau vermessen werden. Die so erhaltenen Daten können mit der geforderten Geometrie in einem CAD-System verglichen werden. Das Volumen, das durch die Differenzmengenoperation der beiden Geometrien entsteht, ist ein robustes Maß zur Bewertung der in der Realität erhaltenen Ergebnisse. In dieser Arbeit konnte bereits von einer hohen Modellgenauigkeit ausgegangen werden. Eine weitere Verbesserung des Modells ist über Parameteridentifikations- und Schätzverfahren durchführbar. Formulierungsansätze von Mehrphasenprobleme wurden beispielhaft gezeigt. Eine genauere Untersuchung von Problemstellungen, die sich in mehrere Phasen unterteilen, ist sehr sinnvoll. Sollpfade, die Kanten oder nicht stetig differenzierbare Abschnitte enthalten, sowie Übergänge des Fräskopfs aus und in das Werkstück hinein sind oft auftretende Fälle beim Fräsen und eignen sich für genauere Betrachtungen. Die Komplexität des Nagelbrettmodells zur Fräskraftberechnung wurde im Optimalsteuerungsproblem durch die Benutzung einer Approximationsvorschrift umgangen. Die Entwicklung einer verbesserten Fräskraftapproximation, welche die Werkstückgeometrie berücksichtigt und gleichzeitig effizient ist, wäre jedoch von Vorteil. Eine Überschneidung im Pfad, wie sie z.B. in Abbildung 5.3 zu sehen ist, führt momentan zu keinem Unterschied im Kraftverlauf, obwohl an dieser Stelle bereits Material abgetragen wurde.

A DH-Parameter

Mit dem DH-Algorithmus kann die Kinematik eines Roboters aufgestellt werden. Ziel ist es die Parametertabelle (vgl. Tabelle 3.1) bzw. die Transformationsmatrix (3.4) herzuleiten. Eine genaue Beschreibung des Verfahrens ist in [54], [71], [55] zu finden.

- d_i ist der Abstand zwischen der x_i-Achse und dem Ursprung des Koordinatensystems S_{i-1} entlang der z_{i-1}-Achse (bei Schubgelenken variabel).
- a_i ist der Abstand zwischen der z_{i-1}-Achse und dem Ursprung des Koordinatensystems S_i entlang der x_i-Achse.
- θ_i ist der Winkel zwischen der x_{i-1}-Achse und der x_i-Achse, gemessen um die z_{i-1}-Achse (bei Drehgelenken variabel).
- α_i ist der Winkel zwischen der z_{i-1}- und der z_i-Achse, gemessen um die x_i-Achse.

B Analytisches Lösen einer DGL durch Variation der Konstanten

Das Verfahren wird anhand von DGL (3.20) erläutert:

◦ Zunächst wird die DGL auf Standardform $x'' + ax' + bx = f(t)$ gebracht.

$$\ddot{x}(t) + \frac{k}{m}x(t) = \frac{1}{m}u(t) \tag{B.1}$$

◦ Die Lösung $x_h(t)$ des homogenen Problems $x'' + ax' + bx = 0$ wird bestimmt. Dazu werden die Nullstellen des *Charakteristischen Polynoms* $\lambda_{1,2} = -\frac{a}{2} \pm \sqrt{(\frac{a}{2})^2 - b}$ berechnet.

$$\lambda_{1,2} = -\frac{k}{2m} \pm \sqrt{\frac{k^2}{4m^2}} \tag{B.2}$$

$$\lambda_1 = 0 \tag{B.3}$$

$$\lambda_2 = -\frac{k}{m} \tag{B.4}$$

◦ Daraus lässt sich die Lösungsbasis bestimmen.

$$(i) \quad \lambda_1, \lambda_2 \in \mathbb{R}, \lambda_1 \neq \lambda_2 \quad \rightarrow \quad x_1(t) = e^{\lambda_1 t},\ x_2(t) = e^{\lambda_2 t} \tag{B.5}$$

$$(ii) \quad \lambda_1, \lambda_2 \in \mathbb{R}, \lambda_1 = \lambda_2 \quad \rightarrow \quad x_1(t) = e^{\lambda_1 t},\ x_2(t) = te^{\lambda_1 t} \tag{B.6}$$

$$(iii)\ \lambda_1 = \alpha + i\beta, \lambda_2 = \overline{\lambda_1} \quad \rightarrow \quad x_1(t) = e^{\alpha t}\cos(\beta t),\ x_2(t) = e^{\alpha t}\sin(\beta t) \tag{B.7}$$

◦ Die homogene Lösung genügt dem Ansatz $x_h(t) = c_1 x_1(t) + c_2 x_2(t)$. Aufgrund von (i) ist:

$$x_h(t) = c_1\ e^{0t} + c_2\ e^{-\frac{k}{m}t} \tag{B.8}$$

$$= c_1 + c_2\ e^{-\frac{k}{m}t} \tag{B.9}$$

◦ Als nächstes wird die partikuläre Lösung $x_p(t)$ bestimmt. Hierfür wird die *Wronsky-Determinante* $W(t) = x_1(t)x_2'(t) - x_2(t)x_1'(t)$ benötigt.

$$W(t) = 1 \cdot (-\frac{k}{m} \cdot e^{-\frac{k}{m}t}) - e^{-\frac{k}{m}t} \cdot 0 \tag{B.10}$$

$$= -\frac{k}{m} \cdot e^{-\frac{k}{m}t} \tag{B.11}$$

◦ Die partikuläre Lösung genügt dem Ansatz $x_p(t) = -x_1(t) \int_0^t \frac{x_2(\tau) \cdot f(\tau)}{W(\tau)}\, d\tau + x_2(t) \int_0^t \frac{x_1(\tau) \cdot f(\tau)}{W(\tau)}\, d\tau$

$$x_p(t) = -1 \int_0^t \frac{(e^{-\frac{k}{m}\tau}) \cdot \frac{1}{m} u(\tau)}{-\frac{k}{m} \cdot e^{-\frac{k}{m}\tau}}\, d\tau + e^{-\frac{k}{m}t} \int_0^t \frac{1 \cdot \frac{1}{m} u(\tau)}{-\frac{k}{m} \cdot e^{-\frac{k}{m}\tau}}\, d\tau \tag{B.12}$$

$$= \frac{1}{k} \int_0^t u(\tau)\, d\tau - e^{-\frac{k}{m}t} \int_0^t \frac{1}{k} u(\tau) \cdot e^{\frac{k}{m}\tau}\, d\tau \tag{B.13}$$

$u(t)$ wurde noch nicht festgelegt, weswegen die Integrale nicht aufgelöst werden.

◦ Die vollständige Lösung der inhomogenen DGL ergibt sich aus der Summe der homogenen und der partikulären Lösung $x(t) = x_p(t) + x_h(t)$.

$$x(t) = \frac{1}{k} \int_0^t u(\tau)\, d\tau - e^{-\frac{k}{m}t} \int_0^t \frac{1}{k} u(\tau) \cdot e^{\frac{k}{m}\tau}\, d\tau + c_1 + c_2\, e^{-\frac{k}{m}t} \tag{B.14}$$

C Karush-Kuhn-Tucker Bedingungen

Die Karush-Kuhn-Tucker (KKT) Bedingungen [64], [1] bilden die notwendigen Bedingungen 1. Ordnung an die Lösung $\boldsymbol{p}^*$ eines restringierten NLP Problems (4.7)-(4.9).

$$\nabla_{\boldsymbol{p}} L(\boldsymbol{p}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*) = 0 \tag{C.1}$$

$$\Leftrightarrow \nabla \phi(\boldsymbol{p}^*) + \sum_{i=1}^{n_a} \mu_i^* \cdot \nabla a_i(\boldsymbol{p}^*) + \sum_{j=1}^{n_b} \lambda_j^* \cdot \nabla b_j(\boldsymbol{p}^*) = 0 \tag{C.2}$$

$$\mu_i^* \cdot a_i(\boldsymbol{p}^*) = 0\,, \quad i = 1, \ldots, n_a \tag{C.3}$$

$$\lambda_j^* \cdot b_j(\boldsymbol{p}^*) = 0\,, \quad j = 1, \ldots, n_b \tag{C.4}$$

$$\lambda_j^* \geq 0 \tag{C.5}$$

Die notwendige Bedingung 2. Ordnung stellt sicher, dass es sich bei der gefundenen Lösung $\boldsymbol{p}^*$ um ein Minimum handelt, indem die Krümmung an dieser Stelle untersucht wird. Die Hessematrix der Lagrangefunktion muss dort positiv definit sein. Eine allgemeinere Formulierung für alle zulässigen Suchrichgungen $\boldsymbol{z} \in \mathbb{R}^{n_p} \setminus \{0\}$ lautet:

$$\boldsymbol{z}^\top \cdot H_L(\boldsymbol{p}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*) \cdot \boldsymbol{z} > 0 \tag{C.6}$$

Literaturverzeichnis

[1] Hemker, Thomas ; von Stryk, Oskar: Materialien zur Vorlesung Optimierung statischer und dynamischer Systeme. (2010)

[2] Grass, Dieter ; Caulkins, Jonathan P. ; Feichtinger, Gustav ; Tragler, Gernot ; Behrens, Doris A.: *Optimal Control of Nonlinear Processes: With Applications in Drugs, Corruption and Terror*. Springer, 2008

[3] Bryson, Arthur E. ; Yu-Chi, Ho: *Applied Optimal Control: Optimization, Estimation, and Control*. John Wiley & Sons Inc, 1979

[4] Griewank, Andreas ; Walther, Andrea: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. SIAM, 2008

[5] Walther, Andrea ; Griewank, Andreas: *ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++*. ver. 2.1.12, 2011

[6] von Stryk, O.: *Numerische Lösung optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung der adjungierten Variablen.* Düsseldorf, Diss., 1994

[7] Becerra, Victor M.: *PSOPT Optimal Control Solver User Manual. Release 2*, 2010. – Available: http://code.google.com/p/psopt/downloads/list

[8] Flury, Walter: *Skript zur Vorlesung Raumfahrtmechanik*. 2006

[9] Betts, John T.: *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, 2001

[10] Mingotti, G. ; Topputo, F. ; Bernelli-Zazzera, F.: Dynamical Systems, Optimal Control and Space Trajectory Design. In: *CelMec V* (2009)

[11] *International Space Station Zero-Propellant Maneuver (ZPM) Demonstration*. `http://www.nasa.gov/mission_pages/station/research/experiments/ZPM.html`

[12] Bhatt, Sagar A.: *Optimal Reorientation of Spacecraft Using Only Control Moment Gyroscopes*, Rice University, Diplomarbeit, 2007

[13] Bedrossian, Nazareth ; Bhatt, Sagar ; Lammers, Mike ; Nguyen, Louis: Zero Propellant Maneuver Flight Results for 180 ISS Rotation. (2008)

[14] Kang, Wei ; Bedrossian, Nazareth: Pseudospectral Optimal Control Theory Makes Debut Flight, Saves NASA 1M in Under Three Hours. In: *SIAM News* vol. 40 num. 7 (2007)

[15] Ross, I.M.: *User's Manual for DIDO: A MATLAB Application Package for Solving Optimal Control Problems*, 2003

[16] von Stryk, Oskar: *User's Guide for DIRCOL A Direct Collocation Method for Numerical Solution of Optimal Control Problems*, 2002

[17] Heim, Alexander ; von Stryk, Oskar: *Documentation of PAREST - A Multiple Shooting Code For Optimization Problems in Differential-Algebraic Equations*, 1996

[18] VON STRYK, O.: Numerische Verfahren zur Parameteridentifikation in Mehrkörpersystemen / Manuskript zum Lehrgang TV 1.03 der Carl-Cranz-Gesellschaft e.V., Weßling-Oberpfaffenhofen. 1999. – Forschungsbericht. – 64 pages

[19] BREITNER, M. H. ; KOSLIK, B. ; VON STRYK, O. ; PESCH, H. J.: Optimal control of investment, level of employment and stockkeeping. In: BACHEM, A. (Hrsg.) ; DERIGS, U. (Hrsg.) ; JÜNGER, M. (Hrsg.) ; SCHRADER, R. (Hrsg.): *Operations Research "93*, Heidelberg: Physica Verlag, 1993, S. 60–63

[20] HEIM, A. ; VON STRYK, O. ; PESCH, H.J. ; SCHÄFFLER, H. ; SCHEUER, K.: Parameteridentifikation, Bahnoptimierung und Echtzeitsteuerung von Robotern in der industriellen Anwendung. In: HOFFMANN, K.-H. (Hrsg.) ; LOHMANN, T. (Hrsg.) ; JÄGER, W. (Hrsg.) ; SCHUNCK, H. (Hrsg.): *Mathematik - Schlüsseltechnologie für die Zukunft*, Springer, 1997, 551-564

[21] VON STRYK, O. ; SCHLEMMER, M.: Optimal control of the industrial robot Manutec r3. In: BULIRSCH, R. (Hrsg.) ; KRAFT, D. (Hrsg.): *Computational Optimal Control, International Series of Numerical Mathematics* Bd. 115, Basel: Birkhäuser, 1994, S. 367–382

[22] FRANKE, J. ; OTTER, M.: The manutec r3 benchmark models for the dynamic simulation of robots / Institute for Robotics and System Dynamics, DLR Oberpfaffenhofen. 1993. – Forschungsbericht

[23] OTTER, M. ; TURK, S.: The dfvrl models 1 and 2 of the manutec r3 robot / Institute for Robotics and System Dynamics, DLR Oberpfaffenhofen. 1987. – Forschungsbericht

[24] REINL, Christian: *Trajektorien- und Aufgabenplanung kooperierender Fahrzeuge: Diskret-kontinuierliche Modellierung und Optimierung*, Technische Universität Darmstadt, Diss., Mar 25 2010. http://tuprints.ulb.tu-darmstadt.de/2285

[25] SHILLER, Zvi ; DUBOWSKY, Steven: On Computing the Global Time-Optimal Motions of Robotic Manipulators in the Presence of Obstacles. In: *IEEE Transactions on Robotics and Automation* vol. 7 (1991), S. 786–797

[26] BOBROW, J. E. ; DUBOWSKY, S. ; GIBSON, J. S.: Time-Optimal Control of Robotic Manipulators Along Specified Paths. In: *The International Journal of Robotics Research* vol. 4 (1985), S. 3–17

[27] SLOTINE, Jean-Jacques E. ; YANG, Hyun S.: Improving the Efficiency of Time-Optimal Path-Following Algorithms. In: *IEEE Transactions on Robotics and Automation* vol. 5 (1989), S. 118–124

[28] SHILLER, Zvi: Time-Energy Optimal Control of Articulated Systems woth Geometric Path Constraints. In: *IEEE International Conference on Robotics and Automation* (1994)

[29] VERSCHEURE, Diederik ; DEMEULENAERE, Bram ; SWEVERS, Jan ; DE SCHUTTER, Joris ; DIEHL, Moritz: Time-Optimal Path Tracking for Robots: a Convex Optimization Approach. In: *IEEE Transactions on Automatic Control* (2008). – Accepted

[30] BOYD, S. ; VANDENBERGHE, L.: *Convex Optimization*. Cambridge University Press, 2004

[31] SARKAR, Pritam K. ; YAMAMOTO, Motoji ; MOHRI, Akira: A Numerical Method to Minimize Tracking Error of Multi-Link Elastic Robot. In: *Proceedings of IEEE Conference on Intelligent Robots and Systems* (1998)

[32] ABELE, E. ; BAUER, J. ; FRIEDMANN, M. ; PISCHAN, M. ; REINL, C. ; VON STRYK, O.: Einsatz von Robotern in der spanenden Fertigung. In: *Wissenschaftsmagazin forschen* 1/2011 (2011), S. 44 – 49

[33] Abele, Eberhard: *Gussputzen mit sensorgeführten Handhabungssystemen*, Technische Universität Stuttgart, Diss., 1983

[34] Schauer, U.: *Qualitätsorientierte Feinbearbeitung mit IR*, TH Karlsruhe, Diss., 1995

[35] Ammann, B.: Robotergerechte Entgratwerkzeuge und diverse Praxisbeispiele. In: *3. Workshop Bearbeiten mit Industrierobotern, IPA Stuttgart* (2007)

[36] Robertsson, A. ; Brogardh, T. ; et. al.: Implementation of Industrial Robot Force Control - Case Study: High Power Stub Grinding and Deburring. In: *Proceedings of IEEE Conference on Intelligent Robots and Systems* (2006)

[37] Ren, X. ; Kuhlenkötter, B.: Simulation and Verification of belt grinding with Industrial Robots. In: *International Journal of Machine Tools & Manufacture* vol.46 (2006), S. 708–716

[38] Weigold, M.: *Kompensation der statischen Werkzeugabdrängung bei der spanenden Bearbeitung mit Industrierobotern*, TU Darmstadt, Diss., 2008

[39] Liepert, B.: Vom Handlanger zur teamfähigen Fachkraft- Industrieroboter in der Verarbeitungsindustrie. In: *KUKA Roboter GmbH* (2008)

[40] Anderl, Reiner: *Virtuelle Produktentwicklung A, CAD-Systeme und CAx-Prozessketten, Skript zur Vorlesung im WS 2009/2010*

[41] Anderl, Reiner ; Gilsdorf, Erik: *Einführung in das rechnergestützte Konstruieren, Skript zu Vorlesung im Sommersemester 2008*

[42] Anderl, Reiner: *Grundlagen des CAE/CAD, Skript zur Vorlesung im Sommersemester 2009*

[43] van Luttervelt, C.A. ; Childs, T.H.C. ; Jawahir, I.S. ; Klocke, F. ; Venuvinod, P.K. ; Altintas, Y. ; Armarego, E. ; Dornfeld, D. ; Grabec, I. ; Leopold, J. ; Lindstrom, B. ; Lucca, D. ; Obikawa, T. ; Shirakashi ; Sato, H.: Present Situation and Future Trends in Modelling of Machining Operations Progress Report of the CIRP Working Group 'Modelling of Machining Operations'. In: *CIRP Annals - Manufacturing Technology* 47 (1998), Nr. 2, 587 - 626. http://dx.doi.org/10.1016/S0007-8506(07)63244-2. – DOI 10.1016/S0007–8506(07)63244–2. – ISSN 0007–8506

[44] van Luttervelt, K.: Modeling of Machining Operations, Final Report on the Activities of the CIRP Working Group. 2002. – Forschungsbericht

[45] Merchant, M. E.: Mechanics of the Metal Cutting Process. I. Orthogonal Cutting and a Type 2 Chip. In: *Journal of Applied Physics* vol. 16 (1945), S. 318–324

[46] Molinari, A. ; Dudzinski, D.: Stationary shear band in high-speed machining. In: *C.R. Acad. Sci. Paris* 315 (1992), Nr. II, S. 399–405

[47] Lee, M. ; Flom, D. G.: Metallurgical Aspects of the Chip Formation, Process at Very High Cutting Speed. In: *Proceedings of the 9th North American Manufacturing Research Conference* (1951), S. 326–333

[48] Tönshoff, H. K.: *Spanen - Grundlagen*. Springer Verlag, 1995

[49] Victor, H. ; Kienzle, O.: Die Bestimmung von Kräften und Leistungen an spanenden Werkzeugmaschinen. In: *VDI-Z* vol.94 (1952), S. 299–305

[50] Altintas, Y.: Modeling Approaches and Software for Predicting the Performance of Milling Operations at MAL-UBC. In: *International Journal of Machining Science and Technology* vol. 4/3 (2000), S. pp. 445–478

[51] Fassen, Ronald: *Chatter Prediction and Control for High-Speed Milling: Modelling and Experiments*, TU Eindhoven, Diss., 2007

[52] Albersmann, Frank: *Simulationsgestützte Prozessoptimierung für die HSC Bearbeitung*, Universität Dortmund, Diss., 1999

[53] Foley, James D. ; VanDam, Andries ; Feiner, Steven K.: *Computer Graphics: Principles and Practice*. Addinson-Wesley, 1992

[54] Siciliano, Bruno ; Khatib, Oussama: *Handbook of Robotics*. Springer Verlag, 2008

[55] von Stryk, Oskar: *Robotik 1, Skript zur Vorlesung im WS 2010/11*. 2010

[56] KUKA GmbH: Datenblatt zum Industrieroboter KR 210-2.

[57] Craig, John J.: *Introduction to Robotics: Mechanics and Control 2nd ed.* Addinson-Wesley Longman, 1989

[58] Hagedorn, Peter: *Technische Mechanik 3 Dynamik*. Verlag Harri Deutsch, 2006

[59] Gross, Dietmar ; Hauger, Werner ; Schröder, Jörg ; Wall, Wolfgang A.: *Technische Mechanik 1*. Springer, 2006

[60] Baraff, David: An Introduction to Physically Based Modeling. In: *Siggraph Course Notes*, 1997

[61] Meyberg, Kurt ; Vachenauer, Peter: *Höhere Mathematik 2*. Springer, 2003

[62] Press, William H. ; Teukolsky, Saul A. ; Vetterling, William T. ; Flannery, Brian P.: *Numerical Recipes The Art of Scientific Computing*

[63] Stoer, J. ; Burlisch, R.: *Numerische Mathematik 2*. Springer, 2005

[64] Harzheim, Lothar: *Strukturoptimierung Grundlagen und Anwendungen*. Verlag Harri Deutsch, 2008

[65] Kelley, C. T.: *Iterative Methods for Optimization*. SIAM, 1999

[66] Wächter, Andreas: Short Tutorial: Getting Started With Ipopt in 90 Minutes. In: Naumann, Uwe (Hrsg.) ; Schenk, Olaf (Hrsg.) ; Simon, Horst D. (Hrsg.) ; Toledo, Sivan (Hrsg.): *Combinatorial Scientific Computing*. Dagstuhl, Germany : Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009 (Dagstuhl Seminar Proceedings 09061). – ISSN 1862–4405

[67] Gill, Phili E. ; Murray, Walter ; Saunders, Michel A.: *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming**, 2008

[68] Kawajir, Yoshiaki ; Laird, Carl ; Wächter, Andreas: *Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT*, 2010

[69] Corke, P.I.: A Robotics Toolbox for MATLAB. In: *IEEE Robotics and Automation Magazine* 3 (1996), März, Nr. 1, S. 24–32

[70] Walter, Marcelo ; Fournier, Alain: Approximate Arc Length Parametrization. In: *SIBGRAPI* (1996)

[71] Siciliano, Bruno ; Sciavicco, Lorenzo ; Villani, Luigi ; Oriolo, Giuseppe: *Robotics: Modelling, Planning and Control*. Springer Verlag, 2010

Abbildungsverzeichnis