Diskret-kontinuierliche Optimalsteuerung: Modellierung, Numerik und Anwendung bei Mehrfahrzeugsystemen

Am Fachbereich Informatik der
Technischen Universität Darmstadt
eingereichte

Dissertation

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)

von

Dipl.-Math. Markus Glocker
(geboren in München)

Referenten der Arbeit:	Prof. Dr. Oskar von Stryk
	Prof. Dr.-Ing./Univ. Tokio Martin Buss, Technische Universität München
Tag der Einreichung:	18. 08. 2005
Tag der mündlichen Prüfung:	21. 10. 2005

D 17

Darmstadt, 2006

Meinen Eltern!

Inhaltsverzeichnis

Notation

Falls nichts Weiteres angegeben ist, werden in dieser Arbeit die hier genannten Notationen verwendet.

1. *Mengen und deren Elemente*
   Die Menge der natürlichen Zahlen wird mit dem Symbol $\mathbb{N} := \{1, 2, \dots\}$ bezeichnet. Es wird angenommen, dass die Null nicht in dieser Menge enthalten ist. Die natürlichen Zahlen inklusive der Null werden mit $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$ bezeichnet. Als Symbol für die ganzen Zahlen wird $\mathbb{Z}$, für die reellen Zahlen $\mathbb{R}$ verwendet. Für alle weiteren Mengen werden Mengenzeichen, wie z. B. $\mathbb{A}$ oder $\mathbb{B}$ gesetzt.

   Die skalaren Elemente einer Menge werden mit gewöhnlichen Kleinbuchstaben, z. B. $x \in \mathbb{R}$, Spaltenvektoren mit fetten, z. B. $\boldsymbol{x} \in \mathbb{R}^n,\ n \in \mathbb{N}$ und Matrizen mit großen und fetten Buchstaben, z. B. $\boldsymbol{H} \in \mathbb{R}^{n \times m}$ mit $n, m \in \mathbb{N}$, geschrieben.

2. *Funktionen*
   Der Übersichtlichkeit halber wird die Auswertung einer Funktion $f(x(t), u(t), t)$ auch durch $f|_{t_a} := f(x(t_a), u(t_a), t_a)$ abgekürzt.

3. *Ableitungen*
   Für Zeitableitungen wird die gebräuchliche Schreibweise $\dot{x} = \frac{\mathrm{d}}{\mathrm{d}\,x}t$ und $\ddot{x} := \frac{\mathrm{d}^2 x}{\mathrm{d}\,t^2}$ verwendet. Partielle Ableitungen werden teilweise dadurch gekennzeichnet, dass die Variable, nach der abgeleitet wird, als Index angehängt wird, wie z. B. bei $f_x := \frac{\partial f}{\partial x}$ oder $f_{xu} := \frac{\partial^2 f}{\partial x\, \partial u}$.

4. *Indizierung*
   Falls es sich nicht um eine Ableitung handelt, wird der rechte untere Index für die Nummerierung der Variablen verwendet. So hat der Vektor $\boldsymbol{x} \in \mathbb{R}^{n_x}$ die Komponenten $x_1, x_2, \dots, x_{n_x}$. Um eine Anzahl auszudrücken, werden die Buchstaben n und m verwendet, und ihre Indizes drücken aus, welche Anzahl genau gemeint ist.

   Rotationsmatritzen tragen im linken oberen und rechten unteren Index die Koordinatensysteme, zwischen denen sie transformieren. Vektoren tragen in ihrem linken oberen Index das Koordinatensystem, in dem sie dargestellt sind. Ist dieses das Inertialsystem, so wird der Index weggelassen. Demnach gilt ${}^{A}\boldsymbol{r} = {}^{A}\boldsymbol{R}_{B}\,{}^{B}\boldsymbol{r}$.

   Steht im rechten oberen Index ein Minus bzw. Plus, so bezeichnet dies den linksseitigen bzw. rechtsseitigen Grenzwert einer Funktion: $x(t_i^-) := \lim_{t \uparrow t_i} x(t)$ und $x(t_i^+) := \lim_{t \downarrow t_i} x(t)$.

5. *Logische Operatoren*

Tabelle 1: Operatoren

$\wedge,\ \vee,\ \oplus$	logisches UND, ODER und exklusives ODER
$\rightarrow,\ \leftrightarrow$	logische Implikation und Äquivalenz
$\overline{a}$	logische Verneinung

6. *Problemspezifische Bezeichnungen*

Tabelle 2: Bezeichnungen bei diskreter und kontinuierlicher Optimierung

y	kontinuierliche Problemvariable (reell)
z	diskrete Problemvariable (ganzzahlig oder binär)
$\lambda,\ \mu$	Lagrangemultiplikatoren
$\mathbb{R},\ g,\ h$	Zulässiger Bereich, Gleichungs- und Ungleichungsnebenbedingungen
Φ	Kosten- bzw. Zielfunktion

Tabelle 3: Bezeichnungen bei gemischt diskret-kontinuierlicher Optimalsteuerung

$t,\ t_0,\ t_f$	Zeit, Anfangs- und Endzeit
x	Zustandsvariablen
$u,\ w$	kontinuierliche und diskrete Steuerungen
$p,\ q$	kontinuierliche und diskrete Steuerparameter
$g,\ h$	Gleichungs- und Ungleichungsnebenbedingungen
$j,\ r$	Transitions- und Rand- bzw. innere Punktbedingungen
f	rechte Seite einer Zustandsdifferentialgleichung
$J,\ \Phi,\ L$	Zielfunktional, Mayer-Term und Integrand des Lagrangeterms
$H,\ \tilde{H}$	Hamiltonfunktion und erweiterte Hamiltonfunktion

7. *Einheiten*
   Zur Kennzeichnung von Größen wird das *SI*-Einheitensystem (frz.: Le Système international d'unitès) verwendet.

Kapitel 1

Einleitung

Wegen der ständig wachsenden Anforderungen bei der Planung und Erfüllung von technischen Aufgaben in den verschiedensten Bereichen, unter Berücksichtigung von Kosten, Sicherheit, Qualität und Komfort, reicht es nicht aus, nur kontinuierliche oder diskrete Vorgänge unabhängig voneinander zu berücksichtigen, sondern es besteht die Notwendigkeit, diese Bereiche gemeinsam zu betrachten, da sie oftmals stark ineinander verwoben sind und erheblichen Einfluss aufeinander ausüben. Um dies zu bewerkstelligen, müssen die verschiedenen Gebiete der diskreten, kontinuierlichen, dynamischen und globalen Optimierung sowie der optimalen Steuerung zusammengeführt werden.

Die Verbindung dieser Bereiche resultiert in der gemischt diskret-kontinuierlichen Optimalsteuerung, die eine Erweiterung der herkömmlichen Optimalsteuerung um diskrete Parameter und diskretwertige Steuerungen darstellt. Damit ist es möglich, *hybride* dynamische Systeme zu untersuchen, die bisher in keinem der genannten Bereiche für sich behandelt werden konnten. In Zusammenhang mit technischen Aufgabenstellungen werden Systeme als hybrid bezeichnet, die verschiedene Konfigurationen haben und zwischen unterschiedlichen Modi wechseln können, wobei sie innerhalb einer Konfiguration oder eines Modus kontinuierliches Verhalten aufweisen. Diese Konfigurationen und Modi werden häufig mit diskreten Parametern und diskretwertigen Steuerungen modelliert. Um die Komplexität, die durch die diskreten Parameter und Steuerungen hinzukommt, in den Griff zu bekommen, müssen neue theoretische Untersuchungen und neue numerische Verfahren untersucht werden.

Der Herausforderung gemischt diskret-kontinuierlicher Optimalsteuerungsprobleme wurde erst in den letzten Jahren systematisch nachgegangen. Dies liegt zum einen an der fortschreitenden Entwicklung von Verfahren zur Lösung von kontinuierlichen Optimalsteuerungsproblemen, die es erlaubt, komplexe Vorgänge mit vielen Freiheitsgraden zu untersuchen, zum anderen an den neuen Algorithmen der diskreten Optimierung, die Probleme mit Tausenden von Variablen lösen, und nicht zuletzt an der Weiterentwicklung der Computer, die es ermöglicht, die nötigen Berechnungen in akzeptabler Zeit durchzuführen. Dennoch befindet sich die Forschung im Bereich der gemischt diskret-kontinuierlichen Optimalsteuerung noch am Anfang ihrer Entwicklung.

1.1 Anwendungsprobleme

Die folgenden Anwendungen sollen verschiedene Szenarien der Regelungs- und Steuerungstechnik zeigen, in denen gemischt diskret-kontinuierliche Optimalsteuerung eingesetzt werden kann.

Unteraktuierte Roboter

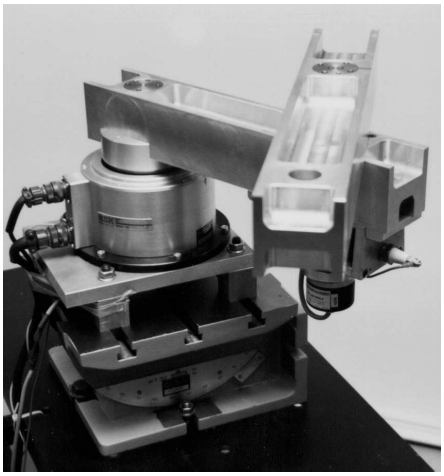


Abbildung 1.1: Unteraktuierter Roboter R2D1 (MBS99)

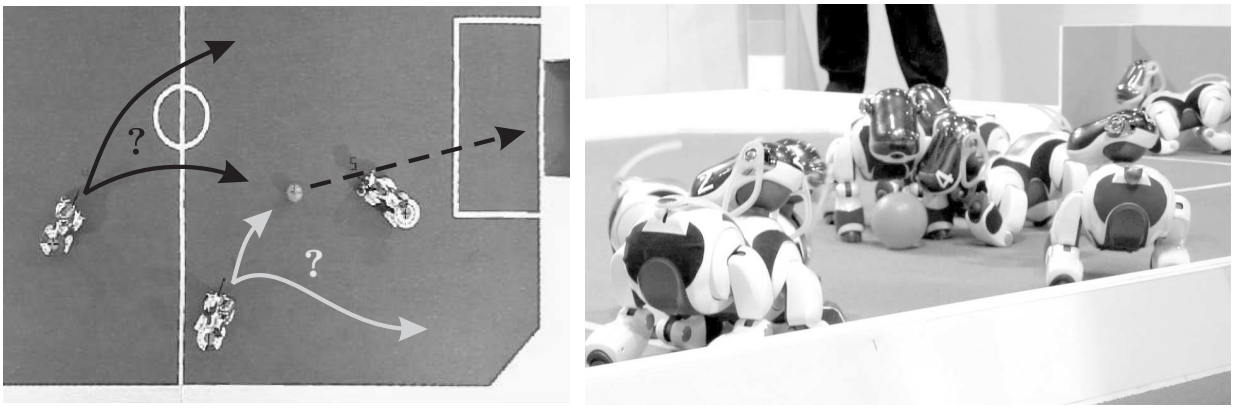
Ein Beispiel für ein hybrides mechanisches System ist der unteraktuierte Roboter R2D1, siehe Abbildung 1.1, der am Lehrstuhl für Steuerungs- und Regelungstechnik der Technischen Universität München entwickelt wurde (MBS98). Bei diesem Roboter handelt es sich um einen planaren Zweiarmroboter (auch SCARA genannt), der mit zwei Drehgelenken ausgestattet ist, wobei das am Montagepunkt liegende Schultergelenk einen zu steuernden Motor besitzt, der den daran befestigten Oberarm und den anschließenden Unterarm bewegt. Im Ellbogengelenk zwischen Ober- und Unterarm ist nur eine Kupplung zum Freigeben oder Fixieren des Unterarms vorhanden. Wird die Kupplung fixiert, sind Ober- und Unterarm starr verbunden, und ein Freiheitsgrad des Systems geht verloren. Dieser Freiheitsgrad muss erneut berücksichtigt werden, sowie die Kupplung freigegeben wird und der Unterarm wieder frei schwingen kann. Es ist unbekannt, wie viele Schaltungen stattfinden. Dafür ist bei nur einer Kupplung die Schaltreihenfolge klar, da nur zwei Zustände existieren, die abwechselnd angenommen werden: Kupplung offen und Kupplung geschlossen. Das Schalten erhält allerdings eine kombinatorische Komplexität, sobald weitere Gelenke mit Kupplungen hinzugefügt werden. Hinzu kommt noch das chaotische Verhalten wie bei einem Doppelpendel, falls mehrere Kupplungen geöffnet sind. Berechnete Lösungen sind damit unter Umständen instabil und können nicht ohne weiteres auf einem realen Roboter eingesetzt werden.

Weitere Gelenke verursachen Kosten bei der Produktion des Roboters, erhöhen aber seine Möglichkeiten bei der Erledigung von Aufgaben. Durch diskrete Parameter ist es auch möglich, die Anzahl der Gelenke, die zur Erledigung einer Aufgabe nötig sind, bei der Optimierung zu berücksichtigen.

Ähnliche Problemstellungen mit Schaltungen sind überall dort zu finden, wo Systeme Schalter besitzen, die im zeitlichen Verlauf an- und abgeschaltet werden können. Dabei kann es sich bei den Systemen um elektronische Bauteile, verfahrenstechnische Anlagen, usw. handeln.

Roboterfußball

Ein Szenario, das Raum für gemischt diskret-kontinuierliche Optimalsteuerung bietet, stellt der RoboCup dar, die Weltmeisterschaft im Roboterfußball. Hier sind verschiedenartige autonome



(a) Problem der Aufgabenverteilung (b) Sony Aibos

Abbildung 1.2: Autonome Roboter beim Fußballspielen

Robotersysteme vertreten, die in verschiedenen Ligen Fußball spielen.

In der Small-Size-Liga spielen kleine, äußerst schnelle, mehrrädrige Roboter. Das Spielgeschehen wird durch eine Deckenkamera verfolgt, deren Bilder von einem externen Rechner ausgewertet werden. Aufgrund dieses globalen Wissens über die Spielsituation erfolgt die Verhaltenssteuerung der einzelnen Roboter. Die Roboter selbst besitzen nur die für die Regelung der Motoren nötige Rechenleistung.

Im Gegensatz dazu sind die Roboter der Middle-Size-Liga voll autonom mit eigenen Kamerasystemen und eigenem Computer. Die Mannschaften bestehen aus radgetriebenen Robotern, die über eine drahtlose Verbindung miteinander kommunizieren können.

Der gleiche Autonomiegrad wie bei der Middle-Size-Liga ist auch in der Vierbeiner-Liga zu finden. Hier kommen ausschließlich die von Sony kommerziell vertriebenen Aibo-Hunde, die in Abbildung 1.2 zu sehen sind, zum Einsatz. Die stark beschränkten Ressourcen an Rechenleistung und die Qualität der Kamerabilder stellen hier neben der vierbeinigen Bewegung die größte Herausforderung dar. Dennoch ist mittlerweile flüssiges Fußballspiel möglich, auch wenn Teamstrategien nur in begrenztem Rahmen verwirklichbar sind.

Als Königsklasse im Roboterfußball kann die Humanoiden-Liga angesehen werden, deren Teilnehmer dem Menschen nachgebildete Roboter sind. Ein flüssiges Spiel kann hier jedoch noch nicht erwartet werden, da stabile zweibeinige Bewegungen, wie sie für das Fußballspiel nötig sind, von autonomen Robotersystemen erst anfänglich bewältigt werden können.

Zur Planung von Spielzügen, zur Optimierung der aktuellen Spielsituation können Aufgaben wie ‚Ball dribbeln‘, ‚Ball schießen‘, ‚Gegner decken‘, usw. den einzelnen Robotern zugewiesen werden. Die Verteilung dieser Aufgaben kann im zeitlichen Verlauf immer wieder wechseln. Bei der Zuweisung sollten nicht nur aktuelle Position und Orientierung der Roboter, sondern auch deren physischen Fähigkeiten, die durch die dynamischen Bewegungseigenschaften gegeben sind, berücksichtigt werden. Auf diese Weise kann eine optimale Strategie im Prinzip als Lösung eines gemischt diskret-kontinuierlichen Optimalsteuerungsproblems gewonnen werden. Diese Lösung kann bei theoretisch gleicher Datenlage jeder im Team für sich berechnen, wodurch ein optimales Zusammenspiel auch ohne Kommunikation möglich ist. Der kontinuierliche Teil ist durch die

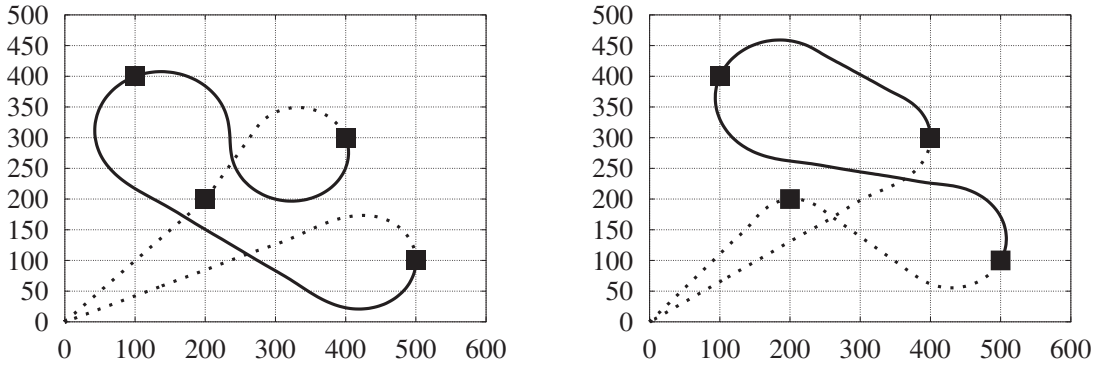


Abbildung 1.3: Optimierung der Reihenfolge von Wegpunkten

Bewegungsdynamik der Roboter gegeben, diskrete Schaltungen stellen die Zuweisung einzelner Aufgaben oder Rollen dar. Die kombinatorische Komplexität steigt mit der Anzahl der Spieler, ihrer Rollen und der möglichen Pässe.

Problemstellungen dieser Art lassen sich überall dort finden, wo mehrere mobile Systeme kooperieren müssen.

Rundreiseprobleme von Flugzeugen und Robotern

Betrachtet wird ein Gebiet, in dem es häufig zu Waldbränden kommt. Durch Satellitenfotos oder andere Luftaufnahmen aus großer Höhe können diese Regionen überwacht werden. Die Auswertung der Aufnahmen liefert mögliche Brandherde, die weiter untersucht werden müssen, um Fehleinsätze zu vermeiden. Mit diesem Ziel werden Wegpunkte für autonome Flugzeuge festgelegt, an denen Nahaufnahmen gemacht werden sollen, um einen Verdacht zu verwerfen oder zu bestätigen. Für einen erfolgreichen Einsatz der Löscheinheiten ist eine möglichst schnelle und exakte Auswertung notwendig. Ein solches Szenario wird z. B. in (MCDO05) betrachtet. Hier wird die Zusammenarbeit verschiedener Flugsysteme untersucht.

Falls mehrere mögliche Brandherde vorliegen, kann die Reihenfolge, in der die Wegpunkte besucht werden, entscheidenden Einfluss auf die Gesamtzeit bis zum Löscheinsatz haben. Die Reihenfolge der anzufliegenden Punkte, von denen aus die Aufnahmen geschossen werden sollen, ist eng an die Flugbahn gekoppelt, da es sich bei dieser um eine glatte Kurve handelt, die durch die individuellen, flugmechanischen Eigenschaften bestimmt wird.

Wie in Abbildung 1.3 dargestellt, können aufgrund der Glattheit der Kurven Flugbahnen zwischen je zwei Wegpunkten nicht unabhängig vom Restverlauf des Fluges betrachtet werden.

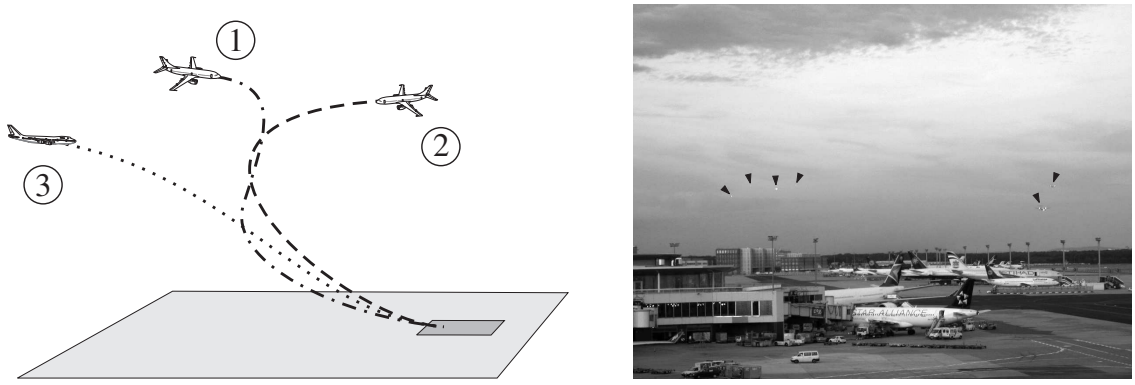
Die Minimierung der Flugzeit unter Berücksichtigung der Reihenfolge der Wegpunkte, der Aerod- und Bewegungsynamik der Flugzeuge und der sich daraus ergebenden Flugbahnen führt auf ein gemischt diskret-kontinuierliches Optimalsteuerungsproblem.

Aufgaben in automatisierten Fertigungsstraßen, wie das Setzen von Schweißpunkten in der Automobilfertigung, führen auf die gleiche Problemstellung wie die Berechnung von Flugbahnen mit

optimaler Reihung der Wegpunkte. In diesem Fall ist eine optimale Reihenfolge der Schweißpunkte ebenso wünschenswert, wie eine optimale Ansteuerung der Gelenke des Roboters. Verharrt der Manipulator während des Schweißens, so können die Bahnen zwischen je zwei Schweißpunkten unabhängig von der restlichen Aufgabe berechnet und schließlich optimal gereiht werden. Bei einem Roboter mit redundanten Freiheitsgraden besteht allerdings die Möglichkeit, dass er sich während des Setzens eines Schweißpunktes weiterbewegt. Zur optimalen Nutzung dieser Freiheit muss das Problem der Steuerung und Reihenfolge geschlossen betrachtet und gelöst werden, da nun die Bahn nach einem Schweißpunkt von der Form der vorherigen Bahn beeinflusst wird.

In beiden Szenarien besteht zusätzlich das Problem der Kollisionsvermeidung, falls mehr als nur ein Robotersystem zum Einsatz kommt.

Einreihen an Verkehrsengpässen



(a) Problem der optimalen Landereihenfolge (b) Flugzeuge im Landeanflug auf Frankfurt

Abbildung 1.4: Landeanflug von Flugzeugen

Die Kapazitäten der Flughäfen in Ballungszentren sind meist ausgelastet. Einen wesentlich beschränkenden Faktor stellen hier die Mindestabstände dar, die die Flugzeuge beim Landeanflug einhalten müssen. Diese Mindestabstände sind aufgrund der Luftverwirbelungen, eines natürlichen Resultats des aerodynamischen Auftriebs, gegeben. Unterschiedliche Flugzeugtypen verursachen mehr oder weniger starke Verwirbelungen. Es hängt auch vom Flugzeugtyp ab, wie gross der Einfluss bestehender Verwirbelungen auf die Flugeigenschaften ist.

Bei einer Landung müssen strenge Sicherheitsrichtlinien beachtet werden, wobei unterschiedliche Interessengruppen verschiedene Optimierungsziele verfolgen. Im Interesse des Flughafenbetreibers liegt eine möglichst hohe Auslastung, während die Fluglinien an einer möglichst genauen Einhaltung der Ankunftszeiten und die Anwohner z. B. an geringer Lärmbelastung interessiert sind.

Um all diesen Kriterien Rechnung zu tragen, ist es notwendig, geeignete Landetrajektorien für die Flugzeuge zu bestimmen. Die Trajektorien hängen von der Landereihenfolge ab, die wiederum von den aerodynamischen Eigenschaften der einzelnen Flugzeuge beeinflusst wird. Die gemeinsame Berechnung der optimalen Reihenfolge und der Trajektorien stellt ein gemischt diskret-kontinuierliches Optimalsteuerungsproblem dar.

1.2 Stand der Forschung und Literaturüberblick

Anwendungen

Im Bereich der Verfahrens- und Prozesstechnik treten viele gemischt diskret-kontinuierliche Fragestellungen auf. In (Gro04) werden verschiedene nichtlineare diskret-kontinuierliche Optimierungsprobleme bei der Vorgangskettenintegration betrachtet. Die Fragestellungen reichen von der Planung bei Umrüstvorgängen bis hin zu Batchdestillationsproblemen. Diese werden auch in (OMHL03) untersucht, wobei hier Optimalsteuerungsaufgaben betrachtet werden.

Auch im Bereich der Steuerung von Fahrzeugen wird hybride Optimalsteuerung untersucht. In (BG03) wird die Schaltung eines Motorrads mit drei halbautomatischen Gängen als zeitdiskretes, lineares Problem untersucht. In (Ger05) wird ebenfalls eine Gangschaltung optimiert, hier allerdings für ein Auto bei der Ausführung eines doppelten Spurwechsels. Für die Berechnungen wird ein nichtlineares Einspurmodell, ein vereinfachtes Rechenmodell des Eigenlenkverhaltens des Fahrzeugs, verwendet.

Im Bereich der Robotik werden häufig unteraktuierte Roboter (MBS98; BSBS00; BHS02; AB03; WL04) untersucht. In diesen Arbeiten jeweils nur mit einem aktuierten und einem geschalteten Gelenk. In (BHS02) wird auch eine Mehrarmtransportaufgabe untersucht. Hier besteht die Aufgabe darin, ein Objekt von einem Roboterarm zum anderen weiterzugeben, wobei jeweils ein oder zwei Arme das Objekt greifen können und Kontakt zum Boden haben kann. Ein ähnliches Problem mit verschiedenen Kontaktsituationen entsteht bei mehrbeinigem Laufen. Je nach Laufmuster haben unterschiedlich viele Beine gleichzeitig Kontakt mit dem Boden. Untersuchungen bei vorgegebenen Laufmustern können in (HS00) gefunden werden.

Auch für mobile Roboter oder unbemannte Flugzeuge werden hybride Fragestellungen untersucht. Dies beinhaltet Aufgabenverteilung bei kooperativem Verhalten aber auch Kollisionsvermeidung zwischen den einzelnen Objekten (ED05; ED04).

Die Untersuchungen kombinatorischer Probleme wie Rundreiseprobleme oder Reihenfolgeoptimierung bei gleichzeitiger Optimierung von Trajektorien sind dem Autor nicht bekannt.

Modellierung hybrider Systeme

Um zu einer Regelung oder Steuerung für ein gemischt diskret-kontinuierliches, zeitveränderliches dynamisches System zu gelangen, muss die Problemstellung zunächst geeignet in verschiedenen Abstraktionsebenen modelliert und formuliert werden.

Bei der Modellierung und Verifikation kommen häufig Werkzeuge zum Einsatz, die aus der Graphen- und Automatentheorie bekannt sind. Diese Werkzeuge sind in den letzten Jahren in Hinblick auf die gemischt diskret-kontinuierlichen zeitveränderlichen Systeme, speziell auf die Klasse der hybriden geschalteten Probleme modifiziert worden. Weit verbreitet ist die Verwendung von *hybriden Automaten* und deren Varianten (ACHH92; GB98; Hen00; BL02; JLS03; AG04), die als

Verallgemeinerung von *zeitlich abgestimmten Automaten* (engl. *timed automata* (Alu99)) betrachtet werden können. Kontinuierliche Vorgänge finden hier in den Knoten statt, diskrete Übergänge in den Kanten. Knoten und Kanten sind mit Gleichungen, die das System beschreiben, markiert. Diese Gleichungen sind meist identisch mit denen, die zur Formulierung eines Optimalsteuerungsproblems verwendet werden. Hybride Automaten geben die Struktur der Phasen eines Optimalsteuerungsproblems gut wieder, weswegen sie zur Modellierung gemischt-ganzzahliger Optimalsteuerungsprobleme geeignet sind.

Eine weitere Möglichkeit der Modellierung bieten Petrinetze (MOE98; BSN$^+$99). Die Theorie dieser Netze geht zurück auf die Dissertation von Carl Adam Petri, 1962. Klassische Petrinetze sind bipartite Graphen, also Graphen, deren Knoten in zwei Teilmengen, hier Plätze und Transitionen, unterteilt werden können, wobei innerhalb der Teilmengen keine Kanten vorhanden sind. Plätze modellieren Bedingungen oder Zustände, Transitionen die Ereignisse. Kanten gehen von Plätzen (Vorbedingungen) zu Transitionen und von Transitionen wiederum zu Plätzen (Nachbedingungen). Die Kanten können dabei noch mit Gewichten assoziiert werden. Der aktuelle Zustand des Netzes ist durch Marken in den Plätzen gegeben. Sind alle Vorbedingungen durch ausreichend viele Marken besetzt, so findet eine Transition statt. Die Marken der Vorbedingungen verschwinden, die der Nachbedingungen werden gesetzt. Zu einem Petrinetz kann ein Erreichbarkeitsgraph assoziiert werden, der zur Verifikation verwendet werden kann. Petrinetze sind gut geeignet, um auch nebenläufige Geschehnisse zu modellieren. Allerdings spiegeln Petrinetze nicht unmittelbar die Struktur von gemischt-ganzzahligen Optimalsteuerungsproblemen wieder. Diese findet sich erst im Erreichbarkeitsgraphen.

Bondgraphen wurden in (Mos97) für die Modellierung von hybriden Systemen weiterentwickelt. Sie werden als gerichteter Graph dargestellt, dessen Knoten Bauelemente eines Systems sind und dessen Kanten (Bonds) gerichtete Energieflüsse. Einige Bauelemente dienen als Energiespeicher, wie Kapazitäten für potentielle Energie (Feder, Kondensator) und Trägheiten für kinetische Energie (Massenträgheit, Induktivität), andere als Übertragungselemente, die die Wirkung von Hebeln, Zahnrädern, Transformatoren für Wechselspannung oder auch Kreiseleffekten darstellen. Weitere Knoten beschreiben die Verknüpfungen der Energieflüsse, also Gleichungen für die den Energiefluss beschreibenden Größen. Die Kanten des Graphen legen neben der Flussrichtung durch eine weitere Marke auch die kausalen Zusammenhänge für die Größen, die den Energiefluss erzeugen, fest. Die Modellierung mit Bondgraphen erfolgt sehr nah an einzelnen Bauteilen des betrachteten Systems, weswegen Bondgraphen für abstraktere Betrachtungen als ungeeignet erscheinen.

Die mathematischen Problemformulierungen, die sich aus den bisher erwähnten Modellierungsansätzen ergeben, haben nicht in erster Linie die Zielsetzung der numerischen Optimierung einer Steuerung. Hier müssen Formulierungen gefunden werden, die zum einen vom gewünschten Optimierungsverfahren verwendet und zum anderen im Sinne der numerischen Optimierung als gutartig betrachtet werden können. Ein Ansatz, der das Problem in Hinblick auf Fragestellungen der Optimierung formuliert, ist die (generalisierte) disjunktive Programmierung (generalized disjunctive programing, GDP) (Gro04; GL03; Gro02; LG01), bei der boolesche Variablen verwendet werden, und das Problem verschiedener möglicher Systemzustände durch Disjunktionen modelliert wird.

Zeitdiskrete geschaltete und hybride Systeme werden in (BM99) in einen gemischt logisch dynamischen (mixed logic dynamic, MLD) (BM99) Rahmen gestellt bzw. äquivalent dazu als stückweise affine Systeme beschrieben (Bem04). Darauf basierend werden in (BBM00) online Optimalsteuerungsprobleme gelöst, welche die Minimierung gewichteter Fehler bei einer Pfadverfolgung über einen endlichen Horizont zur modellbasierten Regelung beinhalten. Die kürzlich entwickelte Beschreibungssprache HYSDEL (TB04) erlaubt es, eine Klasse diskreter hybrider Automaten zu formulieren, die anschließend automatisch in gemischt logische dynamische Systeme übersetzt werden. Die schrittweise Herangehensweise von hybriden Automaten über mathematische Modellierung zu Problemformulierungen für die numerische Lösung erscheint am sinnvollsten, daher wird das Konzept auch in dieser Arbeit verfolgt.

Automaten, Graphen und Netze bieten Werkzeuge, um hybride Systeme darzustellen und zu verifizieren. Hier besteht insbesondere noch Forschungsbedarf bei der Verifikation nichtlinearer Systeme und Systemen mit unendlich vielen diskreten Zuständen. Zur Verifikation wird die Struktur eines Automaten in modale Logik übersetzt, die als Gleichungs- und Ungleichungssysteme für binäre Variablen dargestellt werden kann. Diese Gleichungs- und Ungleichungssysteme werden in dieser Arbeit als Nebenbedingungen für ein gemischt binär-kontinuierliches Optimalsteuerungsproblem genutzt.

Optimierung diskret-kontinuierlicher Systeme

Die Verfahren zur Optimierung von gemischt diskret-kontinuierlichen Optimalsteuerungsproblemen lassen sich zunächst in zwei Gruppen einteilen. Zum einen Verfahren, die erst das Problem durch Diskretisierung in ein gemischt-ganzzahliges Optimierungsproblem umwandeln, das anschließend gelöst wird, und zum anderen solche, die das gemischt diskret-kontinuierliche Optimalsteuerungsproblem direkt angehen.

Verfahren, die auf der Umwandlung in ein gemischt-ganzzahliges Optimierungsproblem basieren, können nochmals untergliedert werden in jene, die totale Diskretisierung, also simultane Steuer- und Zustandsdiskretisierung propagieren, und solche, die nur die Steuerungen diskretisieren.

Auf totaler Diskretisierung basierende Ansätze

Durch die totale Diskretisierung des Problems, z. B. durch Kollokation (Glo00), ergibt sich ein im Allgemeinen nichtlineares gemischt diskret-kontinuierliches Optimierungsproblem, wodurch eine ganze Reihe von Verfahren für diese Problemklasse direkt angewandt werden können. Auch bei zeitdiskreten Systemen, bei denen sich eine Diskretisierung erübrigt, müssen nichtlineare, in Spezialfällen lineare (BBM00; CM01), gemischt diskret-kontinuierliche Optimierungsprobleme gelöst werden. Einen guten Überblick über Verfahren zur gemischt diskret-kontinuierlichen Optimierung geben die Artikel (Gro02; ASF99; BP03; Flo02), sowie die Bücher (Flo95; Kal02). Zur Optimierung von gemischt diskreten Optimierungsproblemen werden im Wechsel verschiedene Teilprobleme gelöst. Die Verfahren werden je nach Art der Teilprobleme klassifiziert (Gro02).

Bei der äußeren Approximation (engl. outer approximation) (VG90; FL94; KAGB04) abwechselnd nichtlineare Teilprobleme zu fixierten diskreten Variablen und ein linearisiertes gemischtes Hauptproblem, zusammengesetzt aus unterstützenden Hyperebenen, gelöst. Die Lösung des ersten Problems liefert eine obere Schranke, die des zuletzt genannten eine globale untere Schranke für die Kosten des Problems. In dem Lösungspunkt eines nichtlinearen Teilproblems wird dann die Linearisierung des Hauptproblems verfeinert. Der Algorithmus konvergiert, falls die untere Schranke größer der kleinsten oberen Schrabnke ist.

Bei der generalisierten Benders Dekomposition (Generalized Benders Decomposition) handelt es sich um eine Erweiterung der Benders Dekomposition (Geo72). Sie ist ähnlich aufgebaut wie eine äußere Approximation, reduziert jedoch das Hauptproblem unter Ausnutzung der Kuhn-Tucker-Bedingungen und unter Elimination der kontinuierlichen Variablen. Die dadurch errechneten unteren Schranken sind allerdings nicht so streng, d. h. hoch, wie die der äußeren Approximation (Gro02).

Erweiterte Schnittebenenverfahren (WP95) benötigen keine Lösung von nichtlinearen Teilproblemen, sondern verwenden nur die der Linearisierungen. In jedem Iterationspunkt wird eine Linearisierung der am stärksten verletzten Nebenbedingung um eben diesen Punkt hinzugefügt und somit dieser unzulässige Teil des Suchgebiets abgeschnitten. Konvergenz ist erreicht, falls alle Nebenbedingungen im Rahmen der gewünschten Genauigkeit erfüllt sind.

Sehr verbreitet sind *Branch-and-Bound*-Verfahren (BMM99; Ley01; GL02). Hier wird für den diskreten Problemteil ein Suchbaum aufgestellt, bei dem in jedem Knoten eine globale untere Schranke für die dort repräsentierte Teilmenge von Lösungskandidaten berechnet wird. Durch Vergleich dieser unteren Schranke mit einer oberen Schranke kann entschieden werden, ob sich die gesuchte Lösung in der Teilmenge befindet oder nicht. Die Teilmengen, in denen eine Lösung vermutet wird, werden weiter aufgespalten und unterschätzt, bis die gesuchte Lösung gefunden ist.

Alle bisher erwähnten Verfahren gehen davon aus, dass es sich um ein konvexes Problem handelt, d. h. alle nichtlinearen Funktionen bezüglich der kontinuierlichen und diskreten Parameter konvex sind. In diesem Fall finden sie, falls vorhanden, die globale Lösung des Problems. Im nichtkonvexen Fall muss für die Lösung der kontinuierlichen Teilprobleme auf globale Optimierungsverfahren zurückgegriffen werden. Die Varianten der dadurch entstehenden Verfahren unterscheiden sich meist nur durch die Art und Weise, wie die eben beschriebenen Verfahren mit den globalen Suchverfahren gekoppelt sind.

Globale Optimierungsverfahren basieren für gewöhnlich auf räumlichen Branch-and-Bound-Verfahren. Räumlich meint hier, dass der Raum der zulässigen kontinuierlichen Variablen successive unterteil wird. Zu jeden Teilbereich wird eine untere Schranke für die Kosten berechnet, wobei nichtkonvexe Terme konvex unterschätzt werden (ADFN98a; ADFN98b; AAF00). Der größte konvexe Unterschätzer ist dabei die konvexe Hülle. Teilbereiche deren untere Schranke größer einer oberen ist, werden von der weiteren Suche ausgeschlossen. Hervorzuheben ist hier das α Branch-and-Bound-Verfahren. Eine Grundvariante wird in (ADFN98a) beschrieben. Alle im Problem vorkommenden Funktionen werden in Terme mit speziellen mathematischen Eigenschaften, wie z. B. lineare, bilineare, trilineare, gebrochen bilineare, gebrochen trilineare, eindimensionale konkave und allgemein nichtkonvexe Terme, aufgespalten, um diese separat zu behandeln, und

damit möglichst gute konvexe Relaxationen zur Schrankengenerierung zu bekommen. Bei der Unterschätzung der allgemein nichtlinearen Terme durch quadratische Funktionen wird ein Shiftparameter α benötigt, der sich aus den zweiten Ableitungen berechnen lässt und der dem Verfahren seinen Namen gibt.

Ansätze die auf totaler Diskretisierung basieren haben den Vorteil, dass die Verfahren, die aus der gemischt-ganzzahligen Optimierung bekannt sind, direkt angewandt werden können. Allerdings können bei der Diskretisierung eines Problems Informationen verloren gehen. Es muss vorab eine für alle Teilprobleme im Suchbaum geeignete Diskretisierung gefunden werden, damit eine optimale Trajektorie berechnet werden kann. Die Diskretisierung, die für ein Teilproblem zuverlässig und schnell gute Lösungen garantiert, kann für ein anderes Teilproblem völlig unzureichend sein. Nach der Diskretisierung ist es nicht mehr möglich flexibel auf den tatsächlichen Systemzustand einzugehen, weswegen die Transformation in ein gemischt-ganzzahliges Optimierungsproblem ungeeignet erscheint und in dieser Arbeit nicht verwendet wird.

Auf sequentieller Diskretisierung basierende Ansätze

Eine Herangehensweise zur Lösung von Optimalsteuerungsproblemen ist die sequentielle Diskretisierung. Hier wird der unendlichdimensionale Raum der Steuerungen durch einen endlichdimensionalen approximiert. Somit können Approximationen der Steuerungen durch Linearkombination von Basisfunktionen dargestellt werden. Als Steuerparameter dienen nun die Faktoren der Linearkombination. Das Optimalsteuerungsproblem wird somit in ein Optimierungsproblem für die Steuerparameter mit eingebetteter Dynamik übergeführt. Dies ist z. B. bei direkten Schießverfahren der Fall. Die numerisch integrierte Dynamik wird als nichtlineare Nebenbedingung aufgefasst, womit es sich wieder um ein nichtlineares Optimierungsproblem handelt, das aber im Allgemeinen nicht glatt ist und sogar unstetig sein kann. Die obigen Herangehensweisen können damit nur für bestimmte Probleme angewandt werden, bzw. es müssen die Unstetigkeitsstellen geeignet behandelt werden.

Unter geeigneten Voraussetzungen kann gezeigt werden, dass ein Optimierungsproblem, das auf die eben beschriebene Weise entsteht, die Voraussetzung der zweifachen Differenzierbarkeit, die für das α Branch-and-Bound-Verfahren notwendig ist, erfüllt. Der Parameter, genannt β, der zur Unterschätzung der aus der Dynamik resultierenden Nebenbedingungen benötigt wird, lässt sich aus den Sensitivitäten zweiter Ordnung berechnen. Das dazugehörige Verfahren heißt β-Branch-and-Bound-Verfahren (EF00). Zur exakten Berechnung des Shiftparameters β muss auch die zu unterschätzende Funktion in expliziter Form vorliegen, was in Problemen mit eingebetteter Dynamik wegen der Abhängigkeit des Zielfunktionals und der Nebenbedingungen der Steuerparameter nur selten der Fall ist. Um dieses Problem zu umgehen, wurden verschiedene Herangehensweisen vorgeschlagen, die entweder auf geeigneter Schätzung von β oder auf der Approximation der Hesse-Matrix beruhen. Zum einen ist eine Schätzung nicht exakt genug, um zu garantieren, dass die Funktionen überall konvex unterschätzt werden, und zum anderen ist die Approximation der Hesse-Matrix zu aufwändig.

Das in (SB04) vorgestellte Verfahren benötigt keine Ableitungen zweiter Ordnung. Basierend dar-

auf, dass ein parameterabhängiger stückweise konvexer Integrand zu einem konvexen Integral führt, werden konvexe Unterschätzer für Probleme mit allgemeinem nichtkonvexen Zielfunktional vom Bolzatyp mit linearer eingebetteter Dynamik konstruiert.

Bei der Herangehensweise über sequentielle Diskretisierung ergeben sich zwar Optimierungsprobleme mit weniger Parametern als bei totaler Diskretisierung, jedoch stellt es eine grössere Herausforderung dar, geeignete Gradientenapproximationen bereitzustellen. Zudem besteht auch hier, wie bei den Ansätzen mit totaler Diskretisierung, das Problem, eine geeignete Diskretisierung für das gemischt diskret-kontinuierliche Optimalsteuerungsproblem zu finden. Aus diesen Gründen erscheint auch die sequentielle Diskretisierung als ungeeignet zur Behandlung von gemischt-ganzzahligen Optimalsteuerungsproblemen.

Ansätze ohne Vorabdiskretisierung

Bisher gibt es noch keine allgemeine Methode, um ein allgemeines nichtlineares gemischt diskret-kontinuierliches Optimalsteuerungsproblem global exakt zu lösen. Die bisher beschriebenen Herangehensweisen diskretisieren zunächst das Problem und befassen sich erst dann mit der Lösung der resultierenden Approximation. Die direkte Optimierung eines gemischt diskret-kontinuierlichen Steuerungsproblems ist in der Literatur nur selten und dann nur für spezielle Problemklassen zu finden. Die Herangehensweise findet dann meist über die Optimalitätsbedingungen statt. Mit Maximumprinzipien für hybride Systeme beschäftigen sich die Arbeiten (Wit66; Sus99; Pic99; RKIZ99; Sus00; XA00; RZK99).

Die Basis dieser Untersuchungen bildet das Pontrjaginsche Maximumprinzip (Gam99) für kontinuierliche Optimalsteuerungsprobleme, das in (Wit66) für zeitkontinuierliche Optimalsteuerungsprobleme mit internen Schaltungen betrachtet wird. Hier wird auch gezeigt, dass die Kozustände an den Schaltzeitpunkten gewisse Sprungbedingungen erfüllen müssen. Auch in (Sei87) werden Probleme ähnlich den intern geschalteten betrachtet, und Bedingungen für die Existenz von optimalen Lösungen von Problemen mit zwei Teilsystemen aufgestellt.

Ein weitgehend allgemeines hybrides Optimalsteuerungsproblem wird in (Pic98) betrachtet, jedoch ist die Analyse und die Existenz von Lösungen nicht ohne weitere Einschränkungen möglich, weswegen die Betrachtungen auf zwei spezielle Klassen eingeschränkt werden, für die eine Variante des Maximumprinzips bewiesen wird.

Verschiedene Versionen des Maximumprinzips für extern geschaltete Systeme werden in (Sus99; Sus00) unter Verwendung verallgemeinerter Ableitungstheorie untersucht. Hier ist auch eine nichtglatte Variante zu finden.

In den Veröffentlichungen (RKIZ99; RZK99) wird das geschaltete dynamische Optimierungsproblem durch Konvexkombination der einzelnen Zustände mit binärwertigen Steuerungen in ein gemischt diskret-kontinuierliches Optimalsteuerungsproblem umgewandelt und das Pontrjaginsche Maximumprinzip direkt angewandt. Im Falle von zeitoptimalen linear quadratischen Problemen werden notwendige Bedingungen zu den Schaltzeitpunkten hergeleitet und weitere für die Stabilität untersucht. Da sich bei geeigneten Voraussetzungen Steuerungen aus dem Maximumprinzip ergeben können, sollte dies in der Modellierung der Probleme berücksichtigt werden.

Ebenfalls auf dem Maximumprinzip basieren die Ansätze in (AA04b; AA04a; AAW05), wo ein Algorithmus zur Berechnung suboptimaler Steuerungen für ausschließlich extern geschaltete Systeme vorgeschlagen wird.

Die Herangehensweisen über das Maximumprinzip eignen sich nur für geschaltete Systeme und sind zurzeit nur auf wenige einfache Problemstellungen anwendbar. Allgemeine gemischt diskret-kontinuierliche Fragestellungen wie Rundreise- oder *Schedulingprobleme* lassen sich damit nicht lösen.

Auch die Hamilton-Jacobi-Bellman-Gleichungen können im hybriden wie im kontinuierlichen Fall durch einen Ansatz mit dynamischer Programmierung hergeleitet werden. Bei den Hamilton-Jacobi-Bellman-Gleichungen handelt es sich um partielle Differentialgleichungen. Eine Version dieser Gleichung für extern geschaltete Systeme ohne Schaltkosten kann in (XA00) gefunden werden. Ein Ansatz, um daraus Lösungen zu berechnen, ist, diese sowohl in Richtung der kontinuierlichen Zustände, sowie in Richtung der Zeit zu diskretisieren, und dann ein Rückwärtssuchverfahren anzuwenden (HR02). Einschränkungen sind, abgesehen von der explodierenden kombinatorischen Komplexität, zyklische Randbedingungen, die nicht ohne weiteres berücksichtigt werden können.

Lösungen für diskret-kontinuierliche Optimalsteuerungsprobleme können auch durch eine Suche im Raum der diskreten Variablen gefunden werden, wobei die Kosten zu einem diskreten Wert durch die Lösung des dazugehörigen kontinuierlichen Problems berechnet werden. Zur Suche sind heuristische Verfahren, wie *genetische Algorithmen* oder Nachbarschaftssuche, denkbar, aber auch deterministische Verfahren, wie das Branch-and-Bound Verfahren. Um aber zu global optimalen Lösungen zu kommen, müssen untere Schranken für Optimalsteuerungsprobleme berechnet werden. Ansätze dazu wurden in (Rub98) untersucht. Basierend auf Arbeiten von Young, die in (Rud92) auf eine Klasse von Optimalsteuerungsproblemen mit Beschränkungen erweitert wurden, wird ein Optimalsteuerungsproblem in einen Maßraum eingebettet. Um dort eine Lösung zu finden, muss ein semi-infinites lineares Programm, das durch Diskretisierung des Maßes aus einem unendlichdimensionalen linearen Programm entsteht, gelöst werden. Durch diesen Ansatz ergibt sich eine näherungsweise globale untere Schranke.

Eine weitere Methode, um globale untere Schranken zu berechnen, beruht auf *Screening-Modellen* (AB97). Zur Konstruktion solcher Screening-Modelle ist aber sehr spezifisches Wissen über das zu lösende Problem notwendig.

Zusammenfassung

Die Vorteile der totalen Diskretisierung liegen darin, dass die, für gemischt ganzzahlige Optimierungsprobleme bekannten, Verfahren ohne weiteres auf das diskretisierte Problem angewandt werden können. Analog gilt dies für die sequenzielle Diskretisierung, auch wenn hier Optimierungsprobleme mit eingebetteter Dynamik entstehen. Beide Ansätze legen zu Beginn ein Diskretisierungsgitter fest und weichen bei der Suche nicht mehr davon ab. Dies kann zu Schwierigkeiten führen, da unter Umständen jeder Modus des untersuchten Systems eine andere Diskretisierung

benötigt. Eine Diskretisierung die für alle Modi geeignet ist, enthält zu viele Gitterpunkte, was zu großem Speicheraufwand und langen Rechenzeiten führt.

Verfahren, die versuchen die Optimalitätsbedingungen zu lösen, sind nur für spezielle Probleme geeignet, da das Aufstellen der notwendigen Bedingungen für realistische Problemstellungen einen zu hohen Aufwand darstellt. Weitere Verfahren, die auf dem Maximumprinzip basieren oder versuchen, die Hamilton -Jacobi-Bellman-Gleichungen zu lösen, sind nur für spezielle Probleme anwendbar. Heuristische Verfahren haben den Nachteil, dass sie keine globalen Lösungen garantieren können.

Forschungsbedarf besteht vor allem bei Verfahren, die gemischt diskret-kontinuierliche Optimalsteuerungsprobleme Lösen, ohne sie vorher durch Diskretisierung in ein Problem einer anderen Klasse umzuwandeln, da bei der Diskretisierung wichtige Informationen verloren gehen.

1.3 Ziele und Struktur dieser Arbeit

Ziel dieser Arbeit ist es, eine allgemeine, anwendbare Herangehensweise zur Modellierung und numerischen Optimierung technisch relevanter gemischt diskret-kontinuierlicher Ingenieursanwendungen bereitzustellen, mit der es möglich ist, geschaltete wie auch allgemeine gemischt diskret-kontinuierliche Fragestellungen zu untersuchen, die in der Literatur bislang so gut wie nicht behandelt werden. Hierbei sollen ein neues Lösungsverfahren entwickelt und neue Fragestellungen gelöst werden.

Um dies zu erreichen, wird in Kapitel 2 anhand exemplarischer Problemstellungen eine allgemeine mathematische Formulierung diskret-kontinuierlicher Optimalsteuerungsprobleme hergeleitet. Anschließend werden aus den Anwendungen heraus wichtige Problemklassen definiert, deren Modellierung gleichartige mathematische Fragestellungen liefern. Nach der Klassifizierung werden theoretische Grundlagen bereitgestellt. Dabei werden in Hinblick auf diskret-kontinuierliche Fragestellungen die notwendigen Bedingungen für dynamische Optimierungsrobleme näher betrachtet, sowie eine Version der Hamilton-Jacobi-Bellman-Gleichung und des Minimumprinzips gegeben.

Bei der Modellierung, die in Kapitel 3 erläutert wird, werden hybride Automaten vorgestellt, die um Steuerungen und Kosten erweitert sind. Dabei kommt ein Konzept zum Einsatz, das unter Verwendung neutraler Systemzustände erlaubt, Probleme mit unbekannter, jedoch begrenzter Anzahl von Schaltungen zu betrachten und effizient zu formulieren. Im zweiten Teil dieses Kapitels werden Methoden angegeben, wie die durch den hybriden Automaten beschriebene Fragestellung in ein, für die numerische Berechnung geeignetes, nichtlineares gemischt binär-kontinuierliches Optimalsteuerungsproblem transfomiert wird.

Die Beschreibung der Lösungsansätze in Kapitel 4 beginnt mit einer intensiven Diskussion von direkten Kollokationsverfahren bei verschiedenen Diskretisierungen zur Lösung von rein kontinuierlichen Optimalsteuerungsproblemen. Da zur Lösung eines gemischt diskret-kontinuierlichen Problems eine große Anzahl dieser rein kontinuierlichen Probleme gelöst werden muss, wird hier

auch insbesondere auf die Möglichkeiten der Verminderung von Rechenzeiten durch Nutzung von Gitterverfeinerung, Skalierung und exakten Sensitivitäten, sowie den Einsatz von Homotopieverfahren wegen der Problematik geeigneter Starttrajektorien eingegangen.

Ein sehr allgemeiner Ansatz, um die möglichen diskreten Problemzustände effizient nach einer optimalen Lösung zu durchsuchen, bietet die Branch-and-Bound-Methode, die im zweiten Teil des dritten Kapitels beschrieben wird. Zunächst wird die Nutzung der Methodik bei gemischt diskret-kontinuierlichen Problemen erörtert. Verschiedene Heuristiken zur Knotenwahl und zur Verzweigung werden vorgeschlagen und auch weitere Aspekte werden untersucht, wie die Generierung erster oberer Schranken und die Verwendung von Homotopien zum Durchlaufen des Baumes.

Bei Problemen mit wenigen kontinuierlichen Übergängen zu Zeitpunkten diskreter Ereignisse kann auch ein neu entwickeltes *Dekompositionsverfahren* eingesetzt werden, das im letzten Teil von Kapitel 4 erläutert wird. Die notwendige Diskretisierung der Zustände und der Aufbau eines Suchgraphen werden hier beschrieben, wie auch die Numerik der linearen und linearen gemischt ganzzahligen Optimierung, die hier zum Einsatz kommt.

Abgerundet wird die Arbeit in Kapitel 5 mit ausgewählten Lösungen zu gemischt diskret-kontinuierlichen Problemstellungen. In einem Beispiel aus dem Luftfahrtmanagement werden erstmalig Trajektorien und Landereihenfolge geschlossen betrachtet und optimiert, um zu einer optimalen Auslastung eines Flughafens zu kommen. Verschiedenen Betrachtungen aus dem Roboterfußball zeigen die Möglichkeit der Optimierung von Teamverhalten und -strategie unter Berücksichtigung der Dynamik der einzelnen Roboter. Hier wird auch gezeigt, wie ein optimales Teamverhalten von den dynamischen Bewegungseigenschaften einzelner Spieler geprägt wird. Zuletzt werden Lösungen des *motorisierten Handlungsreisendenproblems*, eines elementaren Beispiels für Reihenfolgeprobleme bei Industrierobotern und unbemannten Flugkörpern berechnet. Hier wird der Einsatz eines neu entwickelten Dekompositionsansatzes im Vergleich mit einem Branch-and-Bound-Verfahren diskutiert.

Eine Zusammenfassung der wichtigsten Ergebnisse sowie ein Ausblick auf mögliche Weiterentwicklungen bildet in Kapitel 6 den Abschluss dieser Arbeit.

Kapitel 2

Theoretische Grundlagen diskret-kontinuierlicher Optimalsteuerungsprobleme

Zu Beginn dieses Kapitels wird eine allgemeine mathematische Problemstellung aus den Beispielanwendungen der Einleitung heraus entwickelt. Diese Darstellung ist Grundlage für eine Klassifizierung im zweiten Teil dieses Kapitels. Schließlich werden im letzten Teil die notwendigen Bedingungen für optimale Lösungen dynamischer Optimierungsprobleme bereitgestellt.

2.1 Herleitung der Problemstellung

Bei den in der Einleitung genannten Beispielen wird jeweils versucht, optimale Trajektorien und die dazugehörigen Steuerungen zu berechnen. Die Steuerungen bzw. *Steuervariablen* können dabei unterteilt werden in Steuervariablen $\boldsymbol{u} : [t_0, t_f] \rightarrow \mathbb{R}^{n_u}$, die kontinuierlich verändert werden können, und in Steuervariablen $\boldsymbol{w} : [t_0, t_f] \rightarrow \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m_w}\}$, die nur diskrete Werte annehmen dürfen. Zu den kontinuierlichen Systemeingaben gehören z. B. die Motormomente in den kontinuierlich gesteuerten Gelenken eines unteraktuierten Roboters, zu den diskreten z. B. der Zustand seiner Kupplungen.

Durch kontinuierliche und diskrete Parameter kann die Konfiguration des untersuchten Systems beschrieben werden. Dabei werden Längen, Gewichte usw. durch *kontinuierliche Steuerparameter* $\boldsymbol{p} \in \mathbb{R}^{n_p}$ beschrieben. Durch *diskrete Steuerparameter* $\boldsymbol{q} \in \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{m_q}\}$ lassen sich hingegen die Anzahl von Gelenken oder die Reihenfolge der Wegpunkte für Luftaufnahmen beschreiben. Diese Parameter können als Variablen im Optimalsteuerungsproblem vorhanden sein.

Das System selbst wird durch die *Zustandsvariablen* $\boldsymbol{x}(t) : [t_0, t_f] \rightarrow \mathbb{R}^{n_x}$ beschrieben, deren Änderung durch die *Systemdynamik*

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t)$$

gegeben ist. Diese hängt von der Konfiguration des Systems und damit von den Steuerparametern ab und kann im zeitlichen Verlauf aufgrund der Steuerungen $\boldsymbol{u}(t)$ und $\boldsymbol{z}(t)$ variieren. Dies geschieht z. B. durch Betätigung der Kupplung bei unteraktuierten Roboter, wodurch Freiheitsgrade hinzukommen oder verloren gehen. Bei der Betätigung der Kupplung kann das System aufgrund des Stoßes und eventueller Verbiegung Energie verlieren, was sich in einem Sprung bzw. einem nichtglatten Übergang in den Zuständen äußern kann. Solche Sprünge können durch *Transitionsbedingungen*

$$0 = \boldsymbol{j}_i(\boldsymbol{x}(\hat{t}_i^-), \boldsymbol{x}(\hat{t}_i^+), \hat{t}_i)$$

beschrieben werden.

Die Zustände und Steuerungen müssen während des zeitlichen Verlaufs verschiedene Nebenbedingungen einhalten. Sie besitzen untere und obere Schranken oder sind aufgrund der Umgebung eingeschränkt. Bei Flugzeugen muss eine Mindestflughöhe eingehalten werden und auch der Abstand zu anderen Flugzeugen darf nicht zu gering werden. Beim Roboterfußballspiel muss der Abstand zwischen Spieler und Ball klein genug sein, damit er diesen dribbeln kann, jedoch muss der Abstand wieder beliebig sein, sobald der Spieler den Ball abspielt. Um diese Beschränkungen zu formulieren, werden *Gleichungs-* und *Ungleichungsnebenbedingungen*

$$\begin{aligned} 0 &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t) \\ 0 &\geq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t) \end{aligned}$$

benötigt, die von den Zuständen, Steuerungen und Parametern sowie der Zeit abhängen.

Beim Rundreiseproblem für Luftaufnahmen muss das Flugzeug vom Start bis zur Landung zu unbekannten Zeitpunkten t_i an jeweils einem der Wegpunkte sein. Dies ist eine Forderung an die Zustandsvariablen und lässt sich durch *Randbedingungen* und *Gleichungsbedingungen an inneren Punkten*

$$0 = \boldsymbol{r}_i(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}, t_i), \quad i = 0, \ldots, n_c$$

fordern. Die Zeitpunkte $t_i,\ i = 1, \ldots, n_c$ sind damit ebenfalls Variablen des Problems. Die Zeitintervalle $[t_i, t_{i+1}]$ bilden die n_c Phasen des Optimalsteuerungsproblems.

Um logische Zusammenhänge der diskreten Schaltungen und Parameter in die Problemformulierung mit aufzunehmen, werden *lineare Bedingungen*

$$0 \geq \boldsymbol{l}(\boldsymbol{w}(t_0), \ldots, \boldsymbol{w}(t_f), \boldsymbol{q})$$

benötigt. Hier kann z. B. die Forderung beschrieben werden, dass bei einer Rundreise jeder Wegpunkt nur einmal besucht werden darf.

Die Güte einer Trajektorie und der dazugehörigen Steuerung wird schließlich durch ein *Zielfunktional*

$$J[\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{q}] = \sum_{i=0}^{n_c} \Phi_i(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}, t_i) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t) \mathrm{d}\, t$$

gemessen.

Mathematische Problemstellung

Werden die Bedingungen des letzten Abschnitts zusammengefasst, entsteht ein *gemischt diskret-kontinuierliches Optimalsteuerungsproblem*, das die Aufgabe beinhaltet, ein Zielfunktional

$$J[\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{q}] = \sum_{i=0}^{n_c} \Phi_i(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}, t_i) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t)\mathrm{d}\,t \tag{2.1}$$

unter der Berücksichtigung von Nebenbedingungen

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t) && (2.2)\\
0 &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t) && (2.3)\\
0 &\geq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t) && (2.4)\\
0 &= \boldsymbol{r}_i(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}, t_i) && (2.5)\\
0 &= \boldsymbol{j}_i(\boldsymbol{x}(\hat{t}_i^-), \boldsymbol{x}(\hat{t}_i^+), \hat{t}_i) && (2.6)\\
0 &\geq \boldsymbol{l}(\boldsymbol{w}(t_0), \ldots, \boldsymbol{w}(t_f), \boldsymbol{q}) && (2.7)
\end{aligned}$$

zu minimieren. Die Problemvariablen sind die kontinuierlichen Steuervariablen $\boldsymbol{u} : [t_0, t_f] \rightarrow \mathbb{R}^{n_u}$, die diskreten Steuervariablen $\boldsymbol{w} : [t_0, t_f] \rightarrow \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m_w}\}$, die kontinuierlichen Steuerparameter $\boldsymbol{p} \in \mathbb{R}^{n_p}$ und die diskreten Steuerparameter $\boldsymbol{q} \in \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{m_q}\}$. Die Endzeit $t_f \in \mathbb{R}^+$ kann frei oder fest vorgegeben sein. Zu gegebenen Steuerungen $\boldsymbol{u}(t)$ und $\boldsymbol{w}(t)$ sowie zu den Parametern $\boldsymbol{p}$ und $\boldsymbol{q}$ lassen sich aus der Dynamik (2.2) die dazugehörigen Zustandsvariablen $\boldsymbol{x}(t)$ berechnen. Ein *Steuerprozess* $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{q})$ heißt *zulässig*, falls er die Gleichungsnebenbedingungen (2.3), die Ungleichungsnebenbedingungen (2.4), die Gleichungen (2.5) und (2.6) an den Rand- sowie den inneren Punkten t_i und die logischen Bedingungen (2.7) erfüllt. Ein zulässiger Steuerprozess $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{w}^*, \boldsymbol{q}^*)$, der das Funktional (2.1) minimiert, heißt *optimal*.

Das Zielfunktional (2.1) des Optimalsteuerungsproblems wird *Bolza*-Funktional genannt und besteht aus zwei Termen. Der erste davon wird *Mayer*-Term genannt, der zweite *Lagrange*-Term.

Eigenschaften diskret-kontinuierlicher Optimalsteuerungsprobleme

Bemerkung 1.
Das Zielfunktional der Aufgabe kann auch auf reine Mayer-Form gebracht werden, indem eine zusätzliche Variable $x_L(t)$ mit Anfangswert $x_L(t_0) = 0$ und die zusätzliche Differentialgleichung

$$\dot{x}_L = L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), t)$$

eingeführt werden. Der Wert des Integrals entspricht somit dem Wert der Variablen x_L zur Zeit t_f.

Bemerkung 2.
Analog zu Bemerkung 1 kann eine neue Variable $x_G(t)$ verwendet werden, um eine Integralneben-

bedingung

$$\int_{t_0}^{t_f} G(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t)\mathrm{d}\,t = C$$

in eine zusätzliche Differentialgleichung

$$\dot{x}_G = G(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t)$$

umzuwandeln. Dabei muss die neue Variable den Anfangswert $x_G(t_0) = 0$ und den Endwert $x_G(t_f) = C$ besitzen.

Bemerkung 3.
Liegt ein Differentialgleichungssystem höherer Ordnung

$$\boldsymbol{x}^{(n)} = \boldsymbol{f}(\boldsymbol{x}^{(n-1)}(t), \ldots, \dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t)$$

vor, so kann dieses durch zusätzliche Variablen $\boldsymbol{\xi}_i$ in ein Differentialgleichungssystem 1. Ordnung

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{\xi}_1, \\
\dot{\boldsymbol{\xi}}_1 &= \boldsymbol{\xi}_2, \\
&\ldots \\
\dot{\boldsymbol{\xi}}_n &= \boldsymbol{f}(\boldsymbol{\xi}_{n-1}(t), \ldots, \boldsymbol{\xi}_1(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, t),
\end{aligned}$$

transformiert werden.

Bemerkung 4.
In der vorliegenden Formulierung werden Gleichungs- und Ungleichungsnebenbedingungen betrachtet. Das Problem kann aber auf ein äquivalentes Problem, das nur Gleichungsnebenbedingungen besitzt, transformiert werden, da Ungleichungsnebenbedingungen

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}, t) \leq 0$$

durch Einführen von *Schlupffunktionen* $\boldsymbol{z}(t)$ in Differentialgleichungen

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}, t) + \dot{\boldsymbol{z}}^2(t) = 0, \quad \dot{\boldsymbol{z}}^2(t) = (\dot{z}_1(t)^2, \ldots, \dot{z}_{n_h}(t)^2)$$

umgewandelt werden können.

Bemerkung 5.
Aufgabe (2.1) bis (2.5) auf der vorherigen Seite liegt in *nichtautonomer*, d. h. von der Zeit explizit abhängiger Form vor. Um eine *autonome* Form zu erhalten, wird eine zusätzliche Variable x_t mit Anfangswert $x_t(t_0) = 0$ und Dynamik $\dot{x}_t = 1$ verwendet. Der Wert der neuen Variable entspricht zu jedem Zeitpunkt genau der aktuellen Zeit t. Wird nun in der Aufgabe $x_t(t)$ für die Zeit eingesetzt, so taucht t nicht mehr explizit auf.

Bemerkung 6.
Das gegebene Optimalsteuerungsproblem kann auch in ein äquivalentes Problem mit fester Anfangs- und Endzeit transformiert werden. Hierfür werden eine neue Zeitvariable $\tau \in [0, 1]$ und ein

neuer Zustand x_{t_f} mit der Dynamik

$$\frac{dx_{t_f}}{d\tau} = 0$$

eingeführt, aus denen sich die tatsächliche Zeit

$$t = t_0 + (x_{t_f} - t_0)\tau$$

berechnen lässt. Die Endzeit entspricht dann dem Wert des neuen Zustands $x_{t_f} = t_f$. Insgesamt lautet das transformierte Problem:

$$\begin{aligned}
\min \quad & \textstyle\sum_{i=0}^{n_c} \Phi_i(\boldsymbol{x}(\tau_i), \boldsymbol{q}, t_0 + (x_{t_f} - t_0)\tau_i) \\
& + \textstyle\int_0^1 (x_{t_f} - t_0) L(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{w}(\tau), t_0 + (x_{t_f} - t_0)\tau) \mathrm{d}\tau \\
\text{unter} \quad \tfrac{\mathrm{d}\, x_{t_f}}{\mathrm{d}\, \tau} &= 0 \\
\tfrac{\mathrm{d}\, x_i}{\mathrm{d}\, \tau} &= (x_{t_f} - t_0) f_i(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{w}(\tau), \boldsymbol{p}, \boldsymbol{q}, t_0 + (x_{t_f} - t_0)\tau) \\
0 &= \boldsymbol{h}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{w}(\tau), \boldsymbol{p}, \boldsymbol{q}, t_0 + (x_{t_f} - t_0)\tau) \\
0 &= \boldsymbol{g}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{w}(\tau), \boldsymbol{p}, \boldsymbol{q}, t_0 + (x_{t_f} - t_0)\tau) \\
0 &= \boldsymbol{r}_i(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}, t_0 + (x_{t_f} - t_0)\tau_i).
\end{aligned}$$

Im Weiteren wird auf diese äquivalenten Formulierungen zurückgegriffen.

2.2 Klassifizierung hybrider dynamischer Systeme

In der Literatur werden vor allem geschaltete Systeme als Optimalsteuerungsprobleme formuliert und untersucht. Diese repräsentieren einen kleinen, aber äußerst wichtigen Teilbereich diskret-kontinuierlicher Optimalsteuerungsprobleme, die zeitlich abhängige Schaltungen der Dynamik beinhalten. Weitere Problemstellungen entstehen, falls der zu optimierende Steuerprozess diskrete oder diskretwertige Bedingungen einhalten muss. Eine Klassifizierung aus diesem Blickwinkel richtet sich danach, welche der Funktionen von den diskreten Variablen abhängen. Eine weitere Art der Klassifizierung kann daraus abgeleitet werden, in welcher Form die Bedingungen und Variablen vorliegen, also ob es sich z. B. um ein Problem mit diskreter Zeit handelt oder nur lineare Funktionen vorkommen. Diese Aspekte sind hier im Punkt Spezialfälle zusammengefasst. Alle angegebenen Problemklassen können auch in beliebigen Kombinationen vorkommen.

2.2.1 Geschaltete Systeme

Bei geschalteten Systemen ändert sich die Dynamik in Abhängigkeit von diskreten Ereignissen. Das Eintreten dieser Ereignisse wird durch *Schaltgesetze* bestimmt, die zu jeder Zeit angeben, in welchem Zustand sich das System befindet. Die Schaltgesetze beinhalten autonome wie auch erzwungene Schaltungen. Eine umfassende Untersuchung von geschalteten Systemen kann in den

Arbeiten (XA04; XA03; XA01; Xu01; XA00) gefunden werden. Die folgenden beiden Abschnitte sollen einen kurzen Einblick in diese Systeme geben.

Intern schaltende Systeme

Von einem gegebenen Anfangszustand aus ändert sich der Zustand $\boldsymbol{x}(t)$ eines Systems mit Fortschreiten der Zeit entsprechend seiner Dynamik

$$\dot{\boldsymbol{x}}(t) = f_i(\boldsymbol{x}(t), t),\ \boldsymbol{x} \in \mathbb{X}_i,\ t \leq \hat{t},$$

bis eine dem System innewohnende *Schaltbedingung* (auch *Ereignisgenerator*)

$$T_i(\boldsymbol{x}(t), t),$$

die auch aus logischen Ausdrücken bestehen kann, zur Zeit $\hat{t}$ gleich null bzw. wahr wird, oder ihr Vorzeichen ändert und einen Moduswechsel zu einer anderen Dynamik

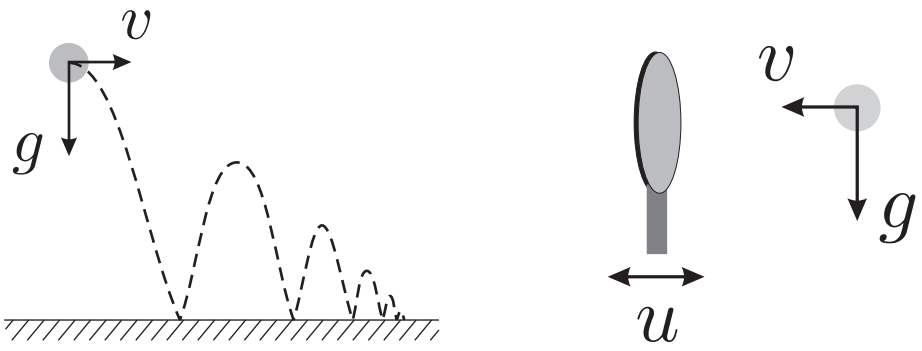
$$\dot{\boldsymbol{x}}(t) = f_j(\boldsymbol{x}(t), t),\ \boldsymbol{x} \in \mathbb{X}_j,\ t \geq \hat{t}$$

einleitet. Der Index der rechten Seite bestimmt den aktuellen Modus des Systems und wird manchmal auch diskreter Zustand des hybriden Systems genannt. Bei einem Moduswechsel kann sich auch die Anzahl der Zustände bzw. die der Gleichungen ändern, falls neue Bindungen aktiv oder inaktiv werden. Der Moduswechsel kann teilweise oder auch ganz glatt, nichtglatt, ja sogar unstetig sein. Die eventuell auftretenden Sprünge der Zustände $\boldsymbol{x}(t)$ sind über Translationsbedingungen

$$\boldsymbol{j}_i(\boldsymbol{x}(\hat{t}^-), \boldsymbol{x}(\hat{t}^+), \hat{t})$$

festgelegt. Diese Schaltung heißt *autonom*, da sie nicht direkt von einer Steuerung ausgelöst wird.

Ein einfaches Beispiel für ein autonom schaltendes System ist der Weitwurf eines Balls, wie er in Abbildung 2.1(a) auf der nächsten Seite dargestellt ist. Der Ball wird zunächst mit einer geeigneten Steuerung beschleunigt und verlässt, wenn er eine gewisse Geschwindigkeit und einen gewissen Abwurfwinkel erreicht hat, die Hand des Werfers. Anschließend fliegt der Ball, bis er mit dem Boden kollidiert und dabei durch den auftretenden Stoß bei nichtglatter Modellierung einen Sprung in seiner Geschwindigkeit sowie eine Richtungsänderung erfährt. Dabei verliert der Ball für gewöhnlich Energie. Dieser Stoß am Boden entspricht einer autonomen Schaltung. Der Ball springt nun so lange weiter, bis er schließlich liegen bleibt. Jeder dieser Stöße entspricht einer Schaltung im System. Hier zeigt sich allerdings schon das Problem von *Zenon* (auch Zeno, griech. Philosoph um 500 v. Chr.), da der Ball prinzipiell unendlich springen kann, bevor er endgültig liegen bleibt. Dies muss geeignet bei der Formulierung berücksichtigt werden. Es ist klar, dass bei diesem Beispiel jede Flugphase früher oder später durch einen Stoß am Boden abgeschlossen wird, worauf dann, falls der Ball noch genügend Energie besitzt, eine weitere Flugphase anschließt. Komplizierter wird es z. B. beim Tischtennis, siehe Abbildung 2.1(b) auf der nächsten Seite. Hier wird der Schläger durch die Steuerungen bewegt und kann nun fast zu beliebiger Zeit gegen den Ball stoßen



(a) Springender Ball (b) Tischtennis

Abbildung 2.1: Intern geschaltete Systeme

oder auch nicht. Damit hängt die Dynamik des Systems auch von den Steuerungen ab:

$$\dot{\boldsymbol{x}}(t) = f_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t),\ \boldsymbol{x} \in \mathbb{X}_i,\ \boldsymbol{u} \in \mathbb{U}_i.$$

Eine Schaltung erfolgt weiter in Abhängigkeit des Zustandes von Schläger und Ball, kann aber durch die Steuerung des Schlägers beeinflusst werden.

Beide Beispiele haben gemein, dass die Schaltbedingung nur von den Zuständen $\boldsymbol{x}(t)$ abhängt, wodurch das System eine gewisse Trägheit gegenüber der Steuerung besitzt und im Allgemeinen nicht unmittelbar aufgrund einer Steuereingabe schalten kann. Schaltungen dieser Art sind sozusagen systemimmanent und werden gar nicht oder nur indirekt über die kontinuierliche Steuerung $\boldsymbol{u}(t)$ beeinflusst.

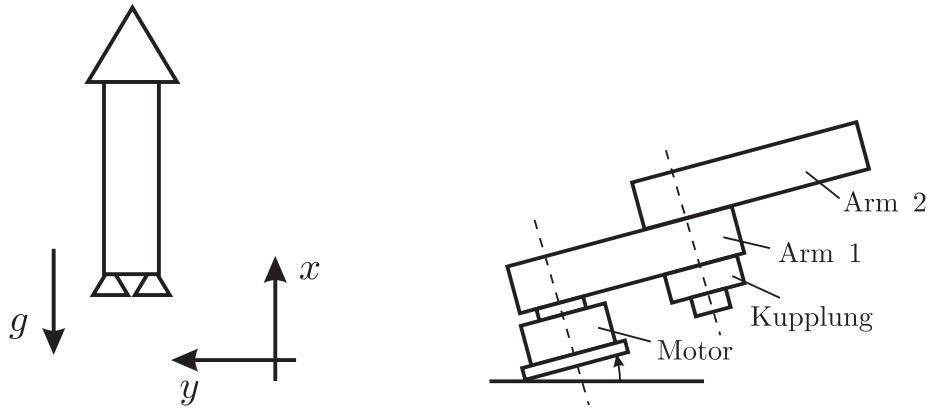
Intern schaltende Systeme haben meist eine niedrige Komplexität, da häufig die Folgezustände durch Schaltbedingungen bestimmt werden können.

Extern geschaltete Systeme

Falls keine Schaltbedingung vorhanden ist, und ein Wechsel der Dynamik allein von einer extern vorgegebenen Größe $\boldsymbol{w}(t)$ abhängt, können Schaltungen zu jeder Zeit auch unabhängig vom aktuellen Zustand des Systems stattfinden. Die diskrete Steuerung $\boldsymbol{w}(t)$ ersetzt sozusagen die Schaltbedingung und

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_i(\boldsymbol{x}(t), t),\ \boldsymbol{x} \in \mathbb{X}_i \text{ solange } \boldsymbol{w}(t) = \boldsymbol{w}_i.$$

Anhand eines einfachen Beispiels soll der Begriff der externen Schaltung genauer erklärt werden. Betrachtet wird der Aufstieg einer Rakete, formuliert wie in (Glo00), die als Punktmasse m modelliert ist; sie soll sich senkrecht nach oben bewegen und maximale Höhe bei minimalem Treibstoffverbrauch erreichen. Dazu kann das Triebwerk zu verschiedenen Zeitpunkten gezündet oder abgeschaltet werden. Die Rakete beschleunigt bei gezündetem Triebwerk maximal mit $\ddot{x}(t) = f_1 := m(a - g)$ und verzögert aufgrund der Erdanziehung bei abgeschaltetem Triebwerk mit $\ddot{x}(t) = f_2 := -mg$. Um den Treibstoffverbrauch möglichst gering zu halten, wird die Zeitspanne vollen Schubs minimiert. Als Schaltung kann einfach $w : [t_0, t_f] \rightarrow \{0, 1\}$ mit



(a) Aufstieg einer Rakete (b) Unteraktuierter Roboter

Abbildung 2.2: Extern geschaltete Systeme

$w(t) = 0$ bei abgeschaltetem und $w(t) = 1$ bei gezündetem Triebwerk verwendet werden. Die hier vorliegende Schaltung kann auch direkt in der Dynamik verwendet werden durch eine Formulierung der Art

$$\ddot{x}(t) = w(t)f_1 + (1 - w(t))\, f_2 = w(t)ma - mg,$$

woraus ersichtlich wird, dass ein Umschalten ohne jegliche Verzögerung stattfinden kann. Bei einem so einfachen Beispiel wie diesem ergibt sich die Steuerung bei der Optimierung automatisch aufgrund der Optimalitätsbedingungen, wie später in Abschnitt 2.3.3 auf Seite 35 erläutert wird. Da die Schaltreihenfolge bei diesem Beispiel trivial ist, gibt es noch andere einfache Formulierungen (Str01), die zur Lösung des Problems führen, hier aber nicht beachtet werden sollen.

Zu den extern geschalteten Systemen gehört auch der bereits in Abschnitt 1.1 auf Seite 2 erwähnte Roboter *R2D1* des Lehrstuhls für Steuerungs- und Regelungstechnik der Technischen Universität München (MBS98), der in 2.2(b) zu sehen ist. Sein Schultergelenk wird durch einen Motor angetrieben, sein Ellbogengelenk besitzt nur eine Kupplung, die den Unterarm zu beliebiger Zeit fixieren oder freigeben kann. Diese Kupplung kann durch eine binärwertige Steuerung beschrieben werden, die angibt, wann die Kupplung offen oder geschlossen ist. Auch hier ist die Schaltreihenfolge zunächst trivial und kann zur Lösung des Problems genutzt werden, indem Reihenfolge und Anzahl der Schaltungen vorgegeben und nur die Schaltzeitpunkte als Variablen in das Problem aufgenommen werden (BSBS00). Ein Ansatz ohne diese Vereinfachung lässt sich in (BGH$^+$02) beobachten. Erweiterungen des Manipulators durch zusätzliche nicht aktuierte Gelenke erlauben diese Vereinfachung nicht mehr. Die Schaltreihenfolge kann nicht mehr vorhergesagt werden. Nun muss eine Formulierung gefunden werden, bei der zu jedem bzw. zu einer bestimmten Anzahl von Zeitpunkten jede beliebige Kupplung geschaltet werden kann.

Extern geschaltete Systeme besitzen bei n_c möglichen Schaltpunkten und m_w Schaltmöglichkeiten ${m_w}^{n_c+1}$ mögliche Abläufe. Welche Komplexität ein Problem hat, hängt allerdings von den weiteren Faktoren ab und kann nicht allgemein beantwortet werden.

Bemerkung 7.
bei der Differentialgleichung $\dot{\boldsymbol{x}}(t) = f_i(\dots)$ wurde in diesem Kapitel die Schreibweise der Schal-

tung über einen Index gewählt, um den Unterschied zwischen intern und extern geschalteten Systemen besser herauszustellen. Beide Problemarten, sowohl die intern als auch die extern geschalteten Systeme, können auch in der allgemeineren Form, wie in (2.1) bis (2.5) auf Seite 17 beschrieben, formuliert werden.

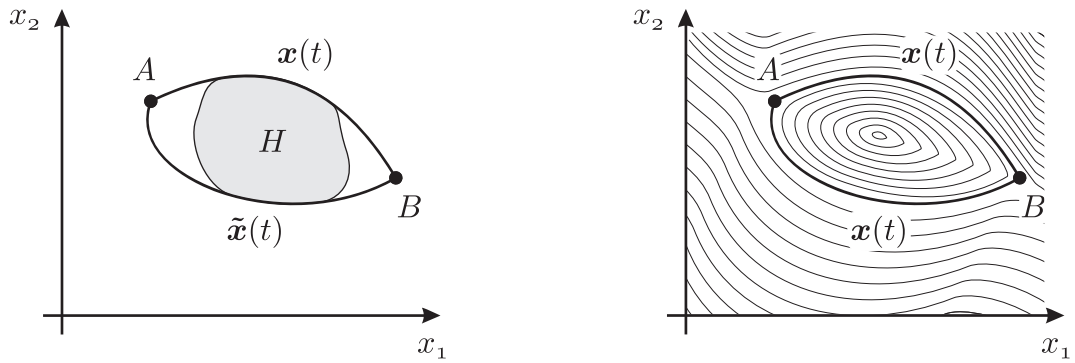
Bemerkung 8.
Häufig wird der diskrete Zustand im Index der rechten Seite auch in zeitabhängiger Form $i(t)$ mit

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f_{i(t)}(\dots) \\ i(t) &= \phi(\boldsymbol{x}(t), \boldsymbol{u}(t), i(t), t), \end{aligned}$$

und einem Schaltgesetz $\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{I} \times \mathbb{R} \to \mathbb{I}$ mit der Menge der Indices von Systemzuständen $\mathbb{I}$ geschrieben (HR99; BHS02; BSBS00). Ein Problem hierbei ist, dass ein Schaltgesetz ϕ in der Optimierung im Allgemeinen nicht bekannt ist, eine optimale Schaltung aber als Lösung gesucht wird.

2.2.2 Allgemeine hybride dynamische Systeme

Ein entscheidender Punkt, nämlich das Umschalten von Nebenbedingungen bei hybriden Systemen, ist bisher nicht diskutiert worden. Durch Hinzunahme von Nebenbedingungen, die von schaltenden Variablen abhängen, erweitert sich das Gebiet der hybriden Systeme beträchtlich, da nun abrupte Veränderungen der Umwelt berücksichtigt werden können. Somit werden die Systemzustände durch die von der Umwelt gegebenen Einschränkungen erweitert. Ein Auto ist dadurch nicht mehr nur im Zustand ‚Fahren' sondern z. B. im Zustand ‚auf der A5 von Darmstadt nach Frankfurt'. Die neu hinzukommenden Problemstellungen werden in diesem Abschnitt in eine Reihe wichtiger Aufgabenstellungen aus der diskreten Optimierung unterteilt.



(a) Bahnplanung (b) lokale Minima

Abbildung 2.3: Geschaltete Nebenbedingungen I

Bahnplanung mit Hindernissen

Durch eine Gleichungsnebenbedingung wird eine Untermannigfaltigkeit des Zustandsraums definiert, auf der die Trajektorie verlaufen darf. Diese Untermannigfaltigkeit kann durch Unglei-

chungsnebenbedingungen weiter eingeschränkt werden. Entstehen dadurch Löcher, sodass eine Trajektorie von einem Punkt A zu einem Punkt B nicht durch stetige Variation der Steuerung in jede andere zulässige Trajektorie überführt werden kann, ohne den zulässigen Bereich zu verlassen, so besteht das Problem der Bahnplanung mit Hindernissen (vgl. Abbildung 2.3(a)). Hier können Schaltungen in den Nebenbedingungen dazu verwendet werden, zwischen möglichen disjunkten zulässigen Teilbereichen zu schalten. Ein Beispiel für ein solches Problem wäre ein Auto, das sich durch ein Straßennetz bewegt. Zum einen müssen hier geeignete Straßen ausgewählt werden, zum anderen muss eine Steuerung berechnet werden, durch die das Auto geeignet bewegt werden kann. Ähnliche Probleme treten auf, falls sich die Hindernisse bewegen, wie bei der Kollisionsvermeidung von Flugzeugen unter der Annahme konstanter Flughöhe, da dann nur links oder rechts ausgewichen werden kann.

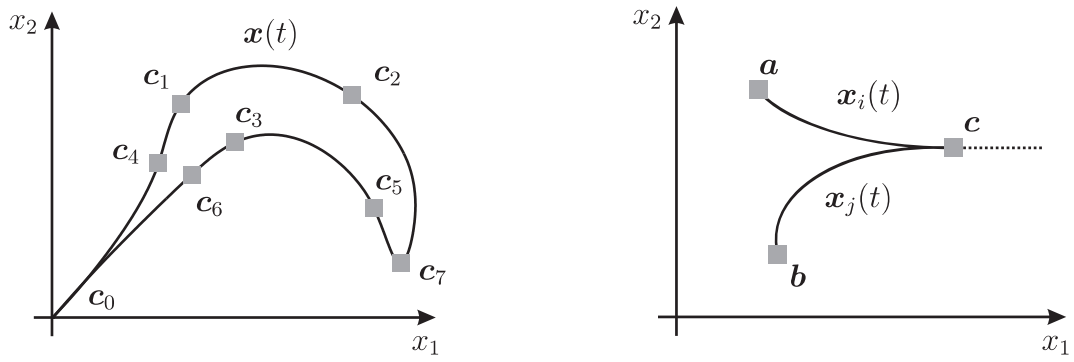
Alternativ können Probleme dieser Art auch so formuliert werden, dass jeder der möglichen Wege als lokales Minimum vorliegt (Abbildung 2.3(b)). In diesem Zusammenhang lässt sich auch die nahe Verwandtschaft von globaler und diskreter Optimierung zu erkennen.

Das allgemeine Problem, eine kollisionsfreie Bahn zu planen, ist PSPACE-schwer (vgl. Definition 23 im Anhang), was in (Rei79) bewiesen wurde (vgl. (RS94)). Dabei kommt es darauf an, wie viele Objekte zu berücksichtigen sind, und wie viele Freiheitsgrade sie besitzen. Die Zeitkomplexität ist polynomial in der Anzahl der Nebenbedingungen, die den Konfigurationsraum beschreiben, und doppelt exponentiell in der Dimension des Konfigurationsraums (Sha89). Ein Algorithmus, der innerhalb von Sekunden eine kürzeste kollisionsfreie Bahn für einen Mehrarmroboter in einer dreidimensionalen Umgebung mit vielen Hindernissen findet, kann daher kaum erwartet werden.

Rundreisen

Eine wichtige Klasse von Problemen sind Varianten des Rundreiseproblems. Sie treten, wie bereits in Abschnitt 1.1 auf Seite 4 beschrieben, z. B. in Fertigungsanlagen bei *Pick-and-Place*-Aufgaben auf, wo Roboter Objekte an definierten Punkten aufnehmen und an anderen definierten Punkten wieder absetzen. Sie finden sich auch im Luftverkehr, wenn z. B. Luftaufnahmen mehrerer Orte oder sogar bewegter Objekte in optimaler Reihenfolge gemacht werden sollen. Eine weitergehende Betrachtung kann in der Beispielanwendung in Abschnitt 5.3 auf Seite 110 gefunden werden.

Mathematisch lassen sich diese Probleme als Rundreise im Zustandsraum beschreiben. Es sei eine Menge von Punkten im Zustandsraum gegeben, von denen jeder Punkt genau einmal besucht werden soll. Falls die Zustände in den zu besuchenden Punkten nicht stetig differenzierbar sein müssen, können Verbindungen zwischen je zwei Punkten unabhängig voneinander berechnet werden, und es handelt sich um ein *klassisches Handlungsreisendenproblem* . Falls hier einige der Zustände stetig differenzierbar sind, oder Kollisionen mit bewegten Hindernissen vermieden werden sollen, kann die Reihenfolge nicht mehr von der Bahn getrennt berechnet werden. Ein weiterer wichtiger Aspekt ist, ob die geschlossene Rundreise im Startpunkt glatt sein soll oder nicht, da hiervon abhängt, ob Ansätze mit dynamischer Programmierung möglich sind. Durch die Forderung der



(a) Rundreisen (b) Scheduling

Abbildung 2.4: Geschaltete Nebenbedingungen II

Glattheit im Startpunkt geht die Unabhängigkeit aktueller Entscheidungen von zukünftigen verloren. Dies ist aber eine Eigenschaft, die für dynamische Programmierung erforderlich ist.

Rundreiseprobleme gehören zu den $\mathcal{NP}$-vollständigen Problemen (vgl. Definition 20 im Anhang). Die Abhängigkeit der Verbindungen zweier Städte von dem Rest der Rundreise verhindert es, effiziente Algorithmen anzuwenden, die für das Standart-Handlungsreisendenproblem gut funktionieren. Der symmetrische, glatte Fall mit n_c Städten besitzt $\frac{(n_c-1)!}{2}$ mögliche Rundreisen. Falls das Fahrzeug beim Verzögern andere Eigenschaften aufweist als beim Beschleunigen, handelt es sich um ein asymmetrisches motorisiertes Handlungsreisendenproblem mit $(n_c - 1)!$ Lösungskandidaten. Weiter kann die Glattheitsforderung der Rundreise in einer der Städte, z. B. in der Anfangsstadt, fallen gelassen werden, womit auch diese Einfluss auf die Lösung nimmt, und die Anzahl der Kandidaten auf $n_c!$ anwächst.

Maschinenbelegungsprobleme

Bei Schedulingproblemen geht es darum, verschiedene Aufgaben, die zu unterschiedlichen Zeitpunkten anfallen, auf Maschinen zu verteilen, die die Aufgaben erledigen. Die Maschinen haben dabei Rüstzeiten, um für eine gewisse Arbeit vorbereitet zu werden. Die Arbeitsaufgabe selbst benötigt eine gewisse Zeit, bis sie erledigt ist, und hat eventuell eine vorgegebene Priorität oder einen vorgegebenen Fertigstellungstermin.

Es können mehrere Aufgaben anfallen und auch mehrere Maschinen für die Bearbeitung zur Verfügung stehen. In vielen Fällen muss eine Arbeitsaufgabe mehrere Maschinen in einer vorgegebenen Reihenfolge durchlaufen, um fertig gestellt zu werden.

Im Zusammenhang mit Steuerungsproblemen werden hier Problemstellungen untersucht, bei denen auch die Bestückung der Maschine oder die Fertigstellung der Aufgabe als kontinuierlicher oder sogar als gemischt diskret-kontinuierlicher Prozess berücksichtigt wird, um eine optimale Belegung zu berechnen.

Eine derartige Problemstellung tritt z. B. am Ende einer Autobahn auf, wenn sich Autos, die zuvor mehrere Spuren zur Verfügung hatten, auf eine einzelne verbleibende einreihen müssen. Auch im

Bereich des *Luftfahrtmanagements* muss eine geeignete Reihenfolge für die Landung von Flugzeugen gefunden werden. Hier entspricht der Flughafen der Maschine, die belegt werden soll, und die Landung der Flugzeuge den zu erledigenden Aufgaben. Als kontinuierlicher Teil kommt die Landetrajektorie der Flugzeuge ins Spiel.

Die Frage einer optimalen *Verteilung von Aufgaben* stellt sich auch dann, wenn mehrere Systeme kooperativ agieren müssen. Diese Systeme bilden nebenläufige Geschehnisse, die zeitlicher Synchronisation bedürfen. Die Komplexität solcher Aufgaben steigt sehr rasch an, da abgesehen von der Verteilung auch die Reihenfolge der abzuarbeitenden Teilaufgaben berücksichtigt werden muss. In der Optimalsteuerung müssen alle gleichzeitig möglichen Verteilungen bedacht werden, da unter ihnen die optimale gesucht wird.

Beispiele für Problemstellungen, bei denen eine Aufgabenverteilung optimiert wird, sind Roboterfarmen in Fertigungsstraßen, falls mehrere Manipulatoren an einem Werkstück arbeiten, oder auch das Teamspiel im Fußball, bei dem die Aufgaben aus Ballführen, Freilaufen, Verteidigen, etc. bestehen.

Häufig gehören Maschinenbelegungsprobleme zu den $\mathcal{NP}$-vollständigen Problemen (Sch03).

2.2.3 Wichtige Spezialfälle

Linear quadratische Probleme

Die Regelungstechnik beschäftigt sich mit der Aufgabe, ein System in einem stationären Arbeitspunkt zu halten oder entlang einer Solltrajektorie zu bewegen. Falls nur mit kleinen Abweichungen des Systems von seinem Sollpunkt zu rechnen ist, reicht oft eine Linearisierung des Systems aus, um geeignete Reglerantworten zu berechnen, oder den Arbeitspunkt auf Stabilität zu untersuchen. Eine Vorgehensweise dieser Art liefert eine lineare Dynamik in den Abweichungen der Zustände von den Sollgrößen, die mit möglichst geringem Aufwand zu null geregelt werden sollen. Dieser Ansatz liefert ein linear-quadratisches Problem, gegeben durch ein quadratisches Zielfunktional

$$J[\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{q}] = \boldsymbol{x}(t_f)^T \boldsymbol{M} \boldsymbol{x}(t_f) + \int_{t_0}^{t_f} (\boldsymbol{x}(t)^T, \boldsymbol{u}(t)^T) \boldsymbol{Q}(\boldsymbol{w}(t), t) \begin{pmatrix} \boldsymbol{x}(t) \\ \boldsymbol{u}(t) \end{pmatrix} \mathrm{d}\, t$$

mit einer Systemdynamik

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(\boldsymbol{w}(t), t)\boldsymbol{x}(t) + \boldsymbol{B}(\boldsymbol{w}(t), t)\boldsymbol{u}(t), \boldsymbol{x} \in \mathbb{X}_i,\ \boldsymbol{u} \in \mathbb{U}_i,$$

die linear in den Zuständen $\boldsymbol{x}(t)$ und den Steuerungen $\boldsymbol{u}(t)$ ist. Die diskrete Steuerung wählt für jeden Zeitabschnitt die Matrizen $\boldsymbol{Q}(\boldsymbol{w}(t), t)$, $\boldsymbol{A}(\boldsymbol{w}(t), t)$ und $\boldsymbol{B}(\boldsymbol{w}(t), t)$, wodurch es sich um ein geschaltetes System (vgl. Abschnitt 2.2.1 auf Seite 19) handelt.

Wegen der Minimierung des quadratischen Terms im Integral streben $\boldsymbol{x}(t)$ und $\boldsymbol{u}(t)$ auf dem gesamten Zeitintervall $[t_0, t_f]$ gegen null. Also wird versucht, den Regelfehler unter Berücksichtigung des Steueraufwands gegen null zu führen.

Für jeden Zeitabschnitt, in dem $\boldsymbol{w}(t)$ konstant ist, kann das Problem explizit gelöst werden. Der Lösungsweg über die notwendigen Optimalitätsbedingungen (vgl. Abschnitt 2.3) liefert eine Matrix-*Riccati*-Differentialgleichung, deren Lösung die optimale Steuerung liefert. An Schaltzeitpunkten ist die Änderung der Riccati-Matrix stetig für externe und unstetig für autonome Schaltungen (vgl. (RKIZ99)).

Zeitdiskrete Probleme

Zeitdiskrete dynamische Probleme ergeben sich für gewöhnlich aus der Diskretisierung eines kontinuierlichen Problems. Bei gemischt diskret-kontinuierlichen zeitdiskreten Problemen wird versucht ein Zielfunktional

$$\min \quad J[\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{p}, \boldsymbol{q}] = \Phi(\boldsymbol{x}(K), \boldsymbol{p}, \boldsymbol{q}, K) + \sum_{k=0}^{K-1} L(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{w}(t), \boldsymbol{p}, \boldsymbol{q}, k)$$

unter den Nebenbedingungen

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{w}(k), \boldsymbol{p}, \boldsymbol{q}, k) \\ 0 &= \boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{w}(k), \boldsymbol{p}, \boldsymbol{q}, k) \\ 0 &\geq \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{w}(k), \boldsymbol{p}, \boldsymbol{q}, k) \end{aligned}$$

zu minimieren. Die Problemvariablen sind nun die Werte der Zustände $\boldsymbol{x}$, der Steuerungen $\boldsymbol{u}$ und $\boldsymbol{w}$ zu den diskreten Zeitpunkten $k = 0, \ldots, K$ sowie die Parameter $\boldsymbol{p}$ und $\boldsymbol{q}$. Damit handelt es sich bei zeitdiskreten Problemen um nichtlineare gemischt diskret-kontinuierliche Optimierungsprobleme der Art

$$\begin{aligned} &\min \quad \Phi(\boldsymbol{y}, \boldsymbol{z}) \\ &\text{unter} \quad \boldsymbol{y} \in \mathbb{R} := \{\boldsymbol{y} \in \mathbb{R}^{n_y}, \boldsymbol{z} \in \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_z}\} \mid \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{z}) = 0, \boldsymbol{h}(\boldsymbol{y}, \boldsymbol{z}) \leq 0\}, \end{aligned}$$

mit $\boldsymbol{y} = (\boldsymbol{x}(0)^T, \ldots, \boldsymbol{x}(K)^T, \boldsymbol{u}(0)^T, \ldots, \boldsymbol{u}(K)^T, \boldsymbol{p})$ und $\boldsymbol{z} = (\boldsymbol{w}(0)^T, \ldots, \boldsymbol{w}(K)^T, \boldsymbol{q})$. Insbesondere bei Echtzeitanwendungen kommen zeitdiskrete, in vielen Fällen sogar linear quadratische Formulierungen zum Einsatz (z. B. (BGKH02)).

2.3 Notwendige Optimalitätsbedingungen für dynamische Optimierungsprobleme

In diesem Abschnitt werden die notwendigen Bedingungen für einen optimalen Steuerprozess diskutiert. Falls eine numerische Lösung des Problems berechnet wurde, besteht die Möglichkeit, anhand der Optmalitätsbedingungen a posteriori zu überprüfen, ob wirklich eine optimale Lösung gefunden wurde.

Die Optimalsteuerungsaufgabe liegt dabei in autonomer Form vor und hat die Anfangszeit $t_0 = 0$,

was durch Anwendung von Bemerkung 5 auf Seite 18 erreicht werden kann.

$$\min \quad J[\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{q}] = \sum_{i=0}^{n_c} \Phi_i(\boldsymbol{x}(t_i), \boldsymbol{q}) + \int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \mathrm{d}\, t \tag{2.8}$$

$$\begin{aligned}
\text{unter} \quad \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) && (2.9)\\
0 &= \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) + \dot{\boldsymbol{z}}^2(t) && (2.10)\\
0 &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) && (2.11)\\
0 &= \boldsymbol{r}_i(\boldsymbol{x}(t_i), \boldsymbol{q}). && (2.12)
\end{aligned}$$

Im Gegensatz zu (2.1) bis (2.7) auf Seite 17 werden hier keine Transitionsbedingungen betrachtet. Falls diese vorliegen, muss das Problem zerlegt und die Bedingungen der separaten Teile müssen gekoppelt werden. Die Ungleichungsnebenbedingungen sind bereits nach Bemerkung 4 auf Seite 18 zu Differentialgleichungen umgewandelt worden. Es wird angenommen, dass die Anzahl der aktiven Ungleichungsnebenbedingungen zusammen mit der Anzahl der Gleichungsbedingungen zu keinem Zeitpunkt größer ist als die Anzahl aller Steuerungen, also noch ‚Raum' für eine Optimierung ist. Des Weiteren hat die Jacobi-Matrix der aktiven Ungleichungsnebenbedingungen $\boldsymbol{h}^a_{\boldsymbol{u},\boldsymbol{w}}$ und die der Gleichungsnebenbedingungen $\boldsymbol{g}_{\boldsymbol{u},\boldsymbol{w}}$ vollen Rang. Um die notwendigen Bedingungen mit den Mitteln der Variationsrechnung herzuleiten, wird angenommen, dass die Funktionen stetig, bzw. wenigstens stückweise stetig differenzierbar sind, und das Problem so geartet ist, dass zulässige Variationen (siehe Definition 24 im Anhang Seite 128) existieren, welche die gegebenen Nebenbedingungen erfüllen. Zuletzt sind keine kontinuierlichen und diskreten Parameter in der Formulierung enthalten und die diskreten Steuerungen zunächst als kontinuierlich angenommen mit $\boldsymbol{w} : [t_0, t_f] \to \mathbb{W} \supset \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m_w}\}$.

Zur kompakten Darstellung der Bedingungen werden die folgenden zwei Definitionen verwendet.

Definition 1 (Hamiltonfunktion)**.**
Die Funktion

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\lambda}) := L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) + \boldsymbol{\lambda}(t)^T \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t))$$

mit den adjungierten Variablen $\boldsymbol{\lambda} : [0, t_f] \to \mathbb{R}^{n_\lambda}$ wird Hamiltonfunktion genannt.

Definition 2 (erweiterte Hamiltonfunktion)**.**
Wird die Hamiltonfunktion zusätzlich mit einem Term gewichteter Nebenbedingungen versehen, wird sie als erweiterte Hamiltonfunktion

$$\begin{aligned}
\tilde{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \quad := \quad & H(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), \boldsymbol{\lambda}(t)) + \\
& \boldsymbol{\mu}_h(t)^T \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) + \\
& \boldsymbol{\mu}_g(t)^T \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t))
\end{aligned}$$

mit den Multiplikatorfunktionen $\boldsymbol{\mu} : [0, t_f] \to \mathbb{R}^{n_\mu},\ \boldsymbol{\mu} = (\boldsymbol{\mu}_h, \boldsymbol{\mu}_g)^T$ bezeichnet.

2.3.1 Bedingungen der Variationsrechnung

Gemäß der *Lagrange'schen Multiplikatorregel* der nichtlinearen Optimierung werden die Nebenbedingungen (2.9) bis (2.12) mit den *Lagrangemultiplikatoren* $\boldsymbol{\lambda}(t)$, $\boldsymbol{\mu}(t)$ und $\boldsymbol{\nu}_i$ an das Zielfunktional (2.8) gekoppelt:

$$\begin{aligned}&\int_0^{t_f} L(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{w}(t)) + \boldsymbol{\lambda}(t)^T\left(\boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{w}(t)) - \dot{\boldsymbol{x}}(t)\right)\\ &\quad + \boldsymbol{\mu}_h(t)^T\left(\boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{w}(t)) + \dot{\boldsymbol{z}}^2(t)\right) + \boldsymbol{\mu}_g(t)^T\boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{w}(t))\mathrm{d}\,t \qquad (2.13)\\ &+ \quad \sum_{i=0}^{n_c}\left(\Phi_i(\boldsymbol{x}(t_i),\boldsymbol{q}) + \boldsymbol{\nu}_i^T\boldsymbol{r}_i(\boldsymbol{x}(t_i),\boldsymbol{q})\right).\end{aligned}$$

Mit den zusätzlichen Variablen $\boldsymbol{x}_u$ und $\boldsymbol{x}_w$ mit $\dot{\boldsymbol{x}}_u = \boldsymbol{u}$ und $\dot{\boldsymbol{x}}_w = \boldsymbol{w}$ und $\bar{\boldsymbol{x}} = (\boldsymbol{x},\boldsymbol{z},\boldsymbol{x}_u,\boldsymbol{x}_w)^T$ sowie der Funktion $\hat{\Phi}_i(\bar{\boldsymbol{x}}(t_i),t_i) := \Phi_i(\boldsymbol{x}(t_i),\boldsymbol{q}) + \boldsymbol{\nu}_i^T\boldsymbol{r}_i(\boldsymbol{x}(t_i),\boldsymbol{q})$ kann (2.13) in der Form

$$\hat{\Phi}_0(\bar{\boldsymbol{x}}(t_0),t_0) + \sum_{i=1}^{n_c}\left(\hat{\Phi}_i(\bar{\boldsymbol{x}}(t_i),t_i) + \int_{t_i-1}^{t_i}\hat{L}(\bar{\boldsymbol{x}}(t),\dot{\bar{\boldsymbol{x}}}(t),t)\mathrm{d}\,t\right)$$

geschrieben werden, wodurch sich die Ergebnisse aus Abschnitt A.1 auf Seite 129 anwenden lassen. Unter Verwendung der Definitionen 1 und 2 ergeben sich die Bedingungen

$$\begin{aligned}
\dot{\boldsymbol{x}} &= H_{\boldsymbol{\lambda}} = \boldsymbol{f} && (2.14)\\
\dot{\boldsymbol{\lambda}} &= -\tilde{H}_{\boldsymbol{x}} = -L_{\boldsymbol{x}} - \boldsymbol{f}_{\boldsymbol{x}}^T\boldsymbol{\lambda} - \boldsymbol{h}_{\boldsymbol{x}}^T\boldsymbol{\mu}_h - \boldsymbol{g}_{\boldsymbol{x}}^T\boldsymbol{\mu}_g && (2.15)\\
\tilde{H}_{\boldsymbol{u}} &= 0 = L_{\boldsymbol{u}} + \boldsymbol{f}_{\boldsymbol{u}}^T\boldsymbol{\lambda} + \boldsymbol{h}_{\boldsymbol{u}}^T\boldsymbol{\mu}_h + \boldsymbol{g}_{\boldsymbol{u}}^T\boldsymbol{\mu}_g && (2.16)\\
\tilde{H}_{\boldsymbol{w}} &= 0 = L_{\boldsymbol{w}} + \boldsymbol{f}_{\boldsymbol{w}}^T\boldsymbol{\lambda} + \boldsymbol{h}_{\boldsymbol{w}}^T\boldsymbol{\mu}_h + \boldsymbol{g}_{\boldsymbol{w}}^T\boldsymbol{\mu}_g && (2.17)\\
0 &= \mu_{h,i}h_i, \quad i = 1,\ldots,n_h && (2.18)\\
0 &\leq \boldsymbol{\mu}_h && (2.19)\\
0 &= (\Phi_{0\,\boldsymbol{x}(t_0)} + \boldsymbol{r}_{0\,\boldsymbol{x}(t_0)}^T\boldsymbol{\nu}_0 + \boldsymbol{\lambda}(t_0))\delta\boldsymbol{x}_0 && (2.20)\\
0 &= (\Phi_{0\,t_0} + \boldsymbol{r}_{0\,t_0}^T\boldsymbol{\nu}_0 - \tilde{H}|_{t_0})\delta t_0 && (2.21)\\
0 &= (\Phi_{n_c\,\boldsymbol{x}(t_f)} + \boldsymbol{r}_{n_c\,\boldsymbol{x}(t_f)}^T\boldsymbol{\nu}_{n_c} + \boldsymbol{\lambda}(t_f))\delta\boldsymbol{x}_f && (2.22)\\
0 &= (\Phi_{n_c\,t_f} + \boldsymbol{r}_{n_c\,t_f}^T\boldsymbol{\nu}_{n_c} - \tilde{H}|_{t_f})\delta t_f && (2.23)\\
\boldsymbol{\lambda}(\hat{t}_i^-) &= \boldsymbol{\lambda}(\hat{t}_i^+) && (2.24)\\
\tilde{H}|_{\hat{t}_i^-} &= \tilde{H}|_{\hat{t}_i^+} && (2.25)\\
\boldsymbol{\lambda}(t_i^-) &= \Phi_{i\,\boldsymbol{x}(t_i)} + \boldsymbol{r}_{n_c\,\boldsymbol{x}(t_i)}^T\boldsymbol{\nu}_{n_c} + \boldsymbol{\lambda}(t_i^+) && (2.26)\\
\tilde{H}|_{t_i^+} &= \Phi_{i\,t_i} + \boldsymbol{r}_{n_c\,t_i}^T\boldsymbol{\nu}_{n_c} + \tilde{H}|_{t_i^+}. && (2.27)
\end{aligned}$$

In den folgenden Abschnitten werden diese Bedingungen und ihre Bedeutung genauer erläutert.

Kanonische Differentialgleichungen

Gleichung (2.14) ist direkt in der Problemstellung enthalten und gibt an, wie die Zustände aus einer gegebenen Steuerung berechnet werden können.

Die Euler-Lagrange-Gleichungen

$$\hat{L}_{\bar{\boldsymbol{x}}} - \frac{d}{dt}\hat{L}_{\dot{\bar{\boldsymbol{x}}}} \;=\; 0$$

führen für den Integranden

$$\hat{L} = L(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w}) + \boldsymbol{\lambda}(t)^T(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w}) - \dot{\boldsymbol{x}}) + \boldsymbol{\mu}_h^T\left(\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w}) + \dot{\boldsymbol{z}}^2\right) + \boldsymbol{\mu}^T\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w})$$

auf

$$\begin{aligned}
\hat{L}_{\boldsymbol{x}} - \frac{d}{dt}\hat{L}_{\dot{\boldsymbol{x}}} &= L_{\boldsymbol{x}} + \boldsymbol{\lambda}(t)^T\boldsymbol{f}_{\boldsymbol{x}} + \boldsymbol{\mu}(t)^T\boldsymbol{g}_{\boldsymbol{x}} + \dot{\boldsymbol{\lambda}} \;=\; 0,\\
\hat{L}_{\boldsymbol{x}_u} - \frac{d}{dt}\hat{L}_{\dot{\boldsymbol{x}}_u} &= 0 \quad\Rightarrow\quad L_{\boldsymbol{u}} + \boldsymbol{\lambda}^T\boldsymbol{f}_{\boldsymbol{u}} + \boldsymbol{\mu}_h^T\boldsymbol{h}_{\boldsymbol{u}} + \boldsymbol{\mu}_g^T\boldsymbol{g}_{\boldsymbol{u}} = \text{konstant},\\
\hat{L}_{\boldsymbol{x}_w} - \frac{d}{dt}\hat{L}_{\dot{\boldsymbol{x}}_w} &= 0 \quad\Rightarrow\quad L_{\boldsymbol{w}} + \boldsymbol{\lambda}(t)^T\boldsymbol{f}_{\boldsymbol{w}} + \boldsymbol{\mu}_h^T\boldsymbol{h}_{\boldsymbol{w}} + \boldsymbol{\mu}_g^T\boldsymbol{g}_{\boldsymbol{w}} = \text{konstant}
\end{aligned}$$

und sind durch die Gleichungen (2.15) bis (2.17) dargestellt. Die Lagrangemultiplikatoren $\boldsymbol{\lambda}(t)$ werden auch als *Kozustände* und die dazugehörigen Differentialgleichungen (2.15) als *Kozustandsdifferentialgleichungen* bezeichnet. Diese bilden mit den Zustandsdifferentialgleichungen (2.14) ein System von Differentialgleichungen 1.Ordnung, die *kanonische Differentialgleichungen* genannt werden.

Die *Koppelgleichungen* (2.16) und (2.17) müssen konstant sein. Aus den Transversalitätsbedingungen (A.15), (A.17) und (A.19) auf Seite 131 folgt, dass diese Konstanten gleich null sind.

Die Steuerungen $\boldsymbol{u}(t)$ und $\boldsymbol{w}(t)$ lassen sich durch die Koppelgleichungen als Funktionen von $\boldsymbol{x}(t)$ und $\boldsymbol{\lambda}(t)$ ausdrücken und in die kanonischen Gleichungen einsetzen.

Ungleichungsnebenbedingungen

Des Weiteren folgt aus den *Euler-Lagrange*-Gleichungen für die Ungleichungsnebenbedingungen

$$\hat{L}_{\boldsymbol{z}} - \frac{d}{dt}\hat{L}_{\dot{\boldsymbol{z}}} = 0 \quad\Rightarrow\quad 2\boldsymbol{\mu}_h^T\dot{\boldsymbol{z}} = \text{konstant}.$$

Für diese Gleichung gilt dasselbe wie für die Koppelgleichungen. Die Konstante muss null sein, womit unmittelbar aus $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{w}) + \dot{\boldsymbol{z}}^2$ die *Komplementarität* (2.18) folgt. Die Positivität der Multiplikatoren $\boldsymbol{\mu}_h$ kann unter Verwendung der Legendreschen Bedingung gezeigt werden (Pap96).

Transversalitätsbedingungen

Um die kanonischen Gleichungen zu lösen, werden noch Randbedingungen benötigt, die aus den Transversalitätsbedingungen (2.20) bis (2.23) hervorgehen. Zur genaueren Erläuterung soll hier die Transversalitätsbedingung

$$\left(\hat{\Phi}_{t_f} + \hat{L}\Big|_{t_f} - \hat{L}_{\dot{\boldsymbol{x}}}\Big|_{t_f}\dot{\bar{\boldsymbol{x}}}\right)\delta t_f = 0$$

für den Zeitpunkt t_f, aus der sich Gleichung (2.23) ergibt, genauer betrachtet werden. δt_f ist hier eine zulässige Variation der Endzeit. Bei fester Endzeit t_f muss eine zulässige Variation $\delta t_f = 0$ sein, womit die Bedingung immer erfüllt ist. Ist die Endzeit t_f jedoch frei, so darf eine zulässige Variation δt_f beliebige Werte annehmen, weshalb

$$\hat{\Phi}_{t_f} + \hat{L}\Big|_{t_f} - \hat{L}_{\dot{\boldsymbol{x}}}\Big|_{t_f} \dot{\boldsymbol{x}} = 0$$

sein muss. Falls vorhanden, müssen auch die Endbedingungen $\boldsymbol{r}_{n_c}(\boldsymbol{x}(t_f), t_f) = 0$ erfüllt werden. Analog gilt dies auch für die Gleichungen (2.20) bis (2.22).

Sind zusätzlich noch Bedingungen an inneren Punkten t_i vorhanden, müssen weitere Transversalitätsbedingungen erfüllt sein. Die Zustände $\boldsymbol{x}(t_i)$ zu den Zeitpunkten t_i sind ebenso wie die Zeitpunkte t_i frei, weshalb die Variationen $\delta \boldsymbol{x}_i$ und δt_i beliebig sein dürfen. Dies führt unmittelbar auf die Gleichungen (2.26) und (2.27) auf Seite 29. Um diese Gleichungen erfüllen zu können, müssen die Kozustände und die Hamiltonfunktion zu den Zeitpunkten t_i Sprünge enthalten.

Ein wichtiger Fall im Zusammenhang mit geschalteten Systemen sei hier noch erwähnt. Falls ein Differentialgleichungssystem mit zustandsabhängigen Schaltungen

$$\dot{\boldsymbol{x}}(t) = \begin{cases} \boldsymbol{f}_1(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)), & \text{wenn} \quad \boldsymbol{T}(\boldsymbol{x}(t), t) \leq 0 \\ \boldsymbol{f}_2(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)), & \text{wenn} \quad \boldsymbol{T}(\boldsymbol{x}(t), t) > 0 \end{cases}$$

vorliegt, so ist die Hamiltonfunktion gegeben durch

$$H = \begin{cases} H_1 = L + \boldsymbol{\lambda}(t)^T \boldsymbol{f}_1 \\ H_2 = L + \boldsymbol{\lambda}(t)^T \boldsymbol{f}_2. \end{cases}$$

Damit eine Schaltung stattfindet, muss $\boldsymbol{T}(\boldsymbol{x}(\tilde{t}), \boldsymbol{u}(\tilde{t}), \tilde{t}) = 0$ erfüllt sein, was einer inneren Punktbedingung entspricht. Wird $\boldsymbol{T}(.) = 0$ wie eine innere Punktbedingung behandelt, kommen die weiteren Bedingungen

$$H_1|_{\tilde{t}^+} = H_2|_{\tilde{t}^-} - \boldsymbol{T}_{\tilde{t}}^T \tilde{\boldsymbol{\nu}}$$

hinzu.

Bemerkung 9.
Da bei extern geschalteten Systemen keine Schaltbedingung $\boldsymbol{T}(\boldsymbol{x}(t), t)$ vorliegt, gilt hier $H_1|_{\tilde{t}^+} = H_2|_{\tilde{t}^-}$.

Weierstraß-Erdmann-Eckbedingungen

Durch die bisher genannten Bedingungen sind einmal stetig differenzierbare $\boldsymbol{x}(t)$ und $\boldsymbol{\lambda}(t)$ bereits festgelegt. Falls unter diesen Funktionen keine Lösungen gefunden werden, können auch Funktionen mit Ecken, also stückweise stetig differenzierbare $\boldsymbol{x}(t)$ und $\boldsymbol{\lambda}(t)$, als Lösungen zugelassen werden. Hierfür müssen die Bedingungen (2.14) bis (2.23) um die Weierstraß-Erdmann-Eckbedingungen (2.24) und (2.25) erweitert werden.

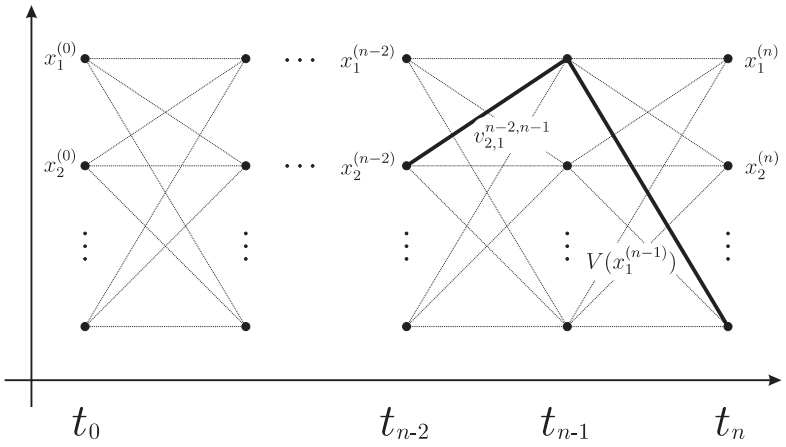


Abbildung 2.5: Dynamische Programmierung

2.3.2 Hamilton-Jacobi-Bellman-Gleichung

Die im letzten Abschnitt vorgestellten Optimalitätsbedingungen basieren auf den Resultaten der Variationsrechnung. In diesem Abschnitt wird nun, ausgehend von dem Bellmanschen Optimalitätsprinzip, die Hamilton-Jacobi-Bellman-Gleichung aufgestellt. Diese können auch zur Herleitung von Lösungsverfahren verwendet werden, wie z.B. dem Dekompositionsansatz der in dieser Arbeit vorgestellt wird.

Bellmansches Optimalitätsprinzip

Wenn ein Optimierungsproblem als gesuchte zeitliche Abfolge von Entscheidungen interpretiert werden kann, wie es bei Optimalsteuerungsproblemen der Fall ist, so lässt sich auch *dynamische Programmierung* zur Lösung einsetzen. Diese basiert auf dem folgenden Satz.

Satz 1. *Bellmansches Optimalitätsprinzip:*
Eine optimale Entscheidungsfolge hat die Eigenschaft, dass für einen beliebigen Anfangszustand $x^{(0)}$ und beliebige (eventuell nichtoptimale) Entscheidungen $u^{(0)}$, die übrigen Entscheidungsfunktionen eine optimale Entscheidungsfolge bilden bezüglich des Zustandes $x^{(1)}$, der sich aus der Entscheidung $u^{(0)}$ ergibt.

Entscheidend für das Bellmansche Optimalitätsprinzip ist, dass zukünftige Entscheidungen unabhängig von den vergangenen sein müssen (*Markoveigenschaft*). Aufgrund dieser Eigenschaft lässt sich nun ein Verfahren ableiten.

Beginnend bei der Endzeit t_n wird die Zeit schrittweise rückwärts durchlaufen. Zu einem Zeitpunkt t_i werden für jeden möglichen Problemzustand $x_j^{(i)}$ mit $i = 0, \ldots, n$ und $j = 1, \ldots, n_x^{(i)}$ die minimalen Restkosten $V(x_j^{(i)})$, um einen zulässigen Zustand zur Endzeit zu erreichen, notiert.

Es seien die optimalen Restkosten $V(x_k^{(i+1)})$ für alle zulässigen Zustände $x_k^{(i+1)}$ zum Zeitpunkt t_{i+1} bereits bekannt. Nun werden für jeden zulässigen Zustand $x_j^{(i)}$ zur Zeit t_i die optimalen Restkosten

als Minimum der Kosten $v_{j,k}^{(i,i+1)}$ zum Erreichen eines Zustands $x_k^{(i+1)}$ zum Zeitpunkt t_{i+1} durch eine optimale Entscheidung $u^{*(i)}$ und den Restkosten $V(x_k^{(i+1)})$ von dort bis zur Endzeit ermittelt.

$$V(x_j^{(i)}) = \min_k v_{j,k}^{(i,i+1)} + V(x_k^{(i+1)}). \tag{2.28}$$

Wird nun ein zulässiger Zustand zur Anfangszeit t_0 erreicht, so können die optimalen Kosten abgelesen werden. Implizite Randbedingungen $r(x_0, x_n) = 0$ können mit diesem Verfahren nicht berücksichtigt werden, da diese die Markoveigenschaft verletzen. Die beschriebene Vorgehensweise wird dynamische Programmierung genannt und hat nicht nur im Diskreten eine wichtige Bedeutung.

Gemischt-ganzzahlige Optimalsteuerungsprobleme können gleichzeitig in Richtung der Zeit als auch in Richtung der Zustände diskretisiert werden. Auf dieser Diskretisierung kann dann mit Hilfe dynamischer Programmierung eine optimale Lösung bestimmt werden.

Wertefunktion

Für den kontinuierlichen Fall wird die Wertefunktion $V : \mathbb{X} \times [t_0, t_f] \to \mathbb{R}$ betrachtet mit

$$V(\boldsymbol{x}, t) = \min_{\boldsymbol{u},\boldsymbol{w}} \left\{ \int_t^{t_f} L(\boldsymbol{x}(\theta), \boldsymbol{u}(\theta), \boldsymbol{w}(\theta), \theta) \mathrm{d}\theta \right\},$$

welche die Mindestkosten zur Überführung eines Zustands $\boldsymbol{x}(t)$ in den Zustand $\boldsymbol{x}(t_f)$ beschreibt. Die Minimierung erfolgt über den Bereich zulässiger Steuerungen $(\boldsymbol{u}, \boldsymbol{w}) \in \mathbb{U}(\boldsymbol{x}, t) \times \mathbb{W}(\boldsymbol{x}, t)$, der durch die Gleichungs- und Ungleichungsnebenbedingungen gegeben ist. Die Überführungskosten können nun zweigeteilt werden in die Kosten des nächsten Wegstücks und die Restkosten.

$$V(\boldsymbol{x}, t) = \min_{\boldsymbol{u},\boldsymbol{w}} \left\{ \int_t^{t+h} L(\boldsymbol{x}(\theta), \boldsymbol{u}(\theta), \boldsymbol{w}(\theta), \theta) \mathrm{d}\theta + V(\boldsymbol{x}(t+h), t+h) \right\}$$

Das Integral wird in erster Näherung über den Rechtecksinhalt angenähert, und die Restkosten in eine *Taylorreihe* um t entwickelt.

$$V(\boldsymbol{x}, t) \;=\; \min_{\boldsymbol{u},\boldsymbol{w}} \left\{ L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), t)h + V_{\boldsymbol{x}}^T \dot{\boldsymbol{x}} h + V_t h + V(\boldsymbol{x}, t) \right\} + o(h)$$

Da $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, t)$ von den Steuerungen $\boldsymbol{u}$ und $\boldsymbol{w}$ abhängt, muss dieser Term bei der Minimierung berücksichtigt werden; $V(\boldsymbol{x}, t)$ und V_t können jedoch auf die linke Seite gebracht werden.

$$-V_t h = \min_{\boldsymbol{u},\boldsymbol{w}} (L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), t) + V_{\boldsymbol{x}}^T \dot{\boldsymbol{x}}) h + o(h)$$

Wird h gekürzt, $\dot{\boldsymbol{x}}$ durch die rechte Seite von (2.9) auf Seite 28 ersetzt und der Grenzwert $h \to 0$ gebildet, muss nur noch über $\boldsymbol{u}(t)$ und $\boldsymbol{w}$ (t) zum Zeitpunkt t minimiert werden.

$$-V_t = \min_{\boldsymbol{u}(t),\boldsymbol{w}(t)} \left\{ L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), t) + V_{\boldsymbol{x}}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, t) \right\} \tag{2.29}$$

Diese nichtlineare partielle Differentialgleichung 1. Ordnung wird Hamilton-Jacobi-Bellman-Gleichung genannt, da Überlegungen dieser Art mit Bellmans Aufgaben zur dynamischen Programmierung in Zusammenhang gebracht werden, auch wenn sie eigentlich zuvor von C. Carathéodory veröffentlicht wurden (PB94).

Die Hamilton-Jacobi-Bellman-Gleichung spielt besonders in der Regelungstechnik eine wichtige Rolle, da eine Lösung ein optimales Regelgesetz liefert.

2.3.3 Minimumprinzip

Ausgehend von den Hamilton-Jacobi-Bellman-Gleichung kann ein Minimumprinzip abgeleitet werden. Dieses ist allerdings schwächer als das so genannte *Pontrjaginsche Maximumprinzip,* da es zweimal stetige Differenzierbarkeit der Wertefunktion verlangt. Welche Bedeutung das Minimumprinzip im Falle diskretwertiger Steuerungen hat, wird im zweiten Teil dieses Abschnitts besprochen.

Formulierung eines Minimumprinzips

Der zu minimierende Term in Gleichung (2.29) erinnert an die Hamiltonfunktion $H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = L(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ aus Definition 1 auf Seite 28.

$$V_t = - \min_{\boldsymbol{u}(t), \boldsymbol{w}(t)} H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, V_{\boldsymbol{x}}, t) =: \mathcal{H}(\boldsymbol{x}, V_{\boldsymbol{x}}, t)$$

Ist $V(\boldsymbol{x}, t)$ zweimal stetig differenzierbar, dann gilt entlang optimaler Trajektorien $\boldsymbol{x}^*$ mit Steuerungen $\boldsymbol{u}^*$ und $\boldsymbol{w}^*$

$$\begin{aligned} V_{t\,\boldsymbol{x}} &= -L_{\boldsymbol{x}} - V_{\boldsymbol{x}\boldsymbol{x}}^T \boldsymbol{f} - V_{\boldsymbol{x}}^T \boldsymbol{f}_{\boldsymbol{x}} \\ \frac{d}{dt} V_{\boldsymbol{x}} &= V_{\boldsymbol{x}\boldsymbol{x}}^T \dot{\boldsymbol{x}} + V_{\boldsymbol{x}\,t}. \end{aligned}$$

Wegen $V_{t\,\boldsymbol{x}} = V_{\boldsymbol{x}\,t}$ ergibt sich die adjungierte Gleichung (2.15) auf Seite 29 mit $\boldsymbol{\lambda} = V_{\boldsymbol{x}}$

$$\frac{d}{dt} V_{\boldsymbol{x}} = -L_{\boldsymbol{x}} - V_{\boldsymbol{x}}^T \boldsymbol{f}_{\boldsymbol{x}}.$$

Damit kann ein Minimumprinzip formuliert werden:

Satz 2. *Minimumprinzip:*
Sei $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{w}^*)$ ein Prozess, der die Optimalsteuerungsaufgabe (2.1) bis (2.5) minimiert, dann existieren Multiplikatorfunktionen $\boldsymbol{\lambda} : [0, t_f] \to \mathbb{R}^{n_\lambda}$, $\boldsymbol{\mu} : [0, t_f] \to \mathbb{R}^{n_\mu}$ und Multiplikatoren $\boldsymbol{\nu} \in \mathbb{R}^{n_\nu}$, sodass die Gleichungen (2.14) bis (2.27) erfüllt sind und für fast alle $t \in [0, t_f]$ gilt

$$H(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{w}^*, \boldsymbol{\lambda}, t) = \min_{\boldsymbol{u}(t), \boldsymbol{w}(t)} H(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\lambda}, t) =: \mathcal{H}(\boldsymbol{x}^*, \boldsymbol{\lambda}, t).$$

Bedeutung für hybride Probleme

Bisher sind die Steuerungen $\boldsymbol{w}$ noch als kontinuierlich angenommen. Da das Minimumprinzip auch unter Ungleichungs- und Gleichungsbedingungen an die Steuerungen gilt, kann hier z. B.

$$(\boldsymbol{w}_i - \boldsymbol{w}_{0,i})(\boldsymbol{w}_i - \boldsymbol{w}_{1,i}) \ldots (\boldsymbol{w}_i - \boldsymbol{w}_{m_w,i}) = 0$$

gefordert werden, womit $\boldsymbol{w}$ wieder auf seinen diskreten Wertebereich eingeschränkt ist (Pap96). Damit kann das Minimumprinzip auch für Steuerungen mit diskretem Wertebereich angewandt werden.

Falls eine Steuerung u_i mit $u_{i,min} \leq u_i \leq u_{i,max}$ in der Hamiltonfunktion linear auftritt und damit $\frac{\partial H}{\partial u_i}$ unabhängig von $\boldsymbol{u}_i$ ist, kann ihr Verlauf anhand des Vorzeichens der Schaltfunktion

$$s_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\lambda}) = \frac{\partial H}{\partial u_i}$$

bestimmt werden.

$$\boldsymbol{u}_i(t) = \begin{cases} u_{i,min} & \text{falls} \quad s_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\lambda}) > 0 \\ u_{i,max} & \text{falls} \quad s_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\lambda}) < 0 \end{cases}$$

Insbesondere gilt dies auch für Steuerungen $\boldsymbol{w}(t)$, die bei entsprechender Formulierung linear auftreten, was im späteren Verlauf noch gezeigt wird. Haben diese gerade den diskreten Wertebereich $w_{i,min}, w_{i,max}$, so wird dieser auch im relaxierten Fall, d. h. $w_i : [0, t_f] \rightarrow [w_{i,min}, w_{i,max}]$, angenommen. Steuerungen, bei denen die Schaltfunktion nur Nullstellen besitzt und nicht auf ganzen Teilintervallen identisch null ist, heißen *Bang-bang-Steuerungen*.

Problematisch ist der Fall, bei dem auf einem ganzen Teilintervall $s_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\lambda}) = 0$ gilt, und somit eine singuläre Steuerung vorliegt. Steuerungen können in diesem Fall aus den Ableitungen $\frac{\mathrm{d}^k s_i}{\mathrm{d}\, t^k}$ mit $\partial \frac{\mathrm{d}^k s_i}{\mathrm{d}\, t^k} / \partial \boldsymbol{u}$ berechnet werden.

Kapitel 3

Modellierung von diskret-kontinuierlichen dynamischen Prozessen

Einer der ersten Schritte zur Berechnung der optimalen Steuerung eines Systems ist die Erzeugung eines geeigneten Modells für die weiteren Arbeitsschritte. Ein Modell ist ein vereinfachtes Abbild eines Originals, das selbst ein Modell sein kann, und enthält nur die Eigenschaften, die dem Modellierer wichtig erschienen. Somit spiegelt ein Modell eine subjektive Wahrnehmung wider und ist nicht eindeutig. Verschiedene Modelle haben unterschiedliche Zielsetzungen und nicht alle sind für die numerische Optimierung geeignet.

Modelle können auf mehreren Abstraktionsebenen angesiedelt sein, von einem konzeptionellen Modell bis hin zu einer mathematischen Beschreibung, die verwendet werden kann, um, wie hier angestrebt, optimale Steuerungen zu berechnen. Sowie die betrachteten Systeme komplexer werden, helfen Modellierungswerkzeuge bei der Validierung und auch bei der automatischen Erzeugung des Codes einer mathematischen Formulierung (TB04). Ein automatisch generierter Code ist im Allgemeinen langsamer als ein handoptimierter, jedoch ist die Zeit nicht zu vernachlässigen, die ein Modellierer benötigt, um ein mathematisches Modell effizient am Computer zu implementieren und zu validieren. In dieser Arbeit wird die automatische Codegenerierung nicht verwendet, die folgenden Kapitel sollen aber die Möglichkeiten dazu aufzeigen.

Im ersten Abschnitt dieses Kapitels wird auf Hilfsmittel der Modellierung eingegangen, wobei die gemischt diskret-kontinuierlichen Aspekte im Vordergrund stehen. Auf die spezielle Modellierung der kontinuierlichen Vorgänge wird weniger stark eingegangen, da diese im Allgemeinen anwendungsspezifisch sind. Aufbauend auf den Ergebnissen des ersten Abschnitts werden anschließend Formalismen besprochen, die es erleichtern, geeignete mathematische Formulierungen für die numerischen Berechnungen zu erzeugen.

3.1 Modellierungswerkzeuge

Nachdem ein konzeptionelles Modell vorliegt, werden hybride Systeme in den technischen Anwendungsgebieten häufig mit hybriden Automaten (ACHH92), Petrinetzen (BSN$^+$99)

und gelegentlich auch mit Bondgraphen (Mos97) modelliert. Modellierer bevorzugen für gewöhnlich diskret-kontinuierliche Erweiterungen der Werkzeuge, die in dem entsprechenden Anwendungsgebiet aus dem rein Diskreten oder rein Kontinuierlichen bereits bekannt sind. Alle Herangehensweisen bergen Vor- und Nachteile. So sind z. B. Petrinetze besonders gut geeignet, um nebenläufige konkurrierende oder kooperierende Systeme zu modellieren (OREM00), auf der anderen Seite werden sie schnell groß und unübersichtlich. Hybride Automaten eignen sich gut zur Modellierung und Untersuchung von geschalteten Systemen. In den folgenden Abschnitten werden hybride Automaten in Verbindung mit gemischt diskret-kontinuierlicher Optimalsteuerung untersucht.

3.1.1 Hybride Automaten

In der Informatik werden Maschinen, die nach bestimmten Regeln arbeiten, Automaten genannt. Diese Maschinen müssen dabei in der Realität nicht zwingend existieren. Sie wurden ursprünglich als einfaches Modell für Nervennetze eingeführt. Die Regeln nach denen ein Automat arbeitet, werden durch ein Programm vorgegeben. Automaten werden hauptsächlich zur Verifikation eingesetzt. Hierbei wird z. B. untersucht, ob gewisse Zustände erreichbar sind, oder ob es *Deadlocks*, also Zustände, die nicht mehr verlassen werden können, gibt. Gegebenenfalls kann auch das Problem von Zenon, d. h. unendlich viele Schaltungen zwischen Systemzuständen in endlicher Zeit, auftreten. Automaten können durch Graphen dargestellt werden, wobei die Knoten den diskreten Systemzuständen entsprechen, und die Kanten den Übergängen zwischen diesen. Falls der Folgezustand nicht durch einen eindeutigen Übergang bestimmbar ist, heißt der Automat nichtdeterministisch, anderenfalls deterministisch.

Die Zustände des Automaten werden instantan durchlaufen. Für Systeme, die in einem Zustand eine gewisse Zeit verharren sollen, kann ein Automat mit Stoppuhren ausgestattet werden, die folgendermaßen definiert sind (vgl. auch Bemerkung 5 auf Seite 18).

Definition 3 (Stoppuhr)**.**
Eine Stoppuhr ist eine Systemvariable mit Anfangswert $x_t(0) = 0$, Flussgleichung $\dot{x}_t = 1$ bei laufender und $\dot{x}_t = 0$ bei stillstehender Uhr.

Wird ein neuer Zustand des Systems erreicht, so startet die Stoppuhr und misst die Zeit, bis der Zustand wieder verlassen wird. Gegebenenfalls kann die Stoppuhr bei einem Wechsel in einen anderen Zustand zurückgestellt werden. Bei Zustandswechseln verstreicht keine Zeit. Produkte solcher Automaten benötigen eine zeitliche Synchronisation. Automaten mit Uhren werden zeitlich abgestimmte Automaten genannt (Alu99).

Durch die Zeitmessung ist es auch denkbar, zeitkontinuierliche Abläufe zu untersuchen. Dies gelingt mit der Einführung *hybrider Automaten* (vgl. (Hen00)), die hier um Steuerungen erweitert werden.

Definition 4 (Hybrider Automat)**.**
Ein hybrider Automat $H = (V, E, \mathbb{X}, \mathbb{U}, init, inv, flow, jump, event)$ besteht aus

- einem endlichen, gerichteten Multigraphen (V, E) (vgl. Abschnitt A.1 auf Seite 126), mit Knoten aus V, die als Systemzustand bezeichnet werden, und Kanten aus E, den so genannten Systemschaltungen;
- einer Menge kontinuierlicher Zustandsvariablen $\mathbb{X} = \{x_1, \ldots, x_{n_x}\}$;
- einer Menge kontinuierlicher Steuervariablen $\mathbb{U} = \{u_1, \ldots, u_{n_u}\}$;
- einer Abbildung $init$, die einer Kante eine Anfangsbedingung zuordnet;
- den *Invarianten*, gegeben durch eine Abbildung inv, die jedem Knoten durch Gleichungs- und Ungleichungsbedingungen einen zulässigen Bereich für die kontinuierlichen Zustände zuordnet;
- einer Abbildung $flow$, die jedem Systemzustand eine *Flussgleichung* bzw. Dynamik zuordnet;
- einer Abbildung $jump$, die den Kanten Sprungbedingungen zuordnet,
- einer Abbildung $event$, die den Kanten Ereignisse, die beim Schalten eintreten, zuordnet.

3.1.2 Zusammenhang zwischen hybriden Automaten und Steuerungsproblemen

Damit mit einem numerisches Verfahren ein optimaler Steuerungsprozess berechnet werden kann, müssen in der mathematischen Problemformulierung alle Lösungskandidaten enthalten sein. Um dies zu erreichen wird angenommen, dass das Problem von Zeno nicht auftritt und Steuerungen existieren, mit denen alle Systemzustände erreicht werden können.

Charakterisierung der Problemphasen

Abbildung 3.1 auf der nächsten Seite zeigt ein Beispiel eines hybriden Automaten. Im Bildbereich I ist eine Anfangsbedingung dargestellt. Hier wird der Zustand festgelegt, in dem sich das System zur Zeit t_0 befindet. Diese Bedingung ist gleichbedeutend mit der Anfangsbedingung (2.5) auf Seite 17 für die Zeit t_0. Nachdem die Anfangswerte gesetzt sind, befindet sich das System in dem ersten Knoten bzw. der ersten Phase. Diesem Knoten ist durch $flow(1)$ eine Dynamik zugeordnet, nach der sich das System unter Berücksichtigung der durch $inv(1)$ zugeordneten Nebenbedingungen kontinuierlich weiterentwickelt, bis die durch $jump(12)$ zugeordneten Bedingungen erfüllt sind und den Sprung in den zweiten Knoten einleiten. In der Optimalsteuerung entspricht die durch $jump(12)$ zugeordnete Bedingung den inneren Punktbedingungen (2.5) auf Seite 17 zu den Zeitpunkten t_i, in diesem speziellen Fall t_1. Damit ist die erste Problemphase abgeschlossen. Die Systemvariablen bleiben beim Übergang kontinuierlich, es sei denn sie werden durch die Translationsbedingungen (2.6) auf Seite 17, die durch $event(12)$ gegeben werden, verändert.

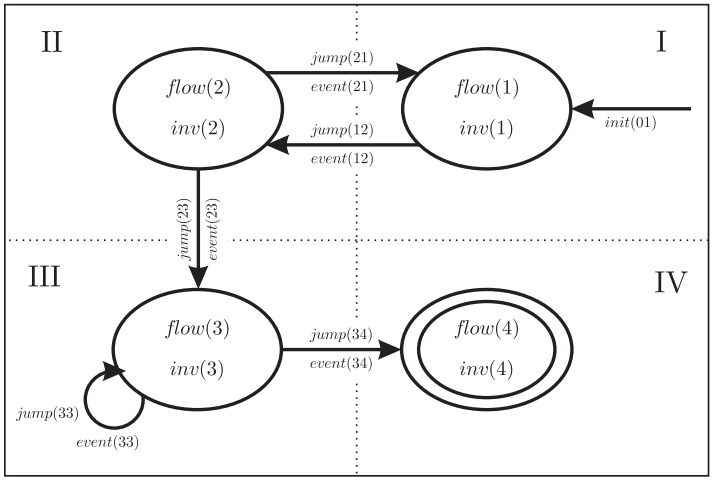


Abbildung 3.1: Beispiel eines hybriden Automaten

Jeder Knoten bildet mit seinen durch $init(01)$ oder $event(ij)$ und den durch Stetigkeitsbedingungen gegebenen Anfangswerten, dem differentialalgebraischen System, gegeben durch $flow(j)$ und $inv(j)$, und seiner Endbedingung, die er durch $jump(jk)$ erhält, den zulässigen Bereich eines einphasigen kontinuierlichen Optimalsteuerungsproblems, genauer gesagt den einer Zeitspanne $[t_i, t_{i+1}]$ des gemischt diskret-kontinuierlichen Problems (2.1) bis (2.5) auf Seite 17, innerhalb der die diskreten Steuerungen konstant sind. Damit ist eine Phase durch die eingehende Kante $\underset{ij}{\longrightarrow}$, den Systemzustand $[j]$ und die ausgehende Kante $\underset{jk}{\longrightarrow}$ eindeutig bestimmt.

Die Knoten werden entsprechend Abbildung 3.1 durchwandert, bis ein Endzustand, wie im Quadranten IV abgebildet, erreicht wird, und bilden dabei jeweils die endlich vielen Phasen eines zulässigen Steuerprozesses des Optimalsteuerungsproblems (2.1) bis (2.7) auf Seite 17.

Bemerkung 10 (zyklische Randbedingung).
Falls keine Anfangsbedingung und kein Endzustand vorhanden sind, und der Automat einen geschlossenen Kreis bildet, handelt es sich um ein periodisches Optimalsteuerungsproblem, bei dem End- und Anfangszustand miteinander verknüpft sind. Hier muss ein Zustand als Anfangszustand definiert und durch eine zyklische Randbedingung mit seinem Vorgänger verknüpft werden.

Automaten mit Kosten

Der letzte Abschnitt hat gezeigt, dass hybride Automaten, die im Hinblick auf die diskreten Entscheidungen deterministisch sind, und deren Menge zulässiger Steuerprozesse nicht leer ist, direkt mit den Nebenbedingungen eines mehrphasigen Optimalsteuerungsproblems in Verbindung gebracht werden können. Um einen direkten Zusammenhang mit Optimalsteuerungsproblemen herzustellen, wird die Definition 4 auf Seite 37 erweitert zu

Definition 5 (Hybrider Automat mit Kosten)**.**
Ein hybrider Automat $H = (V, E, \mathbb{X}, \mathbb{U}, init, inv, flow, jump, event)$ mit

- einer Abbildung $cost_p$, die den Systemzuständen *Laufkosten* zuordnet, und

- einer Abbildung $cost_t$, die den Systemschaltungen *Schaltkosten* zuordnet,

wird hybrider Automat mit Kosten $H_c = (V, E, \mathbb{X}, \mathbb{U}, init, inv, flow, cost_p, jump, event, cost_t)$ genannt.

Damit ist der hybride Automat mit einer Zielfunktion ausgestattet (vgl. auch (BL04)). Die Gesamtkosten eines Laufs

$$\xrightarrow[01]{} [1] \xrightarrow[12]{} [2] \xrightarrow[23]{} [3] \xrightarrow[34]{} \ldots \xrightarrow[(n-2)(n-1)]{} [(n-1)] \xrightarrow[(n-1)n]{} [n]$$

ergeben sich dann aus der Summe der Lauf- und Schaltkosten

$$\sum_{i=0}^{n-1} (cost_p((i+1)) + cost_t(i(i+1)))\,.$$

Diese können direkt auf die Optimalsteuerung übertragen werden, und es ergibt sich (2.1) auf Seite 17.

Durch diese Erweiterung kann bei der Untersuchung des Automaten die Frage der Optimierung der Kosten aufgenommen werden. Die optimalen Kosten eines einzelnen Systemzustands hängen zum einen von dessen kontinuierlicher Steuerung ab, zum anderen auch von den Anfangs- und Endwerten, die sich aus den Übergängen zu benachbarten Systemzuständen ergeben. Die minimalen Kosten eines Systemzustands können nur dann für sich berechnet werden, falls den ein- und ausgehenden Kanten Ereignisse zugeordnet werden, die die Systemzustände entkoppeln.

Mögliche Folgezustände eines Knotens

Werden die Steuerungen als noch unbekannt angenommen, so ist für den optimalen Steuerprozess nicht klar, welcher Systemzustand folgt, falls wie im zweiten Quadranten von Abbildung 3.1 auf der vorherigen Seite dargestellt, Übergänge zu mehreren Folgezuständen möglich sind. In diesem Beispiel kann entweder in den ersten Systemzustand zurück- oder in den Systemzustand drei vorgesprungen werden, je nachdem, welche der durch $jump(21)$ und $jump(23)$ gesetzten Sprungbedingungen zuerst aktiv wird. Da es sich um nur zwei Alternativen handelt, bieten sich verschiedene Ansätze zur Formulierung eines Optimalsteuerungsproblems an, in dem alle möglichen Abläufe des Automaten berücksichtigt sind. Ein Ansatz ist, den Wechsel zwischen Zustand eins und zwei n mal zu wiederholen und dann nach drei überzugehen:

$$\xrightarrow[01]{} [1] \xrightarrow[12]{} [2] \xrightarrow[21]{} [1] \xrightarrow[12]{} [2] \xrightarrow[21]{} \ldots \xrightarrow[12]{} [2] \xrightarrow[23]{}$$

Dieser Ansatz ist sinnvoll, falls sich die ‚unnötigen' Phasen nicht auf die Lösung des Problems auswirken, ansonsten muss die Anzahl gegebenenfalls reduziert oder erhöht werden, bis die optimale Anzahl gefunden ist. Die Komplexität der einzelnen Phasen bleibt hier gering, die Anzahl hängt von einer guten Schätzung ab.

Eine weitere Variante ist es, beide Möglichkeiten als Folgezustand zuzulassen, die dann durch Entscheidungs- bzw. Schaltvariablen gewählt werden, wie in Abschnitt 3.2 auf Seite 44 genauer beschrieben wird.

$$\xrightarrow[01]{} [1] \rightarrow \left\{ \begin{array}{lll} \xrightarrow[12]{} [2] \xrightarrow[21]{} & \vee \\ \xrightarrow[12]{} [2] \xrightarrow[23]{} & \end{array} \right\} \rightarrow \left\{ \begin{array}{lll} \xrightarrow[21]{} [1] \xrightarrow[12]{} & \vee \\ \xrightarrow[23]{} [3] \xrightarrow[33]{} & \vee \\ \xrightarrow[23]{} [3] \xrightarrow[34]{} & \end{array} \right\} \rightarrow \left\{ \begin{array}{lll} \xrightarrow[12]{} [2] \xrightarrow[21]{} & \vee \\ \xrightarrow[12]{} [2] \xrightarrow[23]{} & \vee \\ \xrightarrow[33]{} [3] \xrightarrow[33]{} & \vee \\ \xrightarrow[33]{} [3] \xrightarrow[34]{} & \vee \\ \xrightarrow[34]{} [4] & \end{array} \right\} \rightarrow \ldots$$

$$\ldots \rightarrow \left\{ \begin{array}{lll} \xrightarrow[21]{} [1] \xrightarrow[12]{} & \vee \\ \xrightarrow[23]{} [3] \xrightarrow[33]{} & \vee \\ \xrightarrow[23]{} [3] \xrightarrow[34]{} & \vee \\ \xrightarrow[33]{} [3] \xrightarrow[33]{} & \vee \\ \xrightarrow[33]{} [3] \xrightarrow[34]{} & \vee \\ \xrightarrow[34]{} [4] & \vee \\ \text{Ende} & \end{array} \right\} \rightarrow \left\{ \begin{array}{lll} \xrightarrow[12]{} [2] \xrightarrow[23]{} & \vee \\ \xrightarrow[33]{} [3] \xrightarrow[33]{} & \vee \\ \xrightarrow[33]{} [3] \xrightarrow[34]{} & \vee \\ \xrightarrow[34]{} [4] & \vee \\ \text{Ende} & \end{array} \right\} \rightarrow \left\{ \begin{array}{lll} \xrightarrow[23]{} [3] \xrightarrow[34]{} & \vee \\ \xrightarrow[33]{} [3] \xrightarrow[34]{} & \vee \\ \xrightarrow[34]{} [4] & \vee \\ \text{Ende} & \end{array} \right\} \rightarrow \left\{ \begin{array}{lll} \xrightarrow[34]{} [4] & \vee \\ \text{Ende} & \end{array} \right\} \rightarrow$$

Das System geht in Zustand zwei über und hat nun die Möglichkeit, in Zustand eins oder drei zu wechseln. Anschließend kann es in Zustand zwei, drei oder vier übergehen, je nachdem in welchem es zuvor war. Im nächsten Schritt sind schon vier Zustände möglich. Zum einen die Zustände eins, drei oder vier, die auf Zustand zwei und drei folgen können, zum anderen der virtuelle Zustand ‚Ende', da es keinen Zustand nach vier gibt.

Eigentlich ist das Problem bereits gelöst, wenn der vierte Zustand abgeschlossen ist, aber da vorab nicht bekannt ist, welche Anzahl von Phasen zu einer optimalen Steuerung führt, enthält die Problemformulierung meist mehr als die benötigten Phasen. Für den Zustand ‚Ende'muss ein im Sinne des Optimalsteuerungsproblems *neutraler Systemzustand* definiert werden:

Definition 6 (Neutraler Systemzustand)**.**
Ein Systemzustand heißt neutral, falls sich in ihm weder die Variablen des Systems noch die Systemzeit selbst verändern und er keine Kosten verursacht.

Im Folgenden wird der neutrale Systemzustand mit $null$ bezeichnet und trägt die Nummer 0, falls die Systemzustände nummeriert sind. Da er die Variablen des Systems nicht verändert, muss $flow(0)$ Gleichungen $\dot{\boldsymbol{x}} = 0$ zuordnen, und $inv(0)$ darf keine Bedingungen hinzufügen, die aktiv werden könnten. Wegen der Forderung unveränderlicher Zeit wird eine Stoppuhr in allen Zuständen des Automaten verwendet, wobei die reale Zeit durch die der Stoppuhr ersetzt wird. Sowie der neutrale Systemzustand erreicht wird, wird die Stoppuhr angehalten. Der neutrale Zustand wird nach Verstreichen einer gewissen vorgegebenen Zeitspanne wieder verlassen.

Eine weitere Möglichkeit, die Phasen eines Optimalsteuerungsproblems für einen Automaten festzulegen, basiert darauf, ihn zunächst zu zerlegen. Dazu wird eine Kantenmenge $\hat{E} \subset E$ des Automaten gewählt, deren Sprungbedingungen paarweise verschieden sind. Alle Kanten dieser Menge

werden über Knoten des neutralen Zustands umgeleitet, d. h. die Anfangszustände aller Kanten aus $\hat{E}$ werden mit dem neutralen Zustand verbunden, der wiederum mit den Endzuständen aller Kanten aus $\hat{E}$ verbunden wird. Die Sprungbedingungen müssen paarweise verschieden sein, damit der ursprüngliche Ablauf des Automaten erhalten bleibt. In Abbildung 3.2 ist diese Problematik dargestellt. Gegebenenfalls können die Sprungbedingungen geeignet angepasst werden.

Besonders geeignet zur Umleitung sind Schnitte und vor allem *Brücken* (siehe Definition 15 auf Seite 127), die den Graphen zeitlich trennen. Es wird noch einmal der Automat aus Abbildung 3.1 auf Seite 39 betrachtet. Dieser besitzt die Kanten 23 und 34 als Brücken. Nun wird jeweils die Anzahl der Phasen vom Start bis zur ersten Brücke, die zwischen erster und zweiter Brücke und die bis zum Ende bestimmt oder geschätzt. Hinter den Brücken wird der neutrale Zustand eingefügt. Für die Modellierung werden nur die möglichen Folgezustände bis zu dem neutralen Zustand nach der nächsten Brücke in Betracht gezogen.

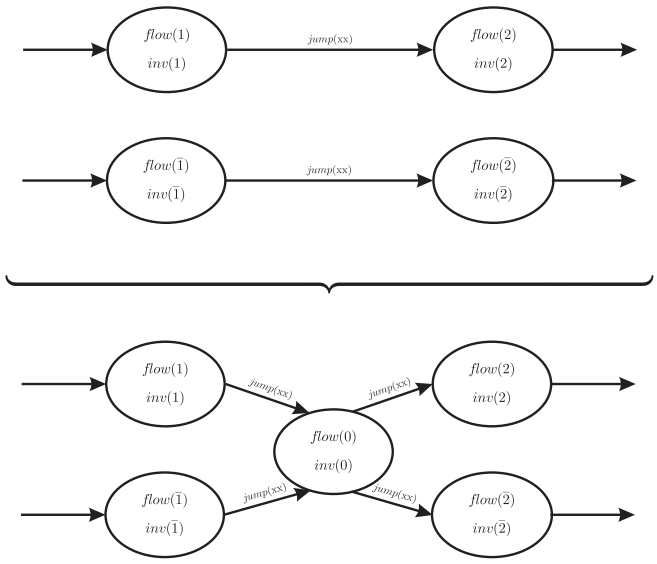


Abbildung 3.2: Mehrdeutige Umleitung bei Verwendung des neutralen Zustands

$$\xrightarrow[01]{}[1]\left\{\begin{array}{lll}\xrightarrow[12]{}[2]\xrightarrow[20]{} & \vee\\ \xrightarrow[12]{}[2]\xrightarrow[20]{} & \end{array}\right\}\rightarrow\left\{\begin{array}{lll}\xrightarrow[20]{}[1]\xrightarrow[00]{} & \vee\\ \xrightarrow[20]{}[0]\xrightarrow[00]{} & \end{array}\right\}\rightarrow\left\{\begin{array}{lll}\xrightarrow[12]{}[2]\xrightarrow[21]{} & \vee\\ \xrightarrow[12]{}[2]\xrightarrow[20]{} & \vee\\ \xrightarrow[00]{}[0]\xrightarrow[00]{} & \end{array}\right\}\rightarrow\dots$$

$$\rightarrow\left\{\begin{array}{lll}\xrightarrow[12]{}[2]\xrightarrow[20]{} & \vee\\ \xrightarrow[00]{}[0]\xrightarrow[00]{} & \end{array}\right\}\rightarrow\left\{\begin{array}{lll}\xrightarrow[20]{}[0]\xrightarrow[03]{} & \vee\\ \xrightarrow[00]{}[0]\xrightarrow[03]{} & \end{array}\right\}\rightarrow\left\{\begin{array}{lll}\xrightarrow[03]{}[3]\xrightarrow[33]{} & \vee\\ \xrightarrow[03]{}[3]\xrightarrow[30]{} & \end{array}\right\}\rightarrow\left\{\begin{array}{lll}\xrightarrow[33]{}[3]\xrightarrow[33]{} & \vee\\ \xrightarrow[33]{}[3]\xrightarrow[30]{} & \vee\\ \xrightarrow[30]{}[0]\xrightarrow[00]{} & \end{array}\right\}\rightarrow$$

$$\dots\rightarrow\left\{\begin{array}{lll}\xrightarrow[33]{}[3]\xrightarrow[33]{} & \vee\\ \xrightarrow[33]{}[3]\xrightarrow[30]{} & \vee\\ \xrightarrow[30]{}[0]\xrightarrow[00]{} & \vee\\ \xrightarrow[00]{}[0]\xrightarrow[00]{} & \end{array}\right\}\rightarrow\dots\rightarrow\left\{\begin{array}{lll}\xrightarrow[33]{}[3]\xrightarrow[34]{} & \vee\\ \xrightarrow[30]{}[0]\xrightarrow[04]{} & \vee\\ \xrightarrow[00]{}[0]\xrightarrow[04]{} & \end{array}\right\}\rightarrow\left\{\begin{array}{lll}\xrightarrow[34]{}[4] & \vee\\ \xrightarrow[04]{}[4] & \end{array}\right\}$$

Der neutrale Systemzustand zerlegt das Problem in mehrere Teilstücke. Zunächst wächst die Anzahl der Möglichkeiten, je weiter in die Zukunft gedacht wird. Dann sorgt der neutrale Zustand dafür, dass sich die Möglichkeiten wieder reduzieren und danach erneut anwachsen. Durch diese Vorgehensweise ergeben sich im Optimalsteuerungsproblem Phasen mit weniger komplexen Funktionen, da weniger parallele Möglichkeiten beachtet werden müssen.

Parallele Abläufe

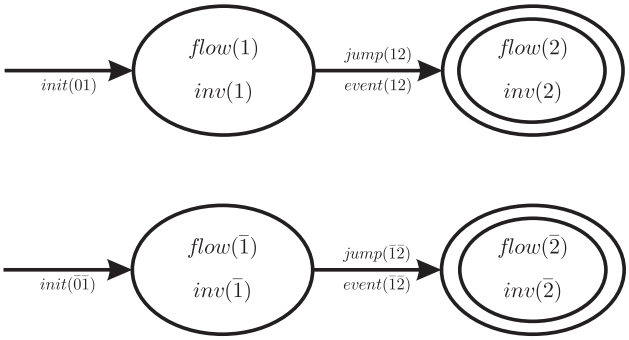


Abbildung 3.3: Hybrider Automat paralleler Abläufe

Bisher wurden zukünftig mögliche Systemzustände betrachtet. Bei technischen Anwendungen laufen häufig mehrere Prozesse parallel ab. Dies können z. B. Flugzeuge im Landeanflug sein. Jedes der Flugzeuge kann für sich durch einen hybriden Automaten dargestellt werden. Abbildung 3.3 zeigt zwei hybride Automaten, die parallel ablaufen sollen. Zunächst wird angenommen dass die Automaten synchronisiert sind. Beide Automaten besitzen eine Anfangsbedingung und gehen in ihren ersten Zustand über, in dem sie sich weiterentwickeln, bis einer von beiden die erste Sprungbedingung erreicht. Hier stellt sich die Frage, welcher der beiden zuerst in den zweiten Zustand übergeht.

$$\left\{\begin{array}{ll} \underset{0(1\wedge\bar{1})}{\longrightarrow}\,[(1\wedge\bar{1})]\ \underset{(1\wedge\bar{1})(2\wedge\bar{1})}{\longrightarrow} & \vee \\ \underset{0(1\wedge\bar{1})}{\longrightarrow}\,[(1\wedge\bar{1})]\ \underset{(1\wedge\bar{1})(1\wedge\bar{2})}{\longrightarrow} & \vee \\ \underset{0(1\wedge\bar{1})}{\longrightarrow}\,[(1\wedge\bar{1})]\ \underset{(1\wedge\bar{1})(2\wedge\bar{2})}{\longrightarrow} & \end{array}\right\} \rightarrow \left\{\begin{array}{ll} \underset{(1\wedge\bar{1})(2\wedge\bar{1})}{\longrightarrow}\,[(2\wedge\bar{1})]\ \underset{(2\wedge\bar{1})\bar{1}}{\longrightarrow} & \vee \\ \underset{(1\wedge\bar{1})(2\wedge\bar{1})}{\longrightarrow}\,[(2\wedge\bar{1})]\ \underset{(2\wedge\bar{1})2\wedge\bar{2}}{\longrightarrow} & \vee \\ \underset{(1\wedge\bar{1})(1\wedge\bar{2})}{\longrightarrow}\,[(1\wedge\bar{2})]\ \underset{(1\wedge\bar{2})1}{\longrightarrow} & \vee \\ \underset{(1\wedge\bar{1})(1\wedge\bar{2})}{\longrightarrow}\,[(1\wedge\bar{2})]\ \underset{(1\wedge\bar{2})2\wedge\bar{2}}{\longrightarrow} & \vee \\ \underset{(1\wedge\bar{1})(2\wedge\bar{2})}{\longrightarrow}\,[(2\wedge\bar{2})]\ \underset{(2\wedge\bar{2})\bar{2}}{\longrightarrow} & \vee \\ \underset{(1\wedge\bar{1})(2\wedge\bar{2})}{\longrightarrow}\,[(2\wedge\bar{2})]\ \underset{(2\wedge\bar{2})2}{\longrightarrow} & \vee \\ \underset{(1\wedge\bar{1})(2\wedge\bar{2})}{\longrightarrow}\,[(2\wedge\bar{2})] & \end{array}\right\} \rightarrow$$

$$\rightarrow \left\{\begin{array}{ll} \underset{(2\wedge\bar{1})\bar{1}}{\longrightarrow}\,[\bar{1}]\ \underset{\bar{1}\bar{2}}{\longrightarrow} & \vee \\ \underset{(2\wedge\bar{1})2\wedge\bar{2}}{\longrightarrow}\,[2\wedge\bar{2}] & \vee \\ \underset{(1\wedge\bar{2})1}{\longrightarrow}\,[1]\ \underset{12}{\longrightarrow} & \vee \\ \underset{(1\wedge\bar{2})2\wedge\bar{2}}{\longrightarrow}\,[2\wedge\bar{2}] & \vee \\ \underset{(2\wedge\bar{2})\bar{2}}{\longrightarrow}\,[\bar{2}] & \vee \\ \underset{(2\wedge\bar{2})2}{\longrightarrow}\,[2] & \vee \\ \text{Ende} & \end{array}\right\} \rightarrow \left\{\begin{array}{ll} \underset{\bar{1}\bar{2}}{\longrightarrow}\,[\bar{2}] & \vee \\ \underset{12}{\longrightarrow}\,[2] & \vee \\ \text{Ende} & \end{array}\right\}$$

Trotz der wenigen Systemzustände jedes einzelnen Automaten gibt es durch den parallelen Ablauf

beider eine Vielzahl von möglichen Systemzuständen in der Zukunft. Parallele Abläufe können unter Verwendung der neuen Systemzustände $(1 \wedge \bar{1})$, $(1 \wedge \bar{2})$, etc., auch in einem einzelnen Automaten dargestellt werden.

Zusammenfassung

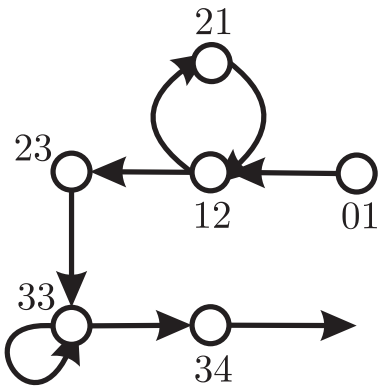


Abbildung 3.4: Phasengraph eines Automaten

Die unterschiedlichen Phasen, die aus dem hybriden Automaten hervorgehen, entsprechen den Verbindungen zwischen den ein- und ausgehenden Kanten innerhalb der Knoten. Eine interessante weitere Darstellungsvariante ergibt sich, wenn die Kanten des Automaten als Knoten dargestellt werden, und die möglichen Phasen als Kanten. In diesem Fall sind die kontinuierlichen Veränderungen des Systems in den Kanten und die Schaltungen in den Knoten. Abbildung 3.4 zeigt den *Phasengraphen* zu dem Automaten aus Abbildung 3.1 auf Seite 39.

Definition 7 (Phasengraph)**.**
Wir bezeichnen einen Graphen $G_P(V_P, E_P)$ als Phasengraph eines Automaten, wobei die Menge der Knoten $V_P := E$ der Menge der Kanten des Automaten entspricht, und $E_P = V_P \times V_P$ die Kanten zwischen diesen neuen Knoten sind.

Momentan liegt das Optimalsteuerungsproblem durch logische Ausdrücke beschrieben vor. Verschiedene Phasen, die zu einem Zeitpunkt vorliegen können, führen zu *Disjunktionen*, parallele Abläufe zu *Konjunktionen* und abhängige Übergänge zu *Implikationen*. Für jede in der Problemstellung mögliche Phase wird eine diskrete Variable eingeführt, um diese eindeutig zu identifizieren und deren Abhängigkeiten formal beschreiben zu können. Gewöhnlich werden binäre Variablen zum ‚An- und Ausschalten' von Termen und ganze Zahlen zum Zählen verwendet.

Oft unterscheiden sich die Phasen nicht komplett, sondern nur in wenigen der Bedingungen, was in der folgenden Formulierung berücksichtigt werden kann.

3.2 Binäre Variablen und logische Zusammenhänge

Ist ein System einmal in Form eines hybriden Automaten dargestellt, können die verschiedenen möglichen Systemabläufe leicht in Form von Relationen beschrieben werden. In diesem Abschnitt wird nun erläutert, wie diese Relationen in Gleichungen und Ungleichungen übersetzt werden können, so dass ein streng formales, konsistentes gemischt diskret-kontinuierliches Optimalsteuerungsproblem vorliegt, das numerisch gelöst werden kann. Der Abschnitt ist an (Kal02) angelehnt,

wo auch weiterführende Informationen gefunden werden können. In Zusammenhang mit hybriden linearen und zeitdiskreten Optimalsteuerungsproblemen sei auf Arbeiten wie (BM99; BG03) verwiesen.

Im ersten Abschnitt wird darauf eingegangen, wie die verschiedenen diskreten Variablentypen durch binäre Variablen vereinheitlicht werden können. Unter Verwendung dieser binären Variablen werden dann mögliche Formalismen beschrieben, um logische Ausdrücke durch Gleichungen auszudrücken. Der letzte Abschnitt beschäftigt sich damit, wie die Bedingungen durch die binären Variablen an- und abgeschaltet werden können.

Diskrete durch binäre Variablen oder logische Ausdrücke durch arithmetische Nebenbedingungen zu ersetzen, ist nicht immer eindeutig. Da viele numerische Lösungsverfahren zur Lösung von Problemen mit diskreten Variablen zwischenzeitlich den zulässigen Bereich vergrößern, auch relaxieren genannt, indem sie statt diskreter kontinuierliche Variablen zulassen, sollte im Zweifelsfall anhand der Relaxierung entschieden werden, welche Nebenbedingungen besser sind. Wünschenswert ist, dass der relaxierte Bereich konvex, möglichst klein und einfach strukturiert ist.

3.2.1 Variablen mit diskreten Wertebereichen

Diskrete Variablen resultieren aus den Systemschaltungen, sie können die Anzahl von Bauteilen angeben oder sie können aus der Auswertung von kontinuierlichen Funktionen an diskreten Stellen stammen.

Satz 3. *Sei $\tilde{z} \in \tilde{\mathbb{Z}}$, wobei $\tilde{\mathbb{Z}} \subset \mathbb{R}^n$ eine höchstens abzählbare Menge diskreter Werte ist. Dann existiert eine Teilmenge $\hat{\mathbb{Z}} \subset \mathbb{Z}$ und eine Bijektion von $\tilde{\mathbb{Z}}$ auf $\hat{\mathbb{Z}}$.*

Die Werte einer solchen Menge können also nummeriert bzw. durch ganze Zahlen identifiziert werden. Dabei ist zu beachten, dass eine Nummerierung auch eine Sortierung angeben kann. Die Auswahl eines diskreten Wertes $\tilde{z} \in \tilde{\mathbb{Z}}$ mit $|\mathbb{Z}| = n$ kann nun durch binäre Variablen $z_i \in \{0, 1\}$ durch

$$\tilde{z} \;=\; \sum_{i=1}^{n} z_i \tilde{z}_i$$

realisiert werden. Hinzu kommt, dass die binären Variablen damit eine so genannte *SOS-1-Menge* bilden. SOS-1-Mengen (engl. *special ordered set of type 1*) sind geordnete Mengen beliebiger Variablen, von denen nur eine Variable ungleich null sein darf. Dies kann hier durch die Konvexitätsbedingung

$$1 \;=\; \sum_{i=1}^{n} z_i$$

gefordert werden. Eine weitere Möglichkeit, Werte dieser Zahlenmenge durch binäre Variablen zu identifizieren, ist, jeweils nur den Differenzbetrag zum Vorgänger zu addieren:

$$\tilde{z} \;=\; \tilde{z}_0 + \sum_{i=2}^{n} z_i(\tilde{z}_i - \tilde{z}_{i-1}), \quad z_i \;\leq\; z_{i-1}, \quad \forall i = 2, \ldots, n.$$

Bei ganzen Zahlen kann auch deren Binärdarstellung verwendet werden, wie der folgende Satz zeigt.

Satz 4. *Eine ganze Zahl $\hat{z}$ kann über die Summe*

$$\hat{z} = \sum_{i=0}^{m} 2^i z_i$$

mit $m \in \mathbb{N}_0$ und $z_i \in \{0, 1\}$ eindeutig dargestellt werden, wobei

$$m = 1 + \text{rnd}_-(\frac{\log(n)}{\log(2)})$$

ist und rnd$_-$ *Abrunden bezeichnet.*

3.2.2 Relationen zweier und mehrerer Argumente

Nachdem alle diskreten Variablen die im Problem vorkommen durch binäre Variablen ausgedrückt sind, lassen sich die logischen Zusammenhänge durch Gleichungen und Ungleichungen beschreiben. Damit können mögliche Abläufe eines hybriden Automaten durch Formeln dargestellt werden. Der Vollständigkeit halber sind die Rechenregeln mit logischen Ausdrücken in Abschnitt A.1 auf Seite 125 kurz zusammengestellt.

Relationen und lineare Gleichungen

Aus dem Modell, das unter Verwendung hybrider Automaten erstellt wurde, ergeben sich verschiedene logische Aussagen über die einzelnen Phasen. Falls nur Relationen für zwei Aussagen vorliegen, ergeben sich die resultierenden Gleichungen und Ungleichungen aus

Satz 5. *Seien A und B Aussagen, denen die Variablen $z_A \in \{0, 1\}$ und $z_B \in \{0, 1\}$ zugeordnet sind, so dass $z_A = 0$, falls Aussage A falsch, und $z_A = 1$, falls Aussage A wahr ist. Analoges gelte für z_B. Logische Ausdrücke lassen sich wie folgt durch Gleichungen und Ungleichungen ausdrücken:*

$$\begin{array}{lcll} A & \wedge & B & \Leftrightarrow \quad z_A + z_B = 2 \\ A & \vee & B & \Leftrightarrow \quad z_A + z_B \geq 1 \\ \overline{A} & & & \Leftrightarrow \quad 1 - z_A = 1 \\ A & \rightarrow & B & \Leftrightarrow \quad z_A - z_B \leq 0 \\ A & \leftrightarrow & B & \Leftrightarrow \quad z_A - z_B = 0 \\ A & \oplus & B & \Leftrightarrow \quad z_A + z_B = 1. \end{array}$$

Die Gültigkeit der *Äquivalenz* dieser Relationen zu den Gleichungen und Ungleichungen kann direkt aus der Wahrheitstabelle Tabelle 3.1 abgelesen werden. W bezeichnet dabei wahre und F falsche Aussagen.

Tabelle 3.1: Wahrheitstabelle mit ersetzenden Gleichungen

		Konjunktion	Disjunktion	Negation	Implikation	Äquivalenz	Antivalenz
A	B	$A \wedge B$	$A \vee B$	$\overline{A}$	$A \rightarrow B$ $\overline{A} \vee B$	$A \leftrightarrow B$ $(A \wedge B) \vee (\overline{A} \wedge \overline{B})$	$A \oplus B$ $(A \wedge \overline{B}) \vee (\overline{A} \wedge B)$
W	W	W	W	F	W	W	F
W	F	F	W	F	F	F	W
F	W	F	W	W	W	F	W
F	F	F	F	W	W	W	F
z_A	z_B	$z_A + z_B = 2$	$z_A + z_B \geq 1$	$z_A = 0$	$z_A - z_B \leq 0$	$z_A - z_B = 0$	$z_A + z_B = 1$
1	1	W	W	F	W	W	F
1	0	F	W	F	F	F	W
0	1	F	W	W	W	F	W
0	0	F	F	W	W	W	F

Wird ein Wert in einer Aussage A verneint wie $\overline{A}$, so ist die dazugehörige Variable $z_{\overline{A}} = 1 - z_A$. Damit lassen sich leicht die Aussagen

$$\overline{A \wedge B}, \quad \overline{(A \vee B)}$$

die nach den Gesetzen von DeMorgan (A.1) auf Seite 125 identisch sind mit

$$\begin{aligned} \overline{A \wedge B} &= \overline{A} \vee \overline{B} \\ \overline{A \vee B} &= \overline{A} \wedge \overline{B} \end{aligned}$$

in arithmetische Ausdrücke umwandeln

$$\begin{aligned} \overline{A} \vee \overline{B} &\Leftrightarrow 1 - z_A + 1 - z_B = 2 \Leftrightarrow z_A = 0, \quad z_B = 0 \\ \overline{A} \wedge \overline{B} &\Leftrightarrow 1 - z_A + 1 - z_B \geq 1 \Leftrightarrow z_A + z_B \leq 1. \end{aligned}$$

Einige Aussagen aus Satz 5 auf der vorherigen Seite lassen sich auch auf mehrere Argumente erweitern

$$\begin{aligned} \textstyle\bigwedge_{i=1}^{n} A_i &\Leftrightarrow \textstyle\sum_{i=1}^{n} z_{A_i} = n \\ \textstyle\bigvee_{i=1}^{n} A_i &\Leftrightarrow \textstyle\sum_{i=1}^{n} z_{A_i} \geq 1. \end{aligned}$$

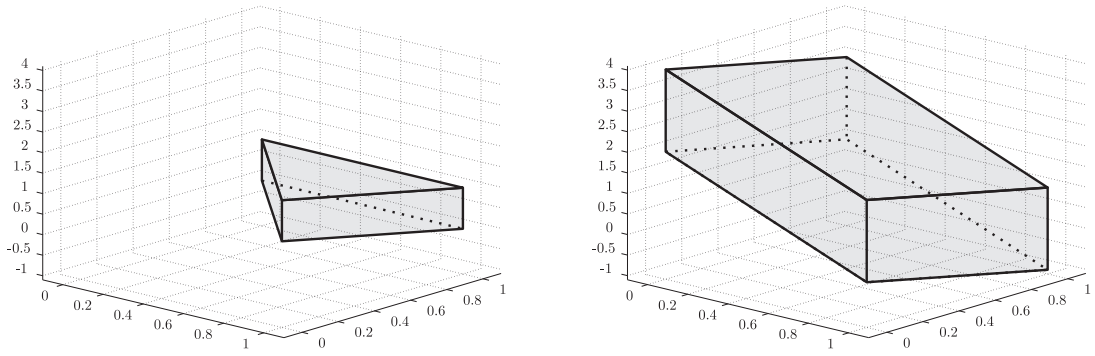
Umformungen

Wie bereits in der Einleitung dieses Abschnitts erwähnt, können oft gleichwertige Bedingungen für die binären Variablen aufgestellt werden, die aber im relaxierten Fall unterschiedliche zulässige Bereiche definieren. Dies soll anhand des folgenden Beispiels aus (Kal02) verdeutlicht werden. Betrachtet wird der logische Ausdruck $A \vee (B \wedge C)$. Dieser ist gleichbedeutend mit $(A \vee B) \wedge (A \vee C)$ und genau dann wahr, wenn

$$z_A + z_B \geq 1, \quad z_B + z_C \geq 1, \tag{3.1}$$

was gleichbedeutend mit

$$2z_A + z_B + z_C \geq 2 \tag{3.2}$$



(a) zu Gleichungen (3.1) (b) zu Gleichung (3.2)

Abbildung 3.5: Relaxierung der Nebenbedingungen bei unterschiedlicher Formulierung

ist.

Der zulässige Bereich der relaxierten Bedingungen ist in 3.5 aufgezeichnet. Die Gleichungen (3.1) liefern einen wesentlich kleineren Bereich als (3.2). Die Bedingung ist also *schärfer*, auch wenn sie für diskrete Werte die gleichen zulässigen Punkte ergeben. Es gilt also nicht, dass weniger Nebenbedingungen oder weniger Variablen ein leichter zu lösendes gemischt diskret-kontinuierliches Optimalsteuerungsproblem ergeben. Vielmehr ist ausschlaggebend, wie der dazugehörige zulässige Bereich aussieht.

Da die Bedingungen, die bei hybriden Automaten den Kanten zugeordnet wurden, eventuell selbst logische Ausdrücke enthalten, ist es möglich, dass nichtlineare binäre Gleichungen entstehen. Diese können durch Hinzunahme weiterer binärer Variablen und linearer Bedingungen eliminiert werden. Zunächst werden dazu Potenzen von binären Variablen durch die Variablen selbst ersetzt, da $z^n = z$ für $z \in \{0, 1\}$ gilt. Produkte zweier binärer Variablen sind wiederum binärwertig, und es gilt genau dann $z_a z_b = 1$, falls $[z_a = 1]$ und $[z_b = 1]$, ansonsten gilt $[z_a z_b = 0]$. Eine Variable z_{ab}, die die Nebenbedingungen $z_{ab} \leq z_a$, $z_{ab} \leq z_b$ und $z_{ab} \geq z_a + z_b - 1$ erfüllt, hat dieselben Eigenschaften wie das Produkt $z_a z_b$ und kann es somit ersetzen.

3.2.3 Logische Verknüpfung von Nebenbedingungen

Nachdem die Abfolge und die Auswahl verschiedener Phasen durch die Nebenbedingungen, die aus den logischen Verknüpfungen stammen, durch zulässige Kombinationen von binären Variablen festgelegt sind, müssen nun die Bedingungen, die eine Phase definieren, entsprechend an- und abgeschaltet werden. Falls noch Terme enthalten sind, die in den diskreten Variablen nichtlinear sind, so können diese folgendermaßen umgeformt werden:

Bemerkung 11.
Sei eine Abbildung $a : \mathbb{U} \times \mathbb{W} \to \mathbb{R}^{n_a}$ mit $\mathbb{W} = \{w_1, \ldots, w_n\}$ gegeben, so kann sie als Linearkombination in der Form $\hat{a}(u, \alpha) = \sum_{i=1}^{n} \alpha_i a(u, w_i)$ mit $\alpha_i \in \{0, 1\}$ und $\sum_{i=1}^{n} \alpha_i = 1$ dargestellt werden.

Eine andere Variante, um dies zu erreichen, liefert

Bemerkung 12.
Sei eine Abbildung $a : \mathbb{U} \times \mathbb{W} \rightarrow \mathbb{R}^{n_a}$ mit $\mathbb{W} = \{w_1, \ldots, w_n\}$ gegeben, so kann sie als Linearkombination in der Form $\hat{a}(u, \alpha) = a(u, w_1) + \sum_{i=2}^{n} \alpha_i (a(u, w_i) - a(u, w_{i-1})$ mit $\alpha_i \in \{0, 1\}$ und $\sum_{i=1}^{n} \alpha_i \leq \alpha_{i-1}$ dargestellt werden.

Dynamik

Im zeitlichen Verlauf werden verschiedene Terme in den Gleichungsnebenbedingungen an- und abgeschaltet. Zu den Gleichungsnebenbedingungen gehören auch die Differentialgleichungen, die die Dynamik des Systems beschreiben. Unter Anwendung von Bemerkung 11 oder Bemerkung 12 kann die Dynamik immer als Summe verschiedener Dynamiken

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \tilde{\boldsymbol{w}}(t), \boldsymbol{p}, \boldsymbol{q}, t) = \sum_{i=1}^{m_w} w_i(t) \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \tilde{\boldsymbol{w}}_i(t), \boldsymbol{p}, \boldsymbol{q}, t)$$

mit $\tilde{\boldsymbol{w}} \in \mathbb{W}$ und $|\mathbb{W}| = m_w$, dargestellt werden, wobei alle binären Steuerungen $\boldsymbol{w}$ linear auftauchen und weitere Bedingungen erfüllen müssen. Dies hat den entscheidenden Vorteil, dass sich diskrete Steuerungen, die im relaxierten Fall wie kontinuierliche zu behandeln sind, wegen des Minimumprinzips, wie in Abschnitt 2.3.3 auf Seite 35 erläutert, gegebenenfalls automatisch zu null bzw. eins ergeben können.

Big-M

Eine Standardmethode, um geschaltete Nebenbedingungen zu formulieren, ist die so genannte ‚Big-M-Methode'. Hierfür werden zunächst obere und untere Schranken für die Nebenbedingungen festgelegt: $m \leq h(y) \leq M$. Solange $z = 0$ gilt, wird y durch

$$h(y) \leq (1 - z)M$$

nicht eingeschränkt. Analog gilt dies für die untere Schranke, und es lassen sich die leicht überprüfbaren Zusammenhänge (vgl. (BM99)) zusammenfassen in:

Satz 6. *Es sei* $f : \mathbb{Y} \rightarrow \mathbb{R}$, $m = \min_{y \in \mathbb{Y}} h(y)$, $M = \max_{y \in \mathbb{Y}} h(y)$ *und* $z \in \{0, 1\}$, *dann gelten die*

folgenden Zusammenhänge:

$$[h(y) \leq 0] \quad \vee [z = 1] \quad \Leftrightarrow h(y) \leq zM \tag{3.3}$$

$$[h(y) \leq 0] \quad \wedge [z = 1] \quad \Leftrightarrow h(y) - z \leq -1 + m(1 - z) \tag{3.4}$$

$$\overline{[h(y) \leq 0]} \quad \Leftrightarrow h(y) > 0 \tag{3.5}$$

$$[z = 1] \rightarrow [h(y) \leq 0] \quad \Leftrightarrow h(y) \leq (1 - z)M \tag{3.6}$$

$$[z = 1] \rightarrow [0 \leq h(y)] \quad \Leftrightarrow (1 - z)m \leq h(y) \tag{3.7}$$

$$[z = 1] \rightarrow [h(y) = 0] \quad \Leftrightarrow (1 - z)m \leq h(y) \leq (1 - z)M \tag{3.8}$$

$$[h(y) \leq 0] \rightarrow [z = 1] \quad \Leftrightarrow h(y) > zm \tag{3.9}$$

$$[h(y) \leq 0] \leftrightarrow [z = 1] \quad \Leftrightarrow \begin{cases} h(y) \leq (1 - z)M \\ h(y) > zm \end{cases} \tag{3.10}$$

Die Beziehungen (3.3) bis (3.10) sind einfache logische Bedingungen an Nebenbedingungen. Bei (3.6) bis (3.8) ist die Schaltvariable z der Auslöser dafür, dass der zulässige Bereich eingeschränkt wird. Im Gegensatz dazu schaltet in (3.9) die kontinuierliche Variable y die binäre z auf eins, sowie die Ungleichung $h(y) \leq 0$ erfüllt ist. In Gleichung (3.10) sind Nebenbedingung und binäre Variable direkt gekoppelt.

Im Folgenden wird Bedingung (3.6) genauer betrachtet, da sie sehr oft bei Problemformulierungen vorkommt. Beim Schalten zwischen mehreren disjunkten Gebieten, die durch $h_i(y) \leq 0$ mit $m_i \leq h(y) \leq M_i$, $i = 1, \ldots n_h$ gegeben sind, ergeben sich durch die Big-M-Formulierung die Gleichungen

$$h_i(y) \leq (1 - z_i)M_i, \tag{3.11}$$

$$\sum_{i=1}^{n_h} z_i = 1, \tag{3.12}$$

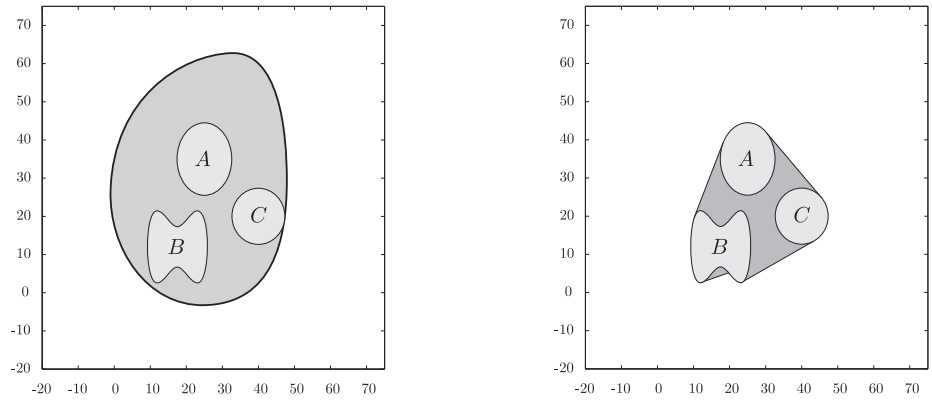
$$z_i \in \{0, 1\}, \quad i = 1, \ldots, n_h. \tag{3.13}$$

Falls es sich bei den Funktionen $h_i(y)$ um konvexe Funktionen handelt, bilden auch die Gleichungen (3.11) bis (3.13) einen konvexen Bereich, der die konvexe Hülle der disjunkten Teilgebiete enthält. Ein Beispiel dazu ist in Abbildung 3.6(a) auf der nächsten Seite zu sehen.

Konvexe Hülle

Eine strengere geschaltete Nebenbedingung als sie der Big-M-Ansatz liefert, ergibt sich aus der konvexen Hülle. Für weitergehende Betrachtungen, sowie Beweise zu den einzelnen Behauptungen sei auf (GL03) und die dortigen Literaturangaben verwiesen.

Betrachtet werden die Nebenbedingungen $h_i(y) \leq 0$, mit $h[0, U] \rightarrow \mathbb{R}$ konvex und $i = 1, \ldots, n_h$, die disjunkte Gebiete definieren wie sie z. B. in Abbildung 3.6(b) auf der nächsten Seite zu sehen



(a) Relaxierung bei Big-M

(b) Relaxierung bei ‚konvexer Hülle‘

Abbildung 3.6: Relaxierung geschalteter, disjunkter Gebiete

sind. Die konvexe Hülle dieser Gebiete ist gegeben durch

$$\sum_{i=1}^{n_h} z_i \xi_i = y, \quad \xi_j \in [0, y_{i,max}], \tag{3.14}$$

$$\sum_{i=1}^{n_h} z_i = 1, \quad z_j \in [0,1], \tag{3.15}$$

$$h_j(\xi_j) \leq 0, \quad j = 1, \ldots, n_h. \tag{3.16}$$

Die Punkte ξ_j liegen wegen der Ungleichungen (3.16) jeweils in einem der disjunkten Gebiete. y stellt durch (3.14) und (3.15) eine Konvexkombination der ξ_j dar. Aufgrund der Konvexität der Funktionen h_j ist damit die Menge aller so gegebenen y konvex. Falls aber die Funktionen h_j nichtkonvex sind, ist im Allgemeinen auch die Relaxierung nichtkonvex, wie auch Abbildung 3.6(b) zeigt.

Um nun die logische Verknüpfung $[z_j = 1] \rightarrow [h_j(\xi) \leq 0]$ zu erhalten, wird sie durch $z_j h_j(y) \leq 0$ formuliert. Der bilineare Term (3.14) kann durch Verwendung von $\xi_i = \frac{\zeta_i}{z_i}$ linearisiert werden. Damit ergibt sich aus (3.14) bis (3.16)

$$\sum_{i=1}^{n_h} \zeta_i = y, \quad \zeta_j \in [0, z_i y_{i,max}], \tag{3.17}$$

$$\sum_{i=1}^{n_h} z_i = 1, \quad z_j \in [0,1], \tag{3.18}$$

$$z_j h_j(\frac{\zeta_i}{z_i}) \leq 0, \quad j = 1, \ldots, n. \tag{3.19}$$

Aufgrund der Bedingungen $0 \leq \zeta \leq z_i y_{i,max}$ ist es offensichtlich, dass ζ mindestens so schnell gegen null strebt wie z, weswegen $\lim_{z \downarrow 0} f(\frac{\zeta}{z}) < \infty$ und $\lim_{z \downarrow 0} z f(\frac{\zeta}{z}) = 0$ zu erwarten sind. Allerdings ist (3.19) in $z_j = 0$ nicht differenzierbar (GL03).

Um eine stetig differenzierbare Funktion zu erhalten, wird (3.19) im Allgemeinen durch

$$(z_j + \epsilon) h_j(\frac{\zeta_i}{z_j + \epsilon})$$

mit z. B. $\epsilon = 10^{-4}$ approximiert. Eine zu kleine Wahl von ϵ kann zu numerischen Problemen führen.

Zusammenfassung

Die Umwandlung der Variablen mit diskreten Wertebereichen in binäre Variablen geht zunächst mit einem Informationsverlust über die Herkunft und Bedeutung der Variablen einher. Diese Informationen sind nun in den Gleichungs- und Ungleichungsnebenbedingungen enthalten, die den zulässigen Bereich für die binären Variablen beschreiben. Die Darstellung dieses zulässigen Bereichs ist nicht eindeutig. Sie sollte so gewählt werden, dass der zulässige Bereich im Falle einer Relaxierung möglichst scharf ist.

Bei der Verknüpfung von Nebenbedingungen ist darauf zu achten, dass nicht zu viele neue Nichtlinearitäten hinzukommen, da diese weitere lokale Minima erzeugen können. Bei geschalteter Dynamik kann eine Linearkombination der Einzelteile dazu führen, dass in Folge des Minimumprinzips kontinuierlich modellierte, binäre Steuerungen automatisch ihre diskreten Werte annehmen.

Gegebenenfalls ist es sinnvoll weitere binäre Variablen und Bedingungen einzuführen, so dass gezielt durch Umschalten dieser Variablen von Null auf Eins auch weitere Variablen schalten. Damit kann die Suche mit einem Branch-and-Bound-Verfahren erheblich beeinflusst werden.

3.3 Das gemischt binär-kontinuierliche Optimalsteuerungsproblem

Das hybride dynamische System wurde durch einen hybriden Automaten dargestellt und anhand der sich daraus ergebenden logischen Zusammenhänge für die möglichen Problemphasen in ein mathematisch formales, nichtlineares, gemischt binär-kontinuierliches Steuerungsproblem transformiert. Da bei dieser Vorgehensweise eine feste Anzahl von Schaltungen der binärwertigen Steuerungen vorgegeben ist, können die Zustände der Schaltung zusammen mit den binären Parametern in einem einzigen Vektor $\boldsymbol{q}_b$ zusammengefasst werden. Spezielle Informationen über die Eigenschaften der ursprünglich nicht zwingend binären Variablen und Parameter müssen bereits bei der Modellierung berücksichtigt und als logische Bedingungen $0 \geq \boldsymbol{L}\boldsymbol{q}_b$ formuliert werden. Zudem können durch weitere binäre Parameter Schaltungsmöglichkeiten geschaffen werden, die im ursprünglichen nicht vorgesehen waren. Damit ist das gemischt binär-kontinuierliche Optimalsteuerungsproblem durch

$$J[\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{q}_b] = \sum_{i=1}^{n_c} \Phi_i(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}_b, t_i) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_b, t) \mathrm{d}\, t \tag{3.20}$$

unter

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_b, t) && (3.21)\\
0 &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_b, t) && (3.22)\\
0 &\geq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_b, t) && (3.23)\\
0 &= \boldsymbol{r}(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}_b, t_i) && (3.24)\\
0 &= \boldsymbol{j}_i(\boldsymbol{x}(\hat{t}_i^-), \boldsymbol{x}(\hat{t}_i^+), \boldsymbol{p}, \boldsymbol{q}_b, \hat{t}_i) && (3.25)\\
0 &\geq \boldsymbol{L}\boldsymbol{q}_b && (3.26)
\end{aligned}$$

mit $\boldsymbol{L} \in \mathbb{R}^{n_L \times n_{q_b}}$ und $\boldsymbol{q}_b \in \{0,1\}^{n_{q_b}}$ gegeben, und es können geeignete numerische Verfahren zu dessen Optimierung entwickelt und eingesetzt werden, wie sie im folgenden Kapitel beschrieben werden.

Kapitel 4

Numerik diskret-kontinuierlicher Optimalsteuerungsprobleme

In diesem Kapitel werden Verfahren beschrieben, die zur Lösung nichtlinearer gemischt diskret-kontinuierlicher Optimalsteuerungsprobleme eingesetzt werden können. Im ersten Abschnitt werden zur Lösung kontinuierlicher Optimalsteuerungsprobleme geeignete Kollokationsverfahren vorgestellt. Der zweite Abschnitt beschäftigt sich mit dem Branch-and-Bound-Suchverfahren, das in einer äußeren Iteration den Raum der diskreten Möglichkeiten absucht und in einer inneren Iteration kontinuierliche Optimalsteuerungsprobleme löst. Im dritten Teil wird schließlich eine weitere Methode vorgestellt, die auf der teilweisen Diskretisierung des Zustandsraums basiert. Dazu werden Familien von kontinuierlichen Optimalsteuerungsproblemen numerisch gelöst und schließlich durch die Lösung eines linearen ganzzahligen Optimierungsproblems zu einer Näherungen des ursprünglichen Problems zusammengesetzt.

4.1 Direkte Kollokation für dynamische Optimierungsprobleme

Im Wesentlichen lassen sich die Verfahren zur numerischen Berechnung von Optimalsteuerungsproblemen in zwei Klassen einteilen, die *indirekten* und die *direkten* Verfahren. Ein Überblick über die verschiedenen Verfahren wird z. B. in (SB92; Bet98) gegeben.

Zu den indirekten Verfahren zählen solche, die expliziten Gebrauch von den notwendigen Bedingungen für die Optimalität eines Steuerprozesses (vgl. Abschnitt 2.3 auf Seite 27) machen. Insgesamt führt diese Herangehensweise auf das Lösen eines *Mehrpunktrandwertproblems*, das numerisch integriert werden kann. Dazu gibt es eine Reihe von Verfahren, die meist auf der Mehrzielmethode beruhen. Ist eine Lösung gefunden, so zeichnet sich diese durch sehr hohe Genauigkeit aus. Dem stehen aber auch einige Nachteile gegenüber.

- Zum Aufstellen der notwendigen Bedingungen ist eine gute Kenntnis der Optimalsteuerungstheorie unabdingbar und erfordert u.a. die explizite Formulierung der sogenannten ad-

jungierten oder Kozustandsdifferentialgleichungen. Diese oft langwierige Vorgehensweise kann durch den Einsatz von symbolischer Formelmanipulation oder durch Techniken der automatischen Differentiation unterstützt werden.

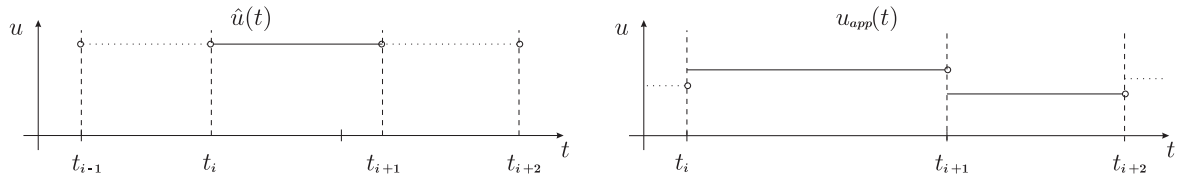
- Um bei der Mehrzielmethode in der Praxis Konvergenz zu erreichen, müssen sehr gute Startschätzungen für die Zustandsvariablen und die adjungierten Variablen vorliegen.
- Zum korrekten Aufstellen des Mehrpunktrandwertproblems ist die Kenntnis der Schaltstruktur notwendig.
- Die Optimalität der Lösung ist gesichert, da die Verfahren die Optimalitätsbedingungen lösen.

Direkte Verfahren basieren auf Ansätzen, das Optimalsteuerungsproblem in ein endlichdimensionales, nichtlineares Optimierungsproblem mit nichtlinearen Gleichungs- und Ungleichungsbedingungen zu transformieren, das dann mit Methoden der nichtlinearen Optimierung gelöst werden kann. Bei *direkten Schießverfahren* ist das dynamische System immer noch in die Nebenbedingungen eingebettet, so dass nach jedem Optimierungsschritt das System simuliert (d.h. numerisch integriert) werden muss. Bei *direkten Kollokationsverfahren* werden die Differentialgleichungen durch Kollokation an ausgewählten Punkten implizit während der Optimierung gelöst. Damit finden Optimierung und Simulation simultan statt. Die Vor- und Nachteile seien hier kurz genannt:

- Um direkte Verfahren anzuwenden, ist kaum Vorwissen über die Optimalsteuerungstheorie notwendig.
- Diese lokal konvergenten Verfahren haben einen relativ großen Konvergenzbereich. Eine Startschätzung der Trajektorien muss also bei weitem nicht so gut sein wie bei den indirekten Verfahren.
- Bei Kollokationsverfahren ist eine Validierung der Optimalität *a posteriori* möglich, da die Adjungierten aus den Lagrangemultiplikatoren des zu lösenden nichtlinearen Optimierungsproblems näherungsweise berechnet werden können (vgl. (SB92; Ret03)).
- Die Genauigkeit der Lösung ist nicht so hoch wie bei indirekten Verfahren, die Lösung kann aber als sehr guter Startwert für solche genommen werden.

Betrachtet wird nun ein Optimalsteuerungsproblem mit fester Anfangs- und Endzeit. Falls das Problem nicht in dieser Form vorliegt, kann es in diese gebracht werden. Weiter sei eine zulässige Wahl der diskreten Variablen und diskreten Steuerungen bereits vorgegeben.

Ist nun im Rahmen der Steuerbeschränkungen eine zulässige kontinuierliche Steuerung gegeben, so kann mit dieser versucht werden, die Systemzustände aus dem differentialalgebraischen System, das aus den Differentialgleichungen, Gleichungs- und Ungleichungsnebenbedingungen besteht, zu berechnen. Abgesehen von etwaigen unbekannten Anfangs- und Endwerten der Zustandsvariablen sind damit die kontinuierlichen Steuerungen die eigentlich gesuchten Unbekannten des Problems.



(a) Basis (b) Approximation

Abbildung 4.1: Stückweise konstante Diskretisierung

4.1.1 Diskretisierung

Steuerdiskretisierung

Da es sich bei den Steuerungen um zeitabhängige Funktionen, also Elemente eines unendlichdimensionalen Raums, handelt, soll dieser Raum zur numerischen Berechnung geeignet durch einen endlichdimensionalen approximiert werden. Die Lösungskandidaten $u(t)$ seien Elemente des Hilbertraums $\mathbb{U}$, und $\mathbb{U}_n \subset \mathbb{U}$ ein endlichdimensionaler Teilraum, so kann eine Approximation $u_{app}(t) \in \mathbb{U}_n$ von $u(t)$ über die endlichdimensionale Basis $\hat{u}_1(t), \ldots, \hat{u}_n(t)$ durch die Parameter $y_{u,1}, \ldots, y_{u,n}$ als

$$u_{app}(t) = \sum_{i=1}^{n} y_{u,i}\hat{u}_i(t)$$

dargestellt werden. In den meisten Fällen wird ein Teilraum $\mathbb{U}_n$ mit stückweise konstanten (Abbildung 4.1(b)), stückweise linearen, stetigen (Abbildung 4.2(b)) oder aber stückweise linearen, unstetigen Funktionen (Abbildung 4.3(b)) gewählt, da diese wenig Parameter benötigen und doch eine ausreichende Approximation darstellen. Die stückweise konstanten Funktionen sind besonders günstig, falls mit Bang-Bang-Verhalten gerechnet werden kann. Ein geeigneter Teilraum $\mathbb{U}_n$ lässt sich durch eine Diskretisierung des Zeitintervalls $[t_0, t_f]$ des Optimalsteuerungsproblems konstruieren.

Für stückweise konstante Approximationen können die Indikatorfunktionen, auch charakteristischen Funktionen genannt,

$$\hat{u}_i(t) = \begin{cases} 1 & \text{falls} \quad t \in [t_i, t_{i+1}), \\ 0 & \text{sonst} \end{cases}$$

verwendet werden (Abbildung 4.1(a)), womit die Parameter dem Wert der Approximation im jeweiligen Intervall entsprechen. Bei dieser Approximation ergeben sich bei einer Zeitdiskretisierung mit $n_{u,\tau}$ Punkten $(n_{u,\tau} - 1)$ Parameter je kontinuierlicher Steuervariable.

Stetige, abschnittsweise lineare Funktionen lassen sich durch die Basis

$$\hat{u}_i(t) = \begin{cases} \frac{t-t_{i-1}}{t_i-t_{i-1}} & \text{falls} \quad t \in [t_{i-1}, t_i), \\ \frac{t_i-t}{t_{i+1}-t_i} & \text{falls} \quad t \in [t_i, t_{i+1}], \\ 0 & \text{sonst} \end{cases}$$

darstellen (vgl. 4.2(a) auf der nächsten Seite), wobei das erste und letzte Intevall gesondert zu be-

trachten sind. Damit ergeben sich bei $n_{u,\tau}$ Diskretisierungspunkten auch $n_{u,\tau}$ Parameter je Steuerung. Diese Parameter zur Darstellung der Approximation sind die Werte von u_{app} auf dem Gitter.

Für eine nichtstetige Variante der abschnittsweise linearen Steuerapproximationen kann eine Basis

$$\hat{u}_{2i}(t) = \begin{cases} \frac{t-t_i}{t_{i+1}-t_i} & \text{falls} \quad t \in [t_{i+1}, t_i], \\ 0 & \text{sonst} \end{cases}$$

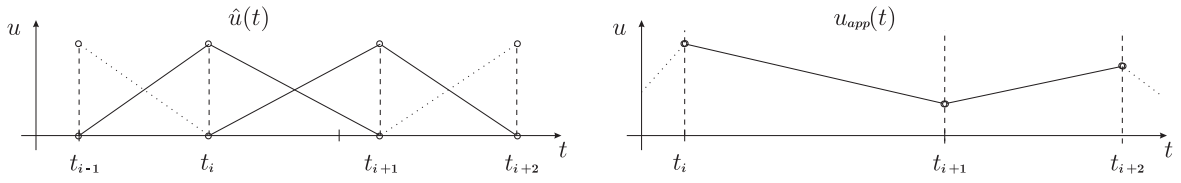
$$\hat{u}_{2i+1}(t) = \begin{cases} \frac{t_{i+1}-t}{t_{i+1}-t_i} & \text{falls} \quad t \in [t_{i+1}, t_i], \\ 0 & \text{sonst} \end{cases}$$

verwendet werden (4.3(a) auf der nächsten Seite). Somit werden $(2n_{u,\tau} - 2)$ Parameter je Steuerung für die Diskretisierung mit $n_{u,\tau}$ Punkten benötigt. Die Parameter sind die Werte der Steuerapproximationen auf dem Gitter bzw. die linksseitigen Grenzwerte an den Gitterpunkten.

Zustandsberechnung

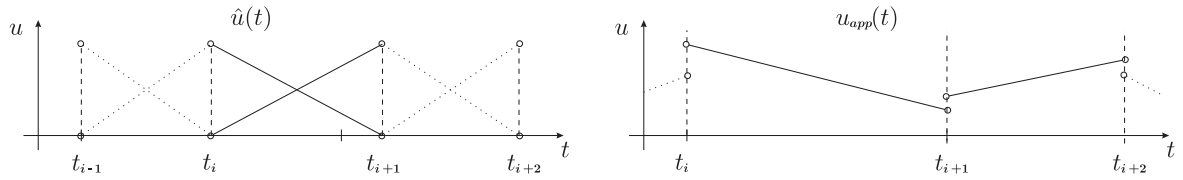
Die Steuerungen können im Folgenden aufgrund einer geeigneten Steuerdiskretisierung durch einen Satz von Parametern als gegeben angesehen werden, und es stellt sich die Frage, wie die Zustände berechnet werden können.

Häufig wird hier eine Mehrzielmethode verwendet. Dabei wird ausgehend von einer Schätzung $\boldsymbol{y}_{x,0}$ des Zustandes $\boldsymbol{x}(t_0)$ zur Anfangszeit vorwärts integriert bis zu einem Zwischenpunkt t^1, von dem aus mit der Schätzung $\boldsymbol{y}_{x,1}$ des Zustandes $\boldsymbol{x}(t^1)$ bis zu dem Zeitpunkt t^2 integriert wird, usw. Dadurch existieren an allen Zwischenpunkten $\boldsymbol{x}(t^i)$ durch Vorwärtsintegration berechnete Werte und Schätzwerte für die Zustände, deren Differenz gleich null werden muss, um eine Lösung zu erhalten. Der Schätzwert für den Zustand zur Anfangszeit und der berechnete Zustand zur Endzeit müssen die Randbedingungen erfüllen. Auf einem Überprüfungsgitter wird getestet, ob alle Nebenbedingungen von der Lösung eingehalten werden. Insgesamt ergibt sich ein nichtlineares Gleichungssystem, in das die Parameter der Steuerdiskretisierung und die Schätzwerte der Zustände an den Punkten t_i als Variablen eingehen. Das Gleichungssystem kann z. B. mit einem *Newton-Verfahren* gelöst werden. Eine Schwierigkeit dabei ist die Berechnung der Gradienten der Nebenbedingungen, da eine numerische Integration als Nebenbedingungen eingeht, was entsprechende Rechenzeiten für eine Gradientenapproximation verursacht. Mögliche Sprünge oder Knicke in den Zuständen müssen hier vom Integrationsverfahren entdeckt und gegebenenfalls muss die Integration neu aufgesetzt werden.



(a) Basis (b) Approximation

Abbildung 4.2: Stetig, stückweise lineare Approximation



(a) Basis (b) Approximation

Abbildung 4.3: Unstetig, stückweise lineare Approximation

In dieser Arbeit kommen Kollokationsverfahren zur Berechnung der Zustände zum Einsatz. Dazu werden, wie zuvor die Steuerungen, auch die Zustände $x(t) \in \mathbb{X}$ durch Funktionen aus einem geeigneten endlichdimensionalen Unterraum $\mathbb{X}_n \subset \mathbb{X}$ angenähert. Angenommen die Funktionen $\hat{x}_1(t), \ldots, \hat{x}_n(t)$ bilden eine Basis von $\mathbb{X}_n$, dann kann die Approximation $x_{app}(t)$ durch

$$x_{app}(t) = \sum_{i=1}^{n} y_{x,i} \hat{x}_i(t)$$

mit den Parametern $y_{x,1}, \ldots, y_{x,n} \in \mathbb{R}$ dargestellt werden. Typischerweise werden hier Polynomsplines verwendet (SB92; Hei98; Bet98; Con02; Ger05).

In (Con02) werden Polynome vierten und fünften Grades verwendet, um Low-Thrust-Manöver für Weltraumfahrzeuge zu berechnen. Hierbei bewegt sich der Schub für etwaige Steuerungen im Bereich von 10^{-3}g bei einer Manöverdauer von bis zu 200 Tagen für einem Erde-Mars-Transfer. Durch die gewählten Approximationen konnte im Vergleich zu anderen Ansätzenbei gleicher Genauigkeit in den Lösungen die Anzahl der Parameter niedrig gehalten werden.

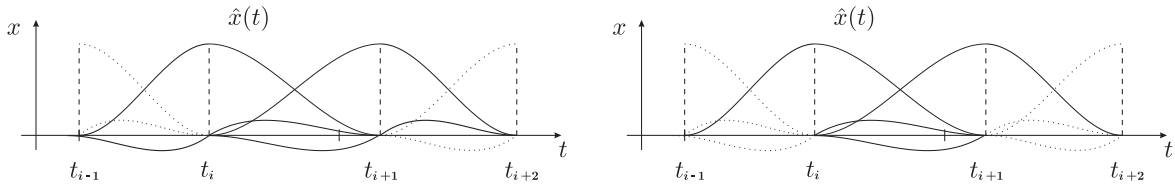
In den meisten Verfahren werden allerdings *kubische Splines* verwendet, da diese aufgrund des niedrigen Grades nicht so stark zu Oszillationen neigen, als solche mit höherem Grad. Außerdem ist der Grad hoch genug, um auch stetige Differenzierbarkeit zu erreichen. Damit besitzen die Basisfunktionen wiederum einen lokalen Träger. Eine geeignete Basis zur Darstellung liefern *Hermite-Polynome* (Hei98). Über dem Intervall $[0, 1]$ lauten diese

$$\begin{aligned}
\Phi_0(t) &= 1 - 3t^2 + 2t^3, \\
\Phi_1(t) &= t - 2t^2 + t^3, \\
\Phi_2(t) &= 3t^2 - 2t^3, \\
\Phi_3(t) &= -t^2 + t^3
\end{aligned}$$

und besitzen die Eigenschaft, dass

$$\begin{array}{llll}
\Phi_0(0) = 1, & \dot{\Phi}_0(0) = 0, & \Phi_0(1) = 0, & \dot{\Phi}_0(1) = 0, \\
\Phi_1(0) = 0, & \dot{\Phi}_1(0) = 1, & \Phi_1(1) = 0, & \dot{\Phi}_1(1) = 0, \\
\Phi_2(0) = 0, & \dot{\Phi}_2(0) = 0, & \Phi_2(1) = 1, & \dot{\Phi}_2(1) = 0, \\
\Phi_3(0) = 0, & \dot{\Phi}_3(0) = 0, & \Phi_3(1) = 0, & \dot{\Phi}_3(1) = 1
\end{array}$$

gilt. Die Koeffizienten einer Linearkombination stellen also genau die Funktionswerte und Stei-



(a) stetig differenzierbar (b) stetig

Abbildung 4.4: Kubische Approximation

gungen an den beiden Randpunkten des Intervalls $[0, 1]$ dar. Für beliebige Intervalle $[t_i, t_{i+1}]$ mit $t_{i+1} = t_i + h_i$ lässt sich diese Eigenschaft erhalten durch

$$\Phi_{0,i}(t) = \Phi_0(\tfrac{t_{i+1}-t}{h_i}), \quad \Phi_{2,i}(t) = \Phi_2(\tfrac{t_{i+1}-t}{h_i}), \quad \Phi_{1,i}(t) = h_i\Phi_1(\tfrac{t_{i+1}-t}{h_i}), \quad \Phi_{3,i}(t) = h_i\Phi_3(\tfrac{t_{i+1}-t}{h_i}).$$

Bei einer stetigen, stetig differenzierbaren Approximation eines Zustandes $x(t)$ wird eine Basis aus den Funktionen

$$\hat{x}_{2i} = \begin{cases} \Phi_{2,i-1}(t) & \text{falls} \quad t \in [t_{i-1}, t_i), \\ \Phi_{0,i}(t) & \text{falls} \quad t \in [t_i, t_{i+1}], \\ 0 & \text{sonst,} \end{cases}$$

$$\hat{x}_{2i+1} = \begin{cases} \Phi_{3,i-1}(t) & \text{falls} \quad t \in [t_{i-1}, t_i), \\ \Phi_{1,i}(t) & \text{falls} \quad t \in [t_i, t_{i+1}], \\ 0 & \text{sonst} \end{cases}$$

verwendet, womit sich der approximierte Zustand durch $2n_{x,\tau}$ Parameter darstellen lässt. Diese sind die Werte und Steigungen der Approximation auf dem Gitter.

Eine stetige, stückweise stetig differenzierbare Funktion besitzt eine Darstellung über die Basis

$$\hat{x}_{3i} = \begin{cases} \Phi_{2,i-1}(t) & \text{falls} \quad t \in [t_{i-1}, t_{i+1}), \\ \Phi_{0,i}(t) & \text{falls} \quad t \in [t_i, t_{i+1}], \\ 0 & \text{sonst,} \end{cases}$$

$$\hat{x}_{3i+1} = \begin{cases} \Phi_{1,i}(t) & \text{falls} \quad t \in [t_i, t_{i+1}], \\ 0 & \text{sonst} \end{cases}$$

$$\hat{x}_{3i+2} = \begin{cases} \Phi_{3,i}(t) & \text{falls} \quad t \in [t_i, t_{i+1}], \\ 0 & \text{sonst,} \end{cases}$$

was zu $(3n_{x,\tau} - 2)$ Parametern zur Darstellung von $x_{app}(t)$ führt.

Die unbekannten Parameter $y_{x,i}$ werden aus den expliziten und impliziten Randbedingungen, den

Tabelle 4.1: Zusammenfassung der Eigenschaften stückweiser Approximationen

Approximation	Eigenschaft an den Gitterpunkten	Parameter je Approximation
stückweise konstant	unstetig	$n_{u,\tau} - 1$
stückweise linear	stetig	$n_{u,\tau}$
	unstetig	$2n_{u,\tau} - 2$
stückweise kubisch	stetig, nicht stetig differenzierbar	$3n_{x,\tau} - 2$
	stetig, stetig differenzierbar	$2n_{x,\tau}$

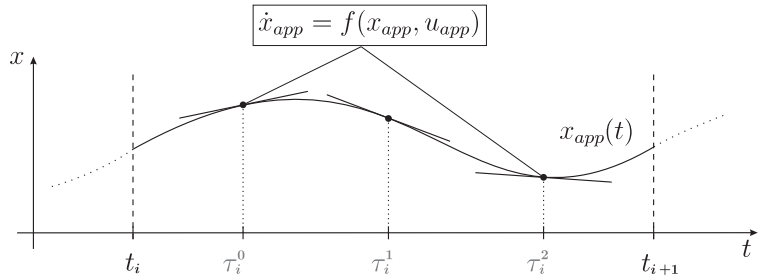


Abbildung 4.5: Kollokation an Gauß-Punkten

inneren Punktbedingungen, sowie den *Kollokationsbedingungen*

$$\dot{x}_{app}(\tau_i^j) = f(x_{app}(\tau_i^j), u_{app}(\tau_i^j), \tau_i^j)$$

bestimmt, letztere entstehen, wenn der Wert der Approximation und deren Ableitung zu geeigneten Zeitpunkten τ_i^j (*Kollokationspunkten*) in die Differentialgleichung eingesetzt werden. Die Kollokationspunkte werden in jedem Teilintervall mit gleicher relativer Lage gewählt. Die Lage der Punkte ist entscheidend für die Güte der Approximation. Schlecht gewählte Punkte können bei direkten Kollokationsverfahren zu Konvergenzproblemen oder auch sehr stark schwingenden Lösungen führen. Die Anzahl der Punkte muss so gewählt sein, dass das Gleichungssystem, bestehend aus den Kollokationsbedingungen bei geeigneter vorgegebener Steuerung und konsistenten Randbedingungen, eine eindeutige Lösung besitzt. Für eine stetige stückweise polynomiale Funktion mit Polynomstücken n-ten Grades werden n Kollokationspunkte benötigt, für stetig differenzierbare genügen $n - 1$ Punkte.

Bei der Kollokation an *Gauß-Punkten*, die als Nullstellen des *Legendrepolynoms* n-ten Grades

$$\frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}x^n}((x^2 - 1)^n)$$

berechnet werden können, ergibt sich eine höhere *Konvergenzordnung* als bei anderen Punkten. Diese Nullstellen befinden sich im Intervall $[-1, 1]$ und sind symmetrisch um den Nullpunkt verteilt, womit im ungeraden Fall die Null eine Nullstelle darstellt. Im Fall $n = 2$ ergeben sich die Nullstellen $\zeta_{1,2} = \pm\sqrt{\frac{1}{3}}$, im hier betrachteten Fall $n = 3$ kommen zur 0 noch $\pm\sqrt{\frac{3}{5}}$ (vgl. 4.5).

In (Hei98) wird ein direktes Kollokationsverfahren für Systeme zweiter Ordnung, die bei Fragestellungen aus den technischen Bereichen wie z. B. der Mechanik häufig vorkommen, vorgestellt. Auch hier wird für die Zustände ein kubischer Spline, der stetig und auch stetig differenzierbar ist,

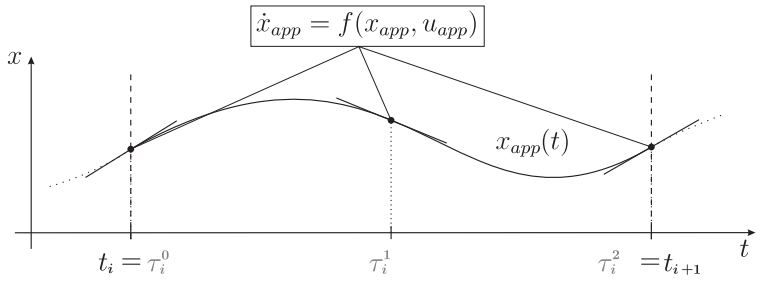


Abbildung 4.6: Kollokation an Lobatto-Punkten

angesetzt. Die erste Ableitung der Zustände stellt somit einen quadratischen Spline dar, der stetig, jedoch nicht mehr stetig differenzierbar ist. Um diesen zu bestimmen, reichen Kollokationsbedingungen an zwei Punkten je Intervall aus.

Im Gegensatz zu den Gauß-Punkten enthalten die Lobatto-Punkte auch die Randpunkte der Teilintervalle. Bei stetiger rechter Seite der Differentialgleichung, insbesondere also bei stetiger Steuerung, sind die linksseitige und die rechtsseitige Ableitung des Zustands auf dem Diskretisierungsgitter gleich:

$$f(x_{app}(t_i^-), u_{app}(t_i^-), t_i) = f(x_{app}(t_i^+), u_{app}(t_i^+), t_i)$$

Damit kann ein stückweise stetig differenzierbarer Polynomspline angesetzt werden, wodurch sich die Anzahl der Parameter reduziert. Da in diesem Fall die Ableitungen des Zustands auf dem Gitter explizit durch die rechte Seite der Differentialgleichung gegeben sind, genügt es, alleine die Kollokationsbedingung in der Mitte des Intervalls zu fordern. Bei diesem Ansatz muss die rechte Seite der Differentialgleichung $(2n_{x,\tau}-1)$-mal ausgewertet werden. Im Gegensatz dazu werden bei unstetiger rechter Seite oder bei der vorher besprochenen Kollokation an Gauß-Punkten $3n_{x,\tau} - 3$ Auswertungen benötigt.

Bemerkung 13.
Die Gitter, die durch die Zeitdiskretisierung für die Steuerung und den Zustand entstehen, müssen nicht identisch sein. Die feinste sinnvolle Diskretisierung für die Steuerungen ergibt sich durch die Punkte, an denen Kollokationsbedingungen ausgewertet werden. Bei gröberen Diskretisierungen können Parameter eingespart werden, die benötigten Werte der Steuerungen werden dann aus deren Approximationen berechnet.

Bemerkung 14.
Falls bei der Lösung des Problems in den Zuständen oder Steuerungen Unstetigkeiten oder Knicke zu erwarten sind, so ist dies bei der Wahl der Approximationen und der Festlegung des Gitters zu beachten. Derartige Punkte sollten auf dem Gitter liegen, oder das Problem sollte in mehrere Phasen aufgeteilt werden, so dass diese Punkte doppelt, mit rechtsseitigen und linksseitigen Grenzwerten, in der Diskretisierung vorkommen.

Nebenbedingungen und Zielfunktional

Bisher können zu gegebener Approximation der Steuerung Approximationen der Zustände aus den Kollokationsbedingungen zusammen mit den Rand- und inneren Punktbedingungen berechnet

werden. Diese sollen aber auch gewisse Gleichungs- oder Ungleichungsnebenbedingungen (inklusive *Box-Bedingungen*) einhalten. Dazu werden die Zustands- und Steuerungsapproximationen zu diskreten Punkten in die Nebenbedingungen eingesetzt und auf Zulässigkeit geprüft. Das Gitter, auf dem die Zulässigkeit überprüft wird, muss nicht mit den bisher besprochenen Gittern übereinstimmen, bei Gleichheit kann allerdings zusätzlicher Rechenaufwand eingespart werden.

Falls bei identischen Gittern eine Zustandsbeschränkung aktiv, also eine Ungleichungsbedingung mit Gleichheit erfüllt ist, neigen die Zustände oft dazu, zwischen den Gitterpunkten überzuschwingen, wie in Abbildung 4.7 zu sehen ist. Dies kann weitgehend vermieden werden, indem die Nebenbedingungen auch in den Intervallmitten des Diskretisierungsgitters überprüft werden (Abbildung 4.7 graue Kurve).

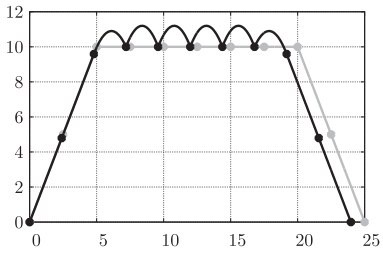


Abbildung 4.7: Überschwingen zwischen Gitterpunkten

Schließlich wird noch das Zielfunktional betrachtet. Wie bereits in Bemerkung 1 auf Seite 17 erwähnt, kann ein Bolza-Zielfunktional auf reine Mayer-Form gebracht werden, wobei dann das Integral über den Zeitbereich als Zustandsdifferentialgleichung geschrieben wird, und der Wert des Integrals dem Wert des entsprechenden Zustands zur Endzeit entspricht. In dem Mayer-Term werden dann die Zustandsapproximationen eingesetzt, womit dieser auch nur noch von den Parametern der Approximation abhängt.

Insgesamt ergibt sich durch die beschriebene Diskretisierung ein nichtlineares Optimierungsproblem

$$\begin{aligned} &\min \quad \Phi(\boldsymbol{y}) \\ &\text{unter} \quad \boldsymbol{y} \in \mathbb{R} := \left\{ \boldsymbol{y} \in \mathbb{R}^{n_y^u + n_y^x} \mid \boldsymbol{g}(\boldsymbol{y}) = 0, \boldsymbol{h}(\boldsymbol{y}) \leq 0 \right\}, \end{aligned} \tag{4.1}$$

dessen Größe je nach Art der gewählten Approximation variiert. Die möglichen Dimensionen in Abhängigkeit der Gitter sind in Tabelle 4.1 auf Seite 60 aufgelistet.

Zusammenfassung

Je nachdem, welche Stetigkeitsanforderungen für die Steuerungen und Zustände vorliegen, kann dies schon bei der Wahl der Diskretisierung berücksichtigt werden. Eine stückweise konstante Steuerung eignet sich z. B. gut bei Bang-Bang-Verhalten.

Bei Teilintervallen, in denen Zustandsbeschränkungen aktiv werden, können die Zustände zwischen den Gitterpunkten der Diskretisierung überschwingen. Hier eignen sich Verfahren, bei denen die Nebenbedingungen auch außerhalb des Diskretisierungsgitters für Zustände und Steuerungen überprüft werden.

Die Kollokation an Lobatto-Punkten liefert Optimierungsprobleme mit weniger Nebenbedingungen als eine Kollokation an Gausspunkten. Dafür ist ihre Struktur etwas komplizierter und die Approximationsordnung niedriger.

4.1.2 Sequentielle quadratische Optimierung

In der statischen nichtlinearen Optimierung soll eine Aufgabe wie oben (4.1) gelöst werden. Dabei bildet die Zielfunktion Φ den zulässigen Bereich $\mathbb{R} \subset \mathbb{R}^{n_y}$ in die Menge der reellen Zahlen $\mathbb{R}$ ab. Der zulässige Bereich selbst ist durch Gleichungs- und Ungleichungsbedingungen mit $\boldsymbol{g} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_g}$ und $\boldsymbol{h} : \mathbb{R}^{n_y} \to \mathbb{R}^{n_h}$ bestimmt. Eine Lösung für diese Aufgabe existiert, sobald der zulässige Bereich $\mathbb{R}$ nichtleer und die Zielfunktion auf $\mathbb{R}$ nach unten beschränkt ist.

Optimalitätsbedingungen

Definition 8 (Lagrangefunktion)**.**
Die Linearkombination aus Zielfunktion und Nebenbedingungen

$$L(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \Phi(\boldsymbol{y}) + \boldsymbol{\lambda}^T \boldsymbol{g}(y) + \boldsymbol{\mu}^T \boldsymbol{h}(y)$$

heißt Lagrangefunktion für Problem (4.1). Die Parameter $\boldsymbol{\lambda} \in \mathbb{R}^{n_g}$ und $\boldsymbol{\mu} \in \mathbb{R}^{+\,n_h}$ werden *Lagrange'sche Multiplikatoren* genannt.

Eine *reguläre* lokale Lösung $\boldsymbol{y}^*$ von (4.1) mit zugehörigen Multiplikatoren $\boldsymbol{\lambda}^*$ und $\boldsymbol{\mu}^*$ erfüllt bei entsprechender Differenzierbarkeit die hinreichenden Bedingungen zweiter Ordnung:

$$\begin{aligned}
\boldsymbol{y}^* &\in \mathbb{R} && (4.2)\\
L_{\boldsymbol{y}}(\boldsymbol{y}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) &= 0,\ \boldsymbol{\mu}^* \geq 0 && (4.3)\\
\mu_j^* &\neq 0 \Leftrightarrow h_j(\boldsymbol{y}^*) = 0,\ j = 1, \ldots, n_h. && (4.4)\\
\boldsymbol{s}^T L_{\boldsymbol{yy}}(\boldsymbol{y}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\boldsymbol{s} &> 0,\ \forall \boldsymbol{s} \in \mathbb{S} \subset \mathbb{R}^{n_y} && (4.5)\\
\sum_{i=0}^{n_g} \alpha_i g_i\, \boldsymbol{y}(\boldsymbol{y}^*) + \sum_{i \in \mathbb{A}(\boldsymbol{y}^*)} \beta_j h_j\, \boldsymbol{y}(\boldsymbol{y}^*) &\neq 0,\ \forall \alpha_i, \beta_j \neq 0 && (4.6)
\end{aligned}$$

mit

$$\begin{aligned}
\mathbb{S} &:= \left\{ \boldsymbol{s} \in \mathbb{R}^{n_y} \;\middle|\; g_i\, \boldsymbol{y}(\boldsymbol{y}^*)^T \boldsymbol{s} = 0,\ i = 1, \ldots, n_g, h_j\, \boldsymbol{y}(\boldsymbol{y}^*)^T \boldsymbol{s} = 0,\ j \in \mathbb{A}(\boldsymbol{y}^*) \right\}\\
\mathbb{A}(\boldsymbol{y}^*) &:= \{ j \in \{1, \ldots, n_h\} \mid h_j(\boldsymbol{y}^*) = 0 \}.
\end{aligned}$$

Die Indexmenge $\mathbb{A}(\boldsymbol{y}^*)$ enthält die Indizes der im Punkt $\boldsymbol{y}^*$ *aktiven* Ungleichungsnebenbedingungen. Eine Nebenbedingung heißt aktiv, falls sie mit Gleichheit erfüllt ist.

Die Optimalitätsbedingungen (4.2) bis (4.6) werden nun genauer betrachtet. Die Gleichungen (4.2) bis (4.4) sind Bedingungen erster Ordnung und werden *Kuhn-Tucker-Bedingungen* genannt. Hierbei wird zunächst für eine Lösung von (4.1) die *primale Zulässigkeit* durch Gleichung (4.2) gefordert.

Die *Multiplikatorregel* (4.3) bedeutet in Verbindung mit *strikter Komplementarität* (4.4) geometrisch, dass der Vektor $-\Phi_{\boldsymbol{y}}(\boldsymbol{y}^*)$ in Richtung fallender Werte der Zielfunktion Φ aus den Außennormalen $g_i\, \boldsymbol{y}(\boldsymbol{y}^*)$, $i = 1, \ldots, n_g$ und $h_j\, \boldsymbol{y}(\boldsymbol{y}^*),\ j \in A(\boldsymbol{y}^*)$ der aktiven Nebenbedingungen linear

kombinierbar ist:

$$-\Phi_{\boldsymbol{y}}(\boldsymbol{y}^*) = \sum_{i=0}^{n_g} \lambda_i g_i\ \boldsymbol{y}(y^*) + \sum_{i \in \mathbb{A}(\boldsymbol{y}^*)} \mu_j h_j\ \boldsymbol{y}(y^*)$$

Folglich wird ein Fortschreiten in dieser Richtung durch den Rand des zulässigen Bereichs verhindert.

Die hinreichende Bedingung zweiter Ordnung (4.5) auf der vorherigen Seite verlangt, dass die Hesse-Matrix der Lagrangefunktion im Lösungspunkt bezüglich der Richtungen tangential zu den Gleichungsnebenbedingungen und zu den aktiven Ungleichungsbedingungen positiv definit ist. Das bedeutet, dass in Richtungen $\boldsymbol{s} \in \mathbb{S}$ entlang des Randes des zulässigen Bereichs mit wachsenden Zielfunktionswerten zu rechnen ist.

Der Punkt $\boldsymbol{y}^*$ erfüllt die Regularitätsbedingung, falls wie in (4.6) gefordert die Gradienten der Gleichungsbedingungen und aktiven Ungleichungsbedingungen linear unabhängig sind.

Die Bedingungen (4.2) bis (4.6) auf der vorherigen Seite stellen sicher, dass der Lösungspunkt ein isoliertes lokales Minimum ist. Falls es sich bei Problem (4.1) auf Seite 62 um ein *konvexes* Optimierungsproblem handelt, also die Zielfunktion Φ und auch der zulässige Bereich $\mathbb{R}$ konvex sind, so ist jedes lokale Minimum auch ein globales, und die Menge der globalen Minima ist ebenfalls konvex. Der zulässige Bereich ist im Allgemeinen dann konvex, falls die Funktionen $\boldsymbol{h}(\boldsymbol{y})$ konvex und die Funktionen $\boldsymbol{g}(\boldsymbol{y})$ linear sind.

Die Gleichungen (4.3) und (4.5) deuten darauf hin, dass auch die Lagrangefunktion für geeignetes $\boldsymbol{\lambda}$ und $\boldsymbol{\mu}$ ein Minimum bezüglich der Variablen $\boldsymbol{y}$ hat. Bei beliebig gewählten Multiplikatoren würde im Allgemeinen der Versuch, die Lagrangefunktion zu minimieren, nicht zu einer Lösung der Aufgabe (4.1) auf Seite 62 führen, da ein Verlassen des zulässigen Bereichs zur Verkleinerung der Kosten genutzt werden kann.

Bemerkung 15. Die Lagrangefunktion des Optimierungsproblems, das aus der Diskretisierung des Optimalsteuerungsproblems durch Kollokation entsteht, stellt eine diskretisierte Version der erweiterten Hamiltonfunktion (siehe Definition 2 auf Seite 28) dar. Damit lassen sich die adjungierten Variablen aus den Lagrangemultiplikatoren berechnen (vgl. (Ret03)), und somit die Optimalitätsbedingungen aus Abschnitt 2.3 auf Seite 27 a posteriori überprüfen.

Algorithmus

Nach dem Vorbild des Newton-Verfahrens wird zu einer gegebenen Iterierten $\Upsilon_k := (\boldsymbol{y}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$ ein möglichst einfaches Problem formuliert, aus dem die nächste Iterierte bestimmt werden kann. Die hinreichenden Bedingungen zweiter Ordnung dieses Problems sollen dabei mit denen von (4.1) auf Seite 62 übereinstimmen. Diese Forderung führt zu den folgenden quadratischen Teilproblemen

$$\begin{aligned} \min \quad & \Phi_k(\boldsymbol{y}) \\ \text{unter} \quad & \boldsymbol{y} \in \mathbb{R}_k \end{aligned} \tag{4.7}$$

mit

$$\begin{aligned}
\Phi_k(\boldsymbol{y}) &:= \nabla\Phi(\boldsymbol{y}_k)^T(\boldsymbol{y}-\boldsymbol{y}_k) + \tfrac{1}{2}(\boldsymbol{y}-\boldsymbol{y}_k)^T\nabla^2_{\boldsymbol{y}\boldsymbol{y}}\mathrm{L}(\Upsilon_k)(\boldsymbol{y}-\boldsymbol{y}_k)\\
R_k &:= \{\boldsymbol{y}\in\mathbb{R}^{n_{\boldsymbol{y}}} : \mathrm{a}_{k,i}(\boldsymbol{y}) = 0,\ i=1,\ldots,n_a,\ \mathrm{b}_{k,j}(\boldsymbol{y}) \le 0,\ j=1,\ldots,n_b\}\\
\mathrm{a}_{k,i}(\boldsymbol{y}) &:= \mathrm{a}_i(\boldsymbol{y}_k) + \nabla\mathrm{a}_i(\boldsymbol{y}_k)^T(\boldsymbol{y}-\boldsymbol{y}_k)\\
\mathrm{b}_{k,j}(\boldsymbol{y}) &:= \mathrm{b}_j(\boldsymbol{y}_k) + \nabla\mathrm{b}_j(\boldsymbol{y}_k)^T(\boldsymbol{y}-\boldsymbol{y}_k).
\end{aligned}$$

Die Lösung Υ^{k*} von (4.7) auf der vorherigen Seite liefert dann den nächsten Iterationspunkt oder zumindest eine gute Suchrichtung $s_k = \Upsilon^{k*} - \Upsilon_k$.

Um die Lösbarkeit von (4.7) auf der vorherigen Seite zu garantieren, und den Aufwand der Teilprobleme zu verringern, wird die exakte Hesse-Matrix $\nabla^2_{\boldsymbol{y}\boldsymbol{y}}\mathrm{L}(\Upsilon_k)$ der Lagrangefunktion meist durch eine symmetrische, positiv definite Approximation M_k ersetzt.

Zur Konstruktion eines global konvergenten Verfahrens wird eine geeignete Bewertungsfunktion $\psi(\boldsymbol{y})$ gewählt, deren unrestringierte Minima mit denen von (4.1) auf Seite 62 zusammenfallen. Mit Hilfe dieser Bewertungsfunktion wird eine Schrittweitensteuerung durchgeführt. Häufig sind die verwendeten Bewertungsfunktionen *erweiterte Lagrangefunktionen* oder *Penaltyfunktionen*.

Der gesamte Ablauf ist in Algorithmus 1 zusammengefasst.

Algorithmus 1 Sequentielle quadratische Optimierung

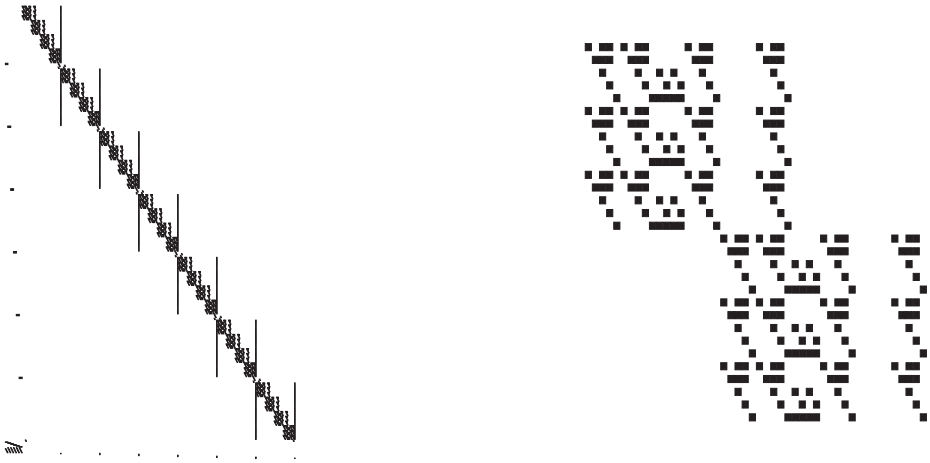
1: wähle $\Upsilon_0 = (\boldsymbol{y}_0, \lambda_0, \mu_0)$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: **if** $\nabla\psi(\boldsymbol{y}_k) = 0$ **then**
4: Ende
5: **end if**
6: berechne M_k
7: löse (4.7) mit Lösung Υ_k^* und setze $s_k = \Upsilon_k^* - \Upsilon_k$
8: bestimme mittels $\psi(\boldsymbol{y}_k)$ eine geeignete Schrittweite σ_k
9: setze $\Upsilon_{k+1} = \Upsilon_k + \sigma_k s_k$
10: **end for**

4.1.3 Konvergenzverbesserungen

In diesem Abschnitt werden Möglichkeiten beschrieben, wie das Lösungsverhalten von direkten Kollokationsverfahren verbessert werden kann. Da dies unabhängig von diskretwertigen Steuerungen und Parametern ist, werden diese der Übersichtlichkeit halber weggelassen.

Strukturausnutzung und Sensitivitäten

Bei den hier beschriebenen direkten Kollokationsverfahren werden die Steuerungen und Zustände durch Funktionen mit lokalem Träger in der Zeit approximiert und durch Parameter ausgedrückt.



(a) Blockstruktur eines 7-phasigen Problems

(b) Beispielstruktur zweier Blöcke innerhalb einer Phase

Abbildung 4.8: Struktur der Jacobi-Matrix des Optimierungsproblems bei direkter Kollokation

Die Parameter haben nur direkten Einfluss innerhalb ihres Trägers und aufgrund von Stetigkeitsforderungen auf die direkten Nachbarn. Wird nun die *Jacobi-Matrix* des resultierenden Optimierungsproblems betrachtet, so hat diese, wie Abbildung 4.8(a) zu sehen ist, eine ausgeprägte *Blockstruktur*, die die Zeitspannen zwischen den Diskretisierungspunkten widerspiegelt. Diese Blockstruktur lässt sich allein mit dem Wissen über die bei der Diskretisierung verwendeten Ansatzfunktionen angeben. Laufzeitverbesserungen durch die Ausnutzung der Blockstruktur wurden z. B. in (GVS00; Str03) untersucht. Typischerweise sind die Funktionen, die ein Optimalsteuerungsproblem formen, jeweils selbst nur von wenigen Problemvariablen abhängig, insbesondere bei Differentialgleichgungsnebenbedingungen höherer Ordnung (vgl. 3 auf Seite 18). Diese Struktur, wie sie in Abbildung 4.8(b) zu sehen ist, hängt vom Problem ab, und muss jeweils vom Anwender gegeben werden. Die volle Strukturausnutzung verbessert die Laufzeit eines Optimierungsverfahrens und auch seine Stabilität.

Die Einträge in der Jacobi-Matrix stellen die Ableitungen der Funktionen nach den Parametern y_i der Diskretisierung dar. Diese können numerisch approximiert oder analytisch berechnet werden. Für die in Abschnitt 4.1 auf Seite 54 besprochenen Diskretisierungen berechnen sich die Ableitungen exemplarisch für eine Ungleichungsnebenbedingung $h_j(\boldsymbol{x}, \boldsymbol{u}, t)$ ohne Parameter durch

$$\left.\frac{\partial h_j}{\partial y_i}\right|_{(.)} = \left.\frac{\partial h_j}{\partial \boldsymbol{x}}\right|_{(.)} \left.\frac{\partial \boldsymbol{x}_{app}}{\partial y_i}\right|_{(.)} + \left.\frac{\partial h_j}{\partial \boldsymbol{u}}\right|_{(.)} \left.\frac{\partial \boldsymbol{u}_{app}}{\partial y_i}\right|_{(.)} + \left.\frac{\partial h_j}{\partial t}\right|_{(.)} \left.\frac{\partial t}{\partial y_i}\right|_{(.)},$$

wobei die $\boldsymbol{u}_{app}$ nur von Parametern $\boldsymbol{y}_u$ abhängen und $\partial h_j / \partial y_i$ nur auftritt, falls $y_i = y_{t_i}$ der die Zeit t_i beschreibende Parameter ist. Bei der Verwendung von Gauß-Punkten zur Kollokation hängt $\boldsymbol{x}_{app}$ nur von Parametern $\boldsymbol{y}_x$ ab. Hingegen ist bei Lobatto-Punkten im Intervall $[t_i, t_{i+1}]$ die Approximation von $\boldsymbol{x}$ gegeben durch

$$x_{app,i}(t) = y_{x,i}\Phi_{0,i}(t) + f(y_{x,i}, y_{u,i}, t_i)h_i\Phi_{2,i}(t) + f(y_{x,i+1}, y_{u,i+1}, t_{i+1})h_i\Phi_{3,i}(t) + y_{x,i+1}\Phi_{4,i}(t),$$

womit $\boldsymbol{x}_{app}$ nicht nur von Parametern $y_{x,i}$, sondern auch von Parametern $y_{u,i}$ und t_i abhängen kann. Die Approximation mit Gauß-Punkten ist weniger verschachtelt als die mit Lobatto-Punkten.

Die Ableitungen $\partial h_j/\partial \boldsymbol{x}_{app}$ und $\partial h_j/\partial \boldsymbol{u}_{app}$ können numerisch approximiert, und daraus kann $\partial h_j/\partial y_i$ berechnet werden, was genauere Sensitivitäten ergibt, als eine direkte numerische Approximation von $\partial h_j/\partial y_i$ (vgl. (Str01)). Alternativ können sie auch vom Anwender analytisch oder durch automatisches Differenzieren bereit gestellt werden. In den meisten Fällen zeigt sich keine merkliche Verbesserung im Konvergenzverhalten bei der Verwendung exakter Ableitungen, jedoch ist die Bereitstellung analytisch berechneter Sensitivitäten fehleranfällig und zeitaufwändig.

Gitterverfeinerung

Bei Kollokationsverfahren erfüllen die Approximationen die Differentialgleichungen an den Kollokationspunkten. Die weiteren Nebenbedingungen werden ebenfalls an diskreten Punkten erfüllt. Außerhalb dieser diskreten Punkte ergeben sich in natürlicher Weise Abweichungen, die sich durch Einsetzen der Näherungen berechnen lassen

$$\begin{aligned} err_{coll_i}(t) &= \left\| \dot{\boldsymbol{x}}_{app}(t) - f(\boldsymbol{x}_{app}(t), \boldsymbol{x}_{app}(t), t) \right\|_2 \\ err_{g_i}(t) &= \left\| g_i(\boldsymbol{x}_{app}(t), \boldsymbol{x}_{app}(t), t) \right\|_2 \\ err_{h_i}(t) &= \max\left\{ h_i(\boldsymbol{x}_{app}(t), \boldsymbol{x}_{app}(t), t), 0 \right\}. \end{aligned}$$

Im numerischen Verfahren können diese Fehler auf einem Testgitter berechnet werden. Ist einer der Fehler größer als eine vorgegebene Schranke, so kann hier die für die Approximationen verwendete Zeitdiskretisierung verfeinert werden. Dabei ist darauf zu achten, dass zwei benachbarte Gitterpunkte nicht zu eng zusammenliegen.

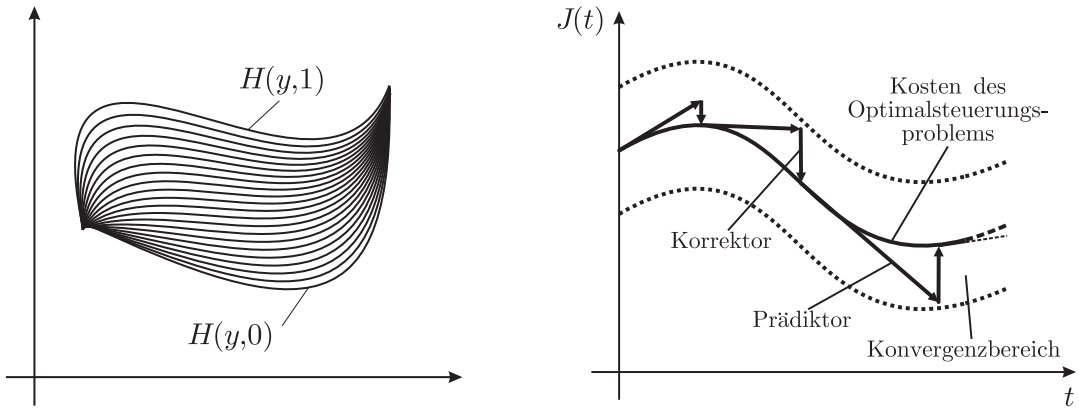
Gegebenenfalls kann es auch sinnvoll sein, Gitterpunkte zu entfernen, wenn eine Aufgabe erneut mit leicht geänderter Problemstellung gelöst werden soll. Bei der gleichzeitigen Nutzung von Gitterverfeinerung und Vergröberung kann es allerdings vorkommen, dass Punkte im Wechsel entfernt und wieder hinzugefügt werden.

Skalierung

Ziel der Skalierung ist es, dass möglichst alle Variablen und Nebenbedingungen des skalierten Optimierungsproblems von der gleichen Größenordnung sind. Zunächst werden die Variablen mit Box-Bedingungen $y_{i,min} \leq y_i \leq Y_{i,max}$ durch

$$\hat{y}_i := \frac{y_i - y_{i,min}}{y_{i,max} - y_{i,min}} = \frac{y_i - y_{i,min}}{y_{i,scale}}$$

skaliert. Nun ist die neue Problemvariable $\hat{y}_i$ mit Wertebereich $[0, 1]$. Bei unbeschränkten Variablen kann die Größenordnung, in der sie sich bewegen, geschätzt und zur Skalierung verwendet werden. Nun werden auch die Problemfunktionen skaliert und so abgeändert, dass sie für die skalierten



(a) Homotopie zweier Kurven

(b) Prädiktor-Korrektor Schritt bei Homotopieverfahren

Abbildung 4.9: Idee bei Homotopieverfahren

Variablen die gewünschten Ergebnisse liefern. Bei einer Ungleichungsnebenbedingung der Form $h_{min} \leq h_i(y) \leq h_{i,max}$ wäre dies

$$0 \leq \frac{h_i(y_{i,min} + \hat{y}_i y_{i,scale}) - h_{i,min}}{h_{i,scale}} \leq 1.$$

Durch diese Transformation ergeben sich die Einträge der Jacobi-Matrix des skalierten Problems zu

$$\frac{\partial h_i}{\partial y} \frac{y_{i,scale}}{h_{i,scale}}.$$

Da es sich hier um Parameter und Funktionen aus der Diskretisierung eines Optimalsteuerungsproblems handelt, sollten all diejenigen Parameter, die zum gleichen Zustand, zur gleichen Steuerung etc., gehören, gleich skaliert werden. Dies gilt analog für die Funktionen.

Bei den beschriebenen Kollokationsverfahren entsprechen die Parameter des resultierenden Optimierungsproblems den Werten der Endzeit der Steuerungen und der Zustände, womit deren Skalierung direkt auf die Parameter übertragen werden kann. Analog geht dies bei den Neben-, Rand- und inneren Punktbedingungen. Bei den Kollokationsbedingungen muss allerdings darauf geachtet werden, dass durch die Ableitung der Zustände auch die Zustands- und Zeitskalierung in die Skalierung eingeht:

$$\frac{\mathrm{d}\hat{x}_i}{\mathrm{d}\hat{t}} = \frac{t_{f\,scale}}{x_{i,scale}} f_i(\boldsymbol{x}, \boldsymbol{u}, \hat{t}).$$

4.1.4 Homotopieverfahren

Zur Verwendung eines Kollokationsverfahrens muss der Benutzer eine Startschätzung für die gesuchte Lösung angeben. Diese muss nicht so exakt sein wie bei indirekten Verfahren. Schlechte Schätzungen können allerdings zur Folge haben, dass keine oder eine physikalisch unsinnige Lösung gefunden wird. Auch die Rechenzeit hängt von der Startnäherung ab. Um auch dann Konvergenz zu erhalten, wenn nur schlechte oder keine Schätzungen für die Lösung vorliegen, können Homotopieverfahren eingesetzt werden.

Definition 9 (homotop)**.**
Zwei stetige Abbildungen $a, b : \mathbb{R}^n \to \mathbb{R}^m$ heißen genau dann *homotop*, wenn es eine stetige Abbildung $H(y, \tau) \in \mathbb{R}^m$ mit $H(y, 0) = a(y)$ und $H(y, 1) = b(y)$ für alle $y \in \mathbb{R}^n$ gilt. Der Parameter $\tau \in [0, 1]$ wird *Homotopieparameter* genannt.

Die Funktion a wird also durch die Veränderung von τ so deformiert, dass schließlich die Funktion b entsteht. Übertragen auf Optimalsteuerungsprobleme bedeutet dies, dass ein einfaches Problem schrittweise in ein komplexeres umgeformt und jeweils die Lösung des Vorgängers als Startschätzung verwendet wird.

Homotopieverfahren und Nullstellensuche

Gewöhnlicherweise werden Homotopieverfahren in Verbindung mit einem Newton-Verfahren zur Nullstellensuche eingesetzt. Dazu wird die Funktion $b(y)$, deren Nullstelle berechnet werden soll, in eine Homotopie, wie z. B. $H(y, \tau) = (1-\tau)a(y)+\tau b(y)$, eingebettet. Der Homotopieparameter wird schrittweise von null auf eins erhöht, und zu jedem Schritt $\Delta\tau_i$ eine Lösung $y_0(\tau_{i+1})$ des dazugehörigen Nullstellenproblems $H(y_0(\tau_{i+1}), \tau_{i+1}) = 0$ mit $\tau_{i+1} = \tau_i + \Delta\tau_i$ berechnet, wobei die Lösung des letzten Schritts als Startwert dient.

Eine kontinuierliche Variante kann aus der Sensitivität $\mathrm{d}y_0(\tau)/\mathrm{d}\tau$ der Lösung des Nullstellenproblems bezüglich des Homotopieparameters gewonnen werden. $\mathrm{d}y_0(\tau)/\mathrm{d}\tau$ lässt sich aufgrund des Satzes über implizite Funktionen durch

$$\frac{\mathrm{d}y_0(\tau)}{\mathrm{d}\tau} = \frac{\partial H(y_0(\tau), \tau)}{\partial y}^{-1} \frac{\partial H(y_0(\tau), \tau)}{\partial \tau}$$

berechnen. Diese Differentialgleichung wird *Davidenko-Differentialgleichung* genannt und ihre Lösung liefert den Verlauf der Nullstellen in Abhängigkeit des Homotopieparameters. Da die rechte Seite im Allgemeinen nicht explizit vorliegt, muss sie numerisch approximiert werden, was mit hohem Aufwand verbunden ist.

Eine weitere Variante ergibt sich durch einen *Prädiktor-Korrektor*-Ansatz , bei dem durch numerische Integration der Davidenko-Differentialgleichung über ein Teilintervall $[\tau_i, \tau_{i+1}]$ eine bessere Näherung der nächsten Nullstellen berechnet wird, die als Startwert zur Bestimmung der Nullstelle mit dem Newton-Verfahren verwendet wird. In (Gri97) werden solche Prädiktor-Korrektor-Verfahren zur Berechnung von optimalen Flugbahnen verwendet. Die Optimalsteuerungsaufgabe wird dabei mit einem indirekten Verfahren gelöst, womit Homotopieverfahren für die Newton-Methode direkt einsetzbar sind. Im Speziellen werden Prädiktoren verschiedener Ordnung untersucht, die sich durch Integration der Davidenko-Differentialgleichung mit Integrationsverfahren verschiedener Ordnung konstruieren lassen. Die Ordnungen können anhand des zu erwartenden Aufwands oder des Erfolgs der vorigen Wahl gesteuert werden. Die Schrittweiten werden so gesteuert, dass der Startwert für den Korrektorschritt bereits so nahe an der Lösung liegt, dass das Newton-Verfahren in einem einzigen Schritt konvergiert. Weitere ähnliche Strategien werden bei den in HOMPACK (WBM87) implementierten Verfahren angewandt.

Bei der Verwendung von Homotopieverfahren ist in jedem Fall auf *Bifurkationen* zu achten. Diese entstehen z. B. bei Verschiebung einer Funktion durch den Homotopieparameter, bei der ab einem bestimmten Parameterwert mehrere mögliche Lösungen zur Verfügung stehen.

Kollokationsverfahren mit Homotopieschritten

Bei der Lösung von Optimalsteuerungsaufgaben mittels Kollokation werden nach der Transformation in ein nichtlineares Optimierungsproblem auf unterster Ebene Nullstellenprobleme gelöst, womit die genannten Prädiktor-Korrektor-Ansätze direkt verwendet werden könnten. Dies wird allerdings durch den Einsatz von Software Dritter erschwert, wenn es überhaupt bei der gegebenen Codestruktur möglich ist. Ein weiteres Problem ist die Diskretisierung beim Kollokationsverfahren. Diese mag beim Start des Homotopieverfahrens noch günstig sein, unter Umständen kann aber nach der vollständigen Deformation des Problems nicht mehr passen.

Bei den einfachsten Varianten von Homotopieverfahren wird auf Informationen des angebundenen Lösers weitgehend verzichtet. Als Startwert des Korrektorlaufs wird jeweils das Ergebnis des vorangegangenen Schrittes verwendet, was einem Prädiktor nullter Ordnung entspricht. Die Schrittweiten werden aufgrund von Heuristiken bestimmt. Das einfachste Vorgehen ist eine Intervallbisektion, bei der die Schrittweite nach Erfolg des Korrektors verdoppelt, nach Misserfolgen halbiert wird. Als Abbruchkriterium können das Erreichen des Zieles, maximale Iterationszahl oder Misserfolg bei kleinster vorgegebener Schrittweite dienen. Eine Untersuchung weiterer Heuristiken zur Schrittweitensteuerung kann in (Kö05) gefunden werden, wo auch adaptive Steuerungen vorgeschlagen werden, die anhand der Anzahl der Iterationen bzw. der Rechenzeit des Korrektors die Schrittweite variieren.

Algorithmus 2 Homotopieverfahren (Prädiktor-Korrektor)

1: $\tau_0 = 0$
2: **while** $\tau \neq 1$ **do**
3: löse Optimalsteuerungsproblem zu Parameter τ_i
4: **if** Lösung gefunden **then**
5: neue Schrittweite $0 < \Delta\tau_i$
6: erneuere Startschätzung
7: **else**
8: neue Schrittweite $-\Delta\tau_{i-1} < \Delta\tau_i < 0$
9: **end if**
10: $\tau_{i+1} = \tau_i + \Delta\tau_i$
11: $i = i + 1$
12: **end while**

4.2 Dekomposition mit Branch-and-Bound

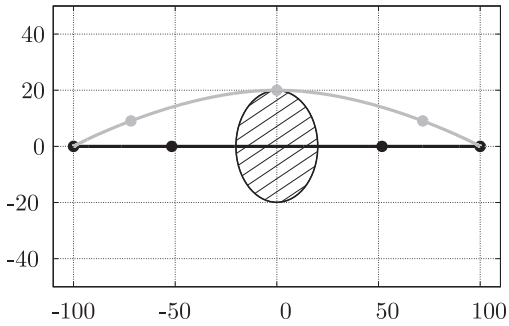


Abbildung 4.10: Informationsverlust durch Diskretisierung

Um eine optimale Wahl für die diskreten Variablen eines nichtlinearen gemischt diskret-kontinuierlichen Optimalsteuerungsproblems (2.1) bis (2.7) auf Seite 17 zu treffen, kann dieses durch totale oder sequentielle Diskretisierung in ein endlichdimensionales nichtlineares gemischt ganzzahliges Optimierungsproblem umgewandelt werden. Durch diese Vorgehensweise verändert sich allerdings der Lösungsraum des Problems, da nur noch zu diskreten Zeitpunkten Informationen vorliegen. Was zwischen diesen Zeitpunkten geschieht, ist unklar. So kann z. B. die Lösungstrajektorie, wie in Abbildung 4.10 gezeigt, durch einen unzulässigen Bereich verlaufen, ohne dass es im diskretisierten Fall bemerkt wird. Des Weiteren ist auf dieser Ebene nicht mehr klar, was für eine Bedeutung die einzelnen Variablen im Optimalsteuerungsproblem haben, ob sie zu einer Steuerung gehören oder einen Parameter darstellen. Dies sind jedoch Informationen, die zur Suche nach einem Optimum ausgenutzt werden können.

Für eine Optimierung, bei der das Problem nicht vorab durch Diskretisierung in ein endlichdimensionales gemischt ganzzahliges Optimierungsproblem transformiert wird, stehen lokale und globale Suchverfahren zur Verfügung. Die Suche erstreckt sich dabei nur auf den Bereich der diskreten Optimierungsvariablen. Zur Bestimmung der Kosten einer diskreten Möglichkeit wird das dazugehörige kontinuierliche Problem gelöst. Zu den lokalen Suchverfahren gehören zum einen Verfahren wie die *Nachbarschaftssuche, Tabu-Search* oder *Simulated Annealing*, die unter den direkten Nachbarn suchen. Hier liegt die Problematik darin, eine geeignete Nachbarschaft zu definieren. Auf der anderen Seite gibt es Verfahren wie genetische Algorithmen, die die Evolution nachahmen. Bei Lösungen, die mit diesen Verfahren gefunden werden, kann allerdings keine Aussage über deren Güte gemacht werden.

Die Methode des Branch-and-Bound ist eine sehr allgemeine, unter gewissen Voraussetzungen deterministische Methode zur Suche nach einem Minimum über einen zulässigen Bereich $\mathbb{R}^0$, wie hier gegeben durch die Nebenbedingungen (2.2) bis (2.7) auf Seite 17 eines gemischt diskret-kontinuierlichen Optimalsteuerungsproblems (SG00; SG01). Das Verfahren startet damit, eine obere Schranke O für das Optimierungsproblem, z. B. die Kosten eines beliebigen zulässigen Steuerprozesses, zu berechnen. Falls die Bestimmung eines zulässigen Punktes nicht ohne Weiteres möglich ist, kann auch mit $O = \infty$ gestartet werden. Die obere Schranke stellt nun die Kosten der besten bisher bekannten Lösung dar. Im weiteren Verlauf wird sukzessive der zulässige Bereich $\mathbb{R}^0$ unterteilt, wodurch ein Baum entsteht. Die Menge $\mathbb{R}^0$ stellt dabei seine Wurzel dar, und in den Blättern sind die einzelnen diskreten Möglichkeiten zu finden.

Die entstehenden Teilmengen $\mathbb{R}^i_0, \ldots, \mathbb{R}^i_{n_i}$ seien paarweise disjunkt, $\mathbb{R}^i_k \cap \mathbb{R}^i_l = \emptyset$ für $k \neq l$ und ihre Vereinigung sei in der ursprüngliche Menge $\mathbb{R}^{i-1}_j \supseteq \mathbb{R}^i_0 \cup \cdots \cup \mathbb{R}^i_{n_i}$ enthalten. Im Fall gemischt binär-kontinuierlicher Probleme (Abschnitt 3.3) kann in die zwei Mengen $\mathbb{R}^i_0$ und $\mathbb{R}^i_1$ verzweigt

werden, wobei $\mathbb{R}_0^i$ die Menge aller Probleme darstellt, bei denen eine der binären Variablen q_{b_k} gleich null ist, und $\mathbb{R}_1^i$ die, bei denen diese eins ist.

Für jeden der Teile $\mathbb{R}_j^i$ wird durch Lösung eines Ersatzproblems eine untere Schranke U_j^i bestimmt, welche die Kosten aller in diesem Teil enthaltenen Lösungskandidaten nach unten abschätzt. Ist nun für einen der Teile die untere Schranke größer als die Kosten der besten bisher bekannten Lösung, also $U_j^i > O$, so kann sich die optimale Lösung nicht in diesem Teil $\mathbb{R}_j^i$ befinden, weswegen er nicht weiter zerteilt und untersucht wird. Sowie eine für das ursprüngliche Problem zulässige Lösung gefunden wird, deren Kosten niedriger sind als die obere Schranke, wird die beste bisher bekannte Lösung durch diese ersetzt. Das Vorgehen wird solange wiederholt, bis der gesamte Suchraum exploriert ist. Die beste bekannte Lösung stellt dann die optimale Lösung dar. Diese ist global, falls alle im Verlauf berechneten Schranken global waren.

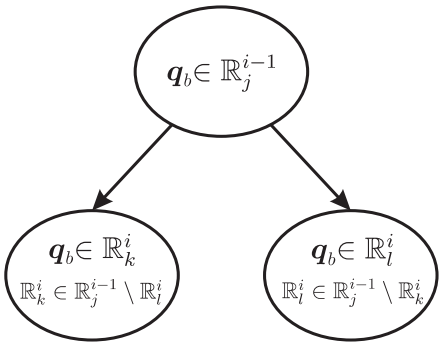


Abbildung 4.11: Verzweigung bei Branch-and-Bound

Algorithmus 3 Branch-and-Bound

```
1:  bestimme obere Schranke O
2:  while noch Knoten da do
3:      wähle Knoten
4:      bound: berechne untere Schranke U
5:      if [U > O] ∨ [es gibt keine Lösung] then
6:          entferne Knoten
7:      else if neue Lösung then
8:          verschiebe Knoten in Lösungsmenge
9:          O = U
10:     else if verzweigbar then
11:         branch: verzweige Knoten
12:     end if
13: end while
```

4.2.1 Traversion des Suchbaums

Der grundlegende Ablauf des Branch-and-Bound-Verfahrens ist zusammengefasst in Algorithmus 3, dessen einzelne Abschnitte im Folgenden spezifiziert werden.

Schrankenberechnung

Zur Berechnung unterer Schranken können Ersatzprobleme erzeugt werden, indem der zulässige Bereich des Problems, z. B. durch eine *LP-Relaxierung*, vergrößert wird. Bei einer LP-Relaxierung wird der Bereich zulässiger diskreter Werte, die noch nicht fixiert sind, auf deren konvexe Hülle

erweitert. Bei binären Parametern $\boldsymbol{q}_b \in \{0,1\}$ bedeutet dies eine Erweiterung auf das Intervall $[0,1]$. Durch die Relaxierung stellen die Ersatzprobleme rein kontinuierliche Optimalsteuerungsprobleme dar, die mit den Methoden aus Abschnitt 4.1 auf Seite 54 gelöst werden können. Die Ersatzprobleme haben im Gegensatz zum ursprünglichen Problem unter Umständen keinen physikalischen Sinn, was auch nicht notwendig ist, solange sie im mathematischen Sinne korrekt gestellt sind.

Die numerische Berechnung allein durch Relaxierung erzeugter Ersatzprobleme liefert im Allgemeinen keine globalen unteren Schranken, da die Probleme im Allgemeinen nach wie vor nicht konvex sind. Damit liefert auch das Branch-and-Bound-Verfahren nur eine lokal optimale Lösung. Zur Generierung globaler unterer Schranken können z. B. Screening-Modelle (AB97) oder globale Lösungsverfahren zum Einsatz kommen. Screening-Modelle sind allerdings problemspezifisch, so dass sie nicht ohne spezielles Wissen verwendet werden können, und globale Lösungsverfahren gibt es bisher nur für spezielle Klassen von Optimalsteuerungsproblemen (Rub98).

Anstelle der LP-Relaxierung besteht auch die Möglichkeit des Einsatzes von *Schnittebenenverfahren*. Der nichtlineare Fall ist in Analogie zum linearen zu sehen, wobei darauf hingewiesen sei, dass wegen der Nichtkonvexität auch hier keine globalen unteren Schranken erzeugt werden. Auch Informationen des *dualen* Problems können dazu verwendet werden, unteren Schranken zu berechnen.

Strategien bei der Knotenwahl

Es gibt viele verschiedene Strategien, welcher Teil des zulässigen Bereichs jeweils als nächster untersucht werden soll. Die wohl bekanntesten Strategien sind *Breitensuche*, *Tiefensuche* und die *kleinste-Schranken-Strategie*. Bei der Breitensuche wird versucht, den Baum, der beim Aufspalten entsteht, möglichst breit zu halten, es wird also ein Knoten für die Verzweigung gewählt, der der Wurzel des Baums möglichst nah ist. Dadurch wird auch der zulässige Bereich des gemischt ganzzahligen Problems in etwa gleichen Teilen abgedeckt, und es ergibt sich ein guter Überblick über die verschiedenen Regionen des zulässigen Bereichs.

Bei der Tiefensuche hingegen wird versucht, möglichst schnell tief in dem Baum vorzudringen, was einer lokalen Suche im zulässigen Bereich gleichkommt. Dazu wird ein Knoten mit möglichst kurzem Weg zu den Blättern gewählt. Die Hoffnung dabei ist, schnell eine neue oder erste zulässige Lösung zu finden, die die aktuelle obere Schranke verbessert, denn je niedriger die obere Schranke ist, desto schneller wird die Menge der Lösungskandidaten eingeschränkt.

Die Kleinste-Schranken-Strategie untersucht immer als nächstes die Teilmenge bzw. den Knoten, die die kleinste untere Schranke besitzt. So wird unter Umständen als erstes in einem vielversprechenden Bereich gesucht.

Eine weitere Wahlmöglichkeit wäre ein Knoten, bei dessen Lösung die relaxierten Variablen möglichst nahe bei null oder eins liegen und dessen untere Schranke möglichst tief ist. Das Ziel ist hier, dass das Festsetzen der Variablen die untere Schranke nicht mehr zu stark ansteigen lässt und dadurch eine möglichst tiefe obere Schranke gefunden wird.

Liefert eine Variante mehrere gleichwertige Kandidaten, so kann unter diesen mit einer der anderen Strategien ausgewählt werden.

Keiner der genannten Strategien kann ohne Weiteres der Vorzug gegeben werden. Die Breitensuche und die Kleinste-Schranken-Strategie kann als globale Suchmethode betrachtet werden, die ein grobes Überblickswissen liefert. Die Tiefensuche kommt hingegen einer lokalen, feineren Suche gleich. Meist ist ein Abwechseln der Strategien sinnvoll. Liegt ein grober Überblick vor, kann ein Ast des Suchbaumes für die Tiefensuche ausgewählt werden. Bei der Suche auf diesem Ast kann wiederum die Suchmethode variiert werden. Die Erfahrung zeigt (Glo00), dass die unteren Schranken im oberen Bereich des Baums so tief und damit so wenig aussagekräftig sind, dass im Wesentlichen erst im unteren Teil echte Fortschritte bei der Suche erzielt werden.

Verzweigungsheuristiken

Des Weiteren besteht noch die Frage, wie ein Teilbereich weiter unterteilt werden soll, um besonders gute Fortschritte zu machen. Im relaxierten gemischt binär-kontinuierlichen Fall entspricht dies der Auswahl einer relaxierten Variablen, die im Folgenden auf null bzw. eins gesetzt wird, und damit zwei neue Teilbereiche liefert, die in die Liste der noch zu untersuchenden Probleme aufgenommen werden.

Eine Variante zur Wahl der Variable, die als nächstes verzweigt werden soll, liefern benutzerdefinierte *Prioritäten*. Damit verlagert sich zwar das Problem einer geeigneten Wahl auf den Benutzer, jedoch hat dieser aus dem Modellierungsprozess heraus das Wissen über die Bedeutung der binären Variablen. Hier können z. B. zusätzliche Variablen eingeführt werden, die es ermöglichen, den zulässigen Bereich in günstige Teilbereiche aufzuspalten. Diese bekommen dann eine höhere Priorität als andere. Falls bekannt ist, dass die Fixierung bestimmter Variablen das Problem in so stark unterschiedliche Teile teilt, dass lange Rechenzeiten zu erwarten sind, sollten diese Verzweigungen möglichst selten, sprich möglichst nahe der Wurzel des Baums stattfinden.

Eine weitere Variante zur Vergabe von Prioritäten ist eine phasenspezifische. Auf diese Weise können alle Möglichkeiten einer einzelnen Phase untersucht werden, wobei die anderen Phasen relaxiert bleiben. Alternativ können die Variablen auch so priorisiert werden, dass möglichst schnell alle Phasen auf eine spezielle Wahl festgesetzt werden.

Bei Variablen, über die kein Wissen darüber vorhanden ist, ob sie bevorzugt behandelt werden sollen oder nicht, wird häufig die Variable gewählt, die bei der Lösung des relaxierten Problems am weitesten von ihren Grenzen entfernt liegt. Das Ziel hierbei ist, dass die Fixierung dieser Variablen eine möglichst große Änderung in den Kosten hervorruft. Meist ist eine derartige Wahl nicht besser als eine zufällige.

Besser geeignet sind Strategien, die auf *Pseudokosten* (vgl. (ASF99)) basieren. Angenommen, J^* sind die Kosten, die entstehen, falls alle Variablen relaxiert sind, und ${q_b}^*$ der Wert einer der relaxierten Variablen. Es seien nun J^0 diejenigen Kosten die entstehen, falls die Variable gleich null und J^1 diejenigen, falls sie gleich eins gesetzt wird. Die Pseudokosten ergeben sich für ein Verzweigen auf null zu $(J^0 - J^*)/{q_b}^*$ und für ein Verzweigen auf eins zu $(J^1 - J^*)/(1 - {q_b}^*)$.

Die Fälle ${q_b}^* = 0$ bzw. ${q_b}^* = 1$ können ausgeschlossen werden, da nicht mehr auf null bzw. eins verzweigt werden muss. Die Variable mit den höchsten Pseudokosten wird für die Verzweigung gewählt, da hier eine möglichst große Änderung in den tatsächlichen Kosten zu erwarten ist. Die Berechnung der Pseudokosten ist relativ aufwändig, da für jeden im Problem befindlichen binären Parameter und jede mögliche binäre Schaltung zwei Optimalsteuerungsprobleme gelöst werden müssen.

Eine weitere Möglichkeit, die Kostenentwicklung bei der Variation eines der relaxierten Parameter q_{bl} über dem zulässigen Bereich $\mathbb{R}_j^i$ abzuschätzen, ergibt sich durch die Betrachtung des optimalen Zielfunktionswerts als Funktion eben dieses Parameters

$$c(q_b) = \min_{\mathbb{R}_j^i, q_{bl}=q_b} J[\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{q}_b].$$

Die untere Schranke für die Probleme aus $\mathbb{R}_j^i$ sei $U_j^i = c({q_b}^*)$. Eine Taylorentwicklung um ${q_b}^*$ liefert

$$c({q_b}^* + \Delta q_b) = c({q_b}^*) + \frac{\mathrm{d}\, c}{\mathrm{d}\, q_b}\bigg|_{{q_b}^*} \Delta q_b + \frac{1}{2}\,\frac{\mathrm{d}^2\, c}{\mathrm{d}\, {q_b}^2}\bigg|_{{q_b}^*} \Delta {q_b}^2 + O(\Delta {q_b}^3).$$

Da $c({q_b}^*)$ ein lokales Minimum ist, gilt $\frac{\mathrm{d}\, c}{\mathrm{d}\, q_b}\Big|_{{q_b}^*} = 0$ für ${q_b}^* \in (0, 1)$, weswegen die Taylorentwicklung nicht vor dem Term zweiter Ordnung abgebrochen werden sollte. Falls alle weiteren Nebenbedingungen des Optimalsteuerungsproblems aus Abschnitt 3.3 erfüllt sind, kann der Kostenzuwachs durch

$$\Delta c(q_b) \approx \frac{\mathrm{d}\, c}{\mathrm{d}\, q_b}\bigg|_{{q_b}^*} (q_b - {q_b}^*) + \frac{1}{2}\,\frac{\mathrm{d}^2\, c}{\mathrm{d}\, {q_b}^2}\bigg|_{{q_b}^*} (q_b - {q_b}^*)^2$$

mit $q_b \in \{0, 1\}$ geschätzt werden. Es wird nun der Knoten gewählt, der den höchsten Zuwachs verspricht. Unter Umständen kann bei dem verwendeten Optimierungsverfahren auf die Ableitungen oder zumindest Näherungen von ihnen zugegriffen werden, falls diese zur Berechnung des Minimum verwendet werden.

Falls keine Ableitungen vorliegen, können die Ableitungen auch numerisch berechnet werden. Dazu können die ersten Schritte eines Homotopieverfahren genutzt werden. Da für gewöhnlich bei einer kleinen Änderung eines Parameters nur eine kleine Änderung der Lösung zu erwarten ist, kann diese schnell berechnet werden. Wird der Parameter wärend der Baumsuche komplett auf eins oder null gesetzt, kann der dadurch erzielte Kostenzuwachs relativ zur Parameteränderung als Pseudokosten für das Verzweigen dieses Parameters übernommen werden. Bevorzugt wird wiederum der Knoten mit höchsten Pseudokosten.

Falls vom Benutzer definierte Prioritäten vorliegen, sind diese den Pseudokosten vorzuziehen. Bei gleicher Priorität werden die Variablen mit größeren Pseudokosten gewählt.

Terminierung

Ein Abbruch eines Optimierungsverfahrens kann aus verschiedenen Gründen erfolgen, was im Branch-and-Bound-Verfahren berücksichtigt werden muss. Falls die optimale Lösung gefunden wurde, ist klar, dass der Knoten weiter verzweigt wird, oder er, falls dies nicht mehr möglich ist, einen Lösungskandidaten darstellt. Eine Meldung, der zulässige Bereich sei leer, führt dazu, dass der Knoten nicht weiter untersucht wird. Falls das relaxierte Problem nicht gelöst werden konnte, da es nicht nach unten beschränkt ist, kann es trotzdem im nicht relaxierten Fall eine Lösung besitzen. Der Knoten bleibt im Baum und wird weiter untersucht. Auch ein schlecht konditioniertes Problem kann unter Umständen leicht zu lösen sein, falls mehr Variablen fixiert sind. In jedem Fall sollte ein Branch-and-Bound-Verfahren dem Benutzer zu jedem Knoten mitteilen, warum er entfernt wurde bzw. mit welcher Meldung der Löser für das Ersatzproblem terminierte, um Knoten, bei denen es Probleme gab, mit geeigneten Mitteln neu zu untersuchen.

4.2.2 Verbesserung der Robustheit und Effizienz

Einsatz von Homotopietechniken: Schrittweise von 0 auf 1

Oft ist der Schritt zu groß den Wert einer relaxierten Variablen auf eine ihrer Schranken zu setzen. Die Lösung des darüber liegenden Knotens bildet einen zu schlechten Startwert, weswegen keine Konvergenz erzielt werden kann, oder das Kollokationsverfahren sehr viele Schritte bis zur Konvergenz benötigt. Hier können nun die in Abschnitt 4.1.4 auf Seite 70 erwähnten Homotopieverfahren zum Einsatz kommen. z^* sei der Wert einer relaxierten Variablen, den sie bei der Lösung des darüber liegenden Teilproblems hatte. Nun wird einmal die obere Schranke der Variablen $z_{upp} = (1-\tau)z^*$ gesetzt, wobei τ ein Homotopieparameter ist, um die Variable auf null zu drücken, und einmal die untere Schranke auf $z_{low} = (1-\tau)z^* + \tau$, um die Variable auf eins zu heben. Somit werden zwar insgesamt mehr Optimalsteuerungsprobleme gelöst, diese sind aber einfach und schnell lösbar, da bei geeigneter Wahl des Homotopieparameters durch das Ergebnis der vorhergehenden Lösung eine sehr gute Startschätzung vorliegt. Durch den Einsatz von Homotopietechniken verhält sich das Branch-and-Bound-Verfahren robuster, da kaum Knoten unter der falschen Annahme, das Problem sei unzulässig oder nicht lösbar, entfernt werden.

Auf gleiche Weise könnten bei einem Schnittebenen-Verfahren die Schnitte schrittweise eingefügt werden.

Bei dem beschriebenen Ansatz ist zu beachten, dass eine Nebenbedingung wie z. B. $2z_i + z_j \geq 2$ mit $z_j = 1$ für $z_i \in [0.5, 1]$ zulässig ist, für $z_i = 0$ aber nicht. Ein Versuch W_i auf null zu drücken, ist also sinnlos und würde nur zu unnötigen Rechenschritten führen. Hier sollte die Zulässigkeit der diskreten bzw. relaxierten Werte am Ende der Homotopieschritte vorab überprüft werden.

Heuristische Verfahren zur Generierung oberer Schranken

Einen sehr wichtigen Einfluss auf die Effizienz eines Branch-and-Bound-Verfahrens hat die erste obere Schranke, denn mit ihr werden alle unteren Schranken der Teilgebiete verglichen. Ist sie zu

groß, muss erst eine bessere gefunden werden, bevor Teile des zulässigen Bereichs von der weiteren Suche ausgeschlossen werden können. Obere Schranken sind im Allgemeinen erst tief unten im Baum zu finden. Hier können heuristische Verfahren, die direkt unter den diskreten Möglichkeiten suchen, verwendet werden. Die Kosten einer diskreten Möglichkeit werden durch die Lösung des dazugehörigen kontinuierlichen Problems berechnet. Somit wird relativ schnell eine gute Lösung gefunden, die dann schnell durch den Branch-and-Bound-Algorithmus verifiziert oder verbessert werden kann.

Hier ist der Einsatz von genetischen Algorithmen möglich (GS02). Bei genetischen Algorithmen wird zunächst eine gewisse Anfangspopulation optimaler Steuerprozesse zu verschiedenen Varianten der diskreten Parameter und Steuerungen erzeugt. Aus dieser *Population* werden zufällig Eltern gezogen und deren diskrete Parameter und Steuerungen gekreuzt, woraus ein neues Element (Kind) entsteht. Um neue Informationen in die Population einzubringen, wird eine zufällige Änderung an dem Kind vorgenommen. Schließlich wird es durch Lösen des dazugehörigen kontinuierlichen Optimalsteuerungsproblems bewertet und der Population hinzugefügt. Diese Vorgehensweise wird eine gewisse Zeit lang wiederholt, und das beste Mitglied der Population wird als Lösung ausgegeben.

Zusammenfassung

Branch-and-Bound-Verfahren bieten eine sehr allgemeine Vorgehensweise, die sich auf viele globale Optimierungsprobleme anwenden lässt. In dieser Arbeit wird nur der Raum der binären Variablen vom Branch-and-Bound-Verfahren nach einer Lösung durchsucht, bei den kontinuierlichen Teilproblemen wird mit lokalen Minima vorlieb genommen. Dies kann unter Umständen dazu führen, dass falsche Entscheidungen bei der Baumsuche getroffen werden. Um eine global optimale Lösung des gemischt binär-kontinuierlichen Optimalsteuerungsproblems zu finden, kann die Branch-and-Bound-Suche auf Teilgebiete des zulässigen Bereichs für die kontinuierlichen Variablen ausgedehnt werden. An den Knoten des Suchbaums müssten dann untere Schranken mittels konvexer Unterschätzer berechnet werden.

Strategien zur Knotenwahl und Verzweigungsheuristiken zeigen erst bei größeren Suchbäumen ihre Stärken. Bei kleinen Bäumen arbeiten sie vergleichbar. Bei der Knotenwahl sollte eine Breitensuche oder kleinste Schrankenstrategie verwendet werden um einen groben globalen Überblick zu bekommen und anschließend mittels Tiefensuche ein Teilbereich genauer exploriert werden. Zur Variablenwahl eignen sich benutzerspezifische Prioritäten oder Pseudokosten am besten, da sie die meisten Informationen über das Optimierungsproblem verwenden.

Ein Verfahrensstart mit einer zu großen oberen Schranke (wie z. B. ∞) führt dazu, dass erst nach einer, unter Umständen langen Suche neue obere Schranken gefunden werden, die zur Verkleinerung des Suchbaums führen. Hier bietet sich eine Suche unter den zulässigen Kombinationen diskreter Variablen (Blättern des Baums) an, um eine erste obere Schranke zu berechnen. Die Zulässigkeit einer Kombination kann hier leicht durch die Gleichungs- und Ungleichungsbedingungen an die diskreten Variablen überprüft werden, ohne ein Optimierungsproblem zu lösen.

4.3 Dekomposition durch Zustandsdiskretisierung

Ähnlich dem Vorgehen der dynamischen Programmierung wird in diesem Abschnitt ein Ansatz verfolgt, bei dem der Zustandsraum diskretisiert wird (Glo04). Ein vergleichbarer Lösungsansatz wird in (BHS02) für ein spezielles Multiarm-Transportproblem verwendet. In der Menge der Lösungen zu dieser Diskretisierung wird durch ganzzahlige lineare Optimierung nach einer Approximation der optimalen Trajektorie gesucht.

4.3.1 Zustandsraumbetrachtung

Wie bereits in Abschnitt 2.3.2 auf Seite 32 erwähnt, basiert die dynamische Programmierung auf dem Bellman'schen Optimalitätsprinzip, das besagt, dass ein Prozess nur dann optimal sein kann, wenn von einem beliebigen Zeitpunkt an die Restkosten des Prozesses minimal sind. Entscheidend dabei ist, dass vergangene Entscheidungen nicht von den zukünftigen abhängen dürfen. Die Restkosten ab einem gegebenen Zeitpunkt t_i können durch Lösung eines Optimalsteuerungsproblems berechnet werden. Der Zustand zum Zeitpunkt t_i ist dabei der Anfangszustand.

Wird nun ein optimaler Steuerprozess betrachtet, so kann ein beliebiges Stück zwischen zwei Zeitpunkten t_i und t_j herausgeschnitten werden, das für sich wiederum einen optimalen Steuerprozess bildet. Dessen Randwerte sind durch Werte an den Schnittpunkten gegeben. Die Idee besteht darin, ein diskret-kontinuierliches Optimalsteuerungproblem geeignet zu zerlegen, und optimale Teilstücke des Steuerprozesses zu berechnen, die anschliessend zu einem optimalen Gesamtprozess zusammengesetzt werden.

Zustandsdiskretisierung

Betrachtet wird ein mehrphasiges gemischt binär-kontinuierliches Optimalsteuerungsproblem (vgl. Abschnitt 3.3 auf Seite 52), wie es sich aus geeigneter Modellierung ergibt. In jeder Phase gibt es verschiedene diskrete Möglichkeiten kontinuierlicher Optimalsteuerungsprobleme. Diese unterscheiden sich in den Randdaten, den Nebenbedingungen und den Zuständen, die stetig oder durch eine Transitionsbedingung bestimmt aus der vorigen Phase kommen bzw. in die nächste Phase gehen. Diese stetigen und durch Transition bestimmten Zustände koppeln die einzelnen Phasen zu einem Ganzen, weshalb eine Phase nicht für sich alleine optimiert werden kann.

Die Randbedingungen bilden, wie im oberen Teil von Abbildung 4.12 auf Seite 80 zu sehen ist, zu einem Zeitpunkt t_i und jedem möglichen Satz binärer Parameter $\boldsymbol{q}_{bj}$ einen Bereich $\mathbb{R}_x^{ij} := \left\{ \boldsymbol{x} \;\middle|\; \boldsymbol{r}(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{q}_{bj}, t_i) = 0 \right\}$, in dem sich die Zustände bei ihrem Übergang in die nächste Phase befinden müssen. Die Vereinigung all dieser Bereiche sei $\mathbb{R}_x^i = \bigcup_{\boldsymbol{q}_{bj} \in \mathbb{Q}} \mathbb{R}_x^{ij}$. Innerhalb dieses Bereichs können nun die Zustandsvariablen diskretisiert werden in

$$\bar{\mathbb{D}}_x^i := \left\{ \boldsymbol{x}_k \;\middle|\; \boldsymbol{r}(\boldsymbol{x}_k, \boldsymbol{p}, \boldsymbol{q}_{bj}, t_i) = 0,\ k = 1, \ldots, m_{xi} \right\} \subset \mathbb{R}_x^i.$$

Die Menge $\bar{\mathbb{D}}_x^i$ kann unter Berücksichtigung zulässiger Transitionen noch weiter eingeschränkt werden in

$$\mathbb{D}_x^{i-} := \left\{ \boldsymbol{x}_k \;\middle|\; \boldsymbol{j}_i(\boldsymbol{x}_k, \boldsymbol{x}_l, \hat{t}_i) = 0,\ \boldsymbol{x}_k, \boldsymbol{x}_l \in \bar{\mathbb{D}}_x^i \right\}$$

für zulässige Werte zur Zeit t^- unmittelbar vor dem Übergang in die nächste Phase bzw.

$$\mathbb{D}_x^{i+} := \left\{ \boldsymbol{x}_k \;\middle|\; \boldsymbol{j}_i(\boldsymbol{x}_l, \boldsymbol{x}_l, \hat{t}_i) = 0,\ \boldsymbol{x}_k, \boldsymbol{x}_l \in \bar{\mathbb{D}}_x^i \right\}$$

für zulässige Werte zur Zeit t^+ unmittelbar nach dem Übergang. Somit lassen sich zu jeder Kombination binärer Parameter $\boldsymbol{q}_{bj}$ die voneinander unabhängigen kontinuierlichen Optimalsteuerungsprobleme

$$J[\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{q}_{bj}] = \Phi_i(\boldsymbol{x}(t_i), \boldsymbol{p}, \boldsymbol{q}_{bj}, t_i) + \int_{t_{i-1}}^{t_i} L(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_{bj}, t) \mathrm{d}\, t \tag{4.8}$$

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_{bj}, t) && (4.9)\\
0 &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_{bj}, t) && (4.10)\\
0 &\geq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{q}_{bj}, t) && (4.11)\\
&\boldsymbol{x}(t_{i-1}) \in \mathbb{D}_x^{(i-1)+} && (4.12)\\
&\boldsymbol{x}(t_i) \in \mathbb{D}_x^{i-} && (4.13)
\end{aligned}$$

formulieren (Abbildung 4.12 unten rechts), und daraus jeweils die dazugehörigen Kosten c_{ij} berechnen. Unter Umständen kommen einige Phasen im Lauf der Zeit mehrfach vor. Für diese Phasen muss, falls sie in autonomer Form vorliegen, der Diskretisierungsprozess nur einmal durchgeführt werden. Die Ergebnisse können dann, da sie vom Startzeitpunkt unabhängig sind, immer wieder verwendet werden.

Aufbau eines Graphen

Werden die Mengen $\mathbb{D}_x^i := \mathbb{D}_x^{i-} \cup \mathbb{D}_x^{i+}$ betrachtet, so korrespondieren sie mit den Knoten des Phasengraphen $G_P = (V_P, E_P)$ des Automaten, der dem Problem zugrundeliegt (vgl. Definition 7 auf Seite 44). Anstelle dieser Knoten werden jetzt die gesamten Mengen $\mathbb{D}_x^i$ und deren Elemente als Knoten eines neuen Graphen $G_D = (V_D, E_D)$ mit $V_D = \bigcup_i \mathbb{D}_x^i$ und $E_D = V_D \times V_D$ verwendet. Es wird angenommen, dass die Knoten aus V_D durchnummeriert sind, und es wird kurz $i \in V_D$ zur Bezeichnung der Knoten verwendet. Die Kanten werden über ihren Anfangsknoten i und Endknoten j durch ij bezeichnet. Eine Kante des Phasengraphen entspricht einem *Bündel* von Kanten $ij \in E_D$.

Die Kanten ij dieses neuen Graphen entsprechen den Optimalsteuerungsproblemen, die durch die Gleichungen (4.8) bis (4.13) auf dieser Seite gegeben sind, und werden mit deren Kosten c_{ij} gewichtet. Falls keine Translationsbedingungen vorliegen, entsteht ein k-partiter Graph, ansonsten sind die Verbindungen zwischen den Knoten der Mengen $\mathbb{D}_x^{i-}$ und $\mathbb{D}_x^{i+}$, die nicht disjunkt sein müssen, durch die Transitionsbedingungen gegeben. Die Gewichte der Kanten entsprechen den Schaltkosten. Um nun eine Tour in diesem Graphen darzustellen, werden die Kanten ij mit den

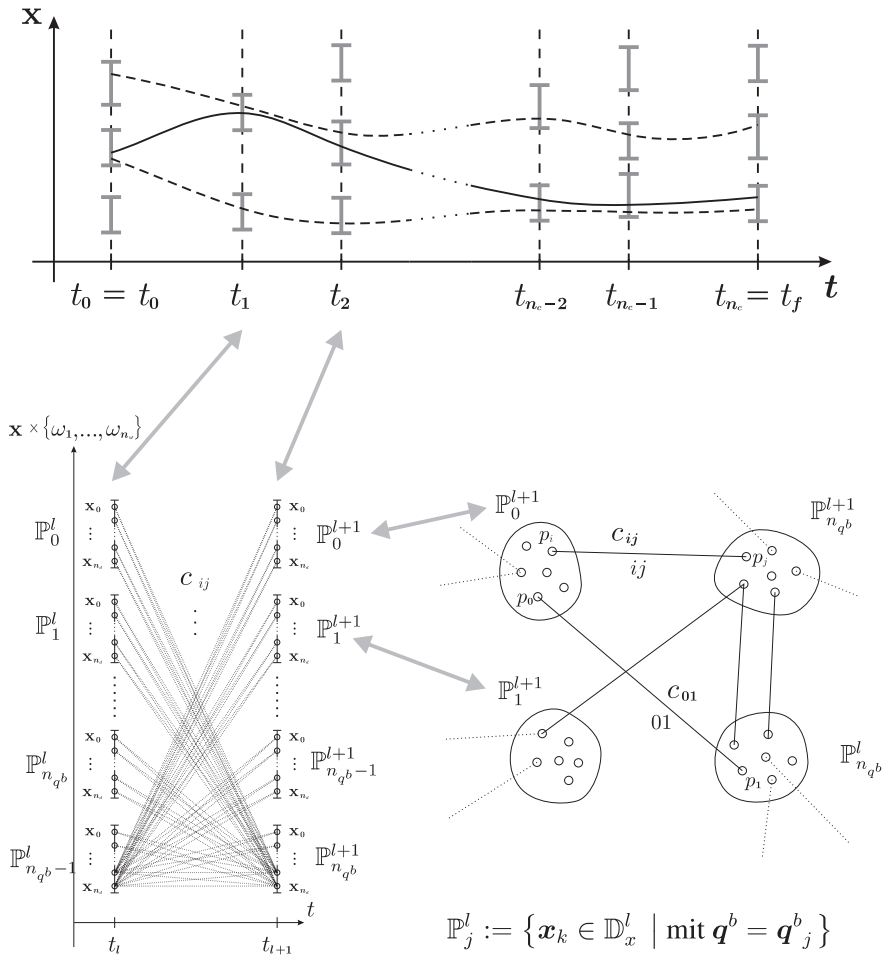


Abbildung 4.12: Dekomposition durch Zustandsdiskretisierung an inneren Punkten

neuen binären Variablen ω_{ij} versehen, wobei $\omega_{ij} = 1$, falls die Kante ij Teil der Tour ist, und $\omega_{ij} = 0$ in allen anderen Fällen.

Die Menge der Touren, die Approximationen zulässiger Lösungen des gemischt diskret-kontinuierlichen Optimalsteuerungsproblems ergeben, lassen sich nun durch lineare Gleichungs- und Ungleichungsbedingungen an die binären Variablen ω_{ij} beschreiben. Somit ergibt sich eine Näherung der Lösung des nichtlinearen gemischt diskret-kontinuierlichen Optimalsteuerungsproblems als Lösung des *linearen ganzzahligen Programms*

$$\begin{aligned} \min \quad & \sum_{ij \in E_D} \omega_{ij} c_{ij} \\ \text{unter} \quad & \sum_{ij \in G_k} \omega_{ij} a_{ijk} = \bar{a}_k, \; k = 1, \ldots, n_G \\ & \sum_{ij \in U_k} \omega_{ij} b_{ijk} \leq \bar{b}_k, \; k = 1, \ldots, n_U \end{aligned} \tag{4.14}$$

mit $a_{ijk}, b_{ijk}, \bar{a}_k, b_k \in \mathbb{R}$. Die Nebenbedingungen von Problem (4.14) lassen sich in zwei Gruppen aufteilen. Zum einen die Nebenbedingungen, die dafür sorgen, dass eine zulässige Tour durch den Phasengraphen gefunden wird, und zum anderen die Nebenbedingungen, die sicherstellen, dass aus den zu den Knoten des Phasengraphen zugeordneten Bündel solche Kanten ausgewählt werden,

die die Stetigkeits- und die Transitionsbedingungen erfüllen.

Fehlerabschätzung und effiziente Berechnung

Um genügend genaue Approximationen zu bekommen, stellt sich die Frage, wie die Diskretisierung gewählt werden soll, und wie genau die berechnete Lösung ist. Dazu werden zwei Kanten aus E_D als benachbart bezeichnet, falls sich die dazugehörigen Optimalsteuerungsprobleme in nur genau einer Komponente x_k der durch die Diskretisierung festgesetzten Randwerte $\boldsymbol{x}(t_{i-1})$ und $\boldsymbol{x}(t_i)$ unterscheiden, und diese Komponente zwei benachbarte Werte der Diskretisierung von x_k enthält. Eine Diskretisierung sollte so geartet sein, dass die Kostendifferenz von je zwei benachbarten Kanten etwa gleich groß ist.

Zur Berechnung der Kosten der einzelnen Kanten wird ähnlich wie bei Homotopieverfahren die Lösung des nächsten Nachbarn, sprich einer benachbarten Kante, als Startschätzung verwendet. Damit kann die Differenz der Kosten dieser beiden benachbarten Kanten sofort verglichen und bei zu großer Differenz weiter unterteilt werden. Ist die Differenz hingegen kleiner einer vorgegebenen Größe, kann der nächste Nachbar weiter entfernt gewählt werden. Somit wird das Gitter adaptiv festgelegt.

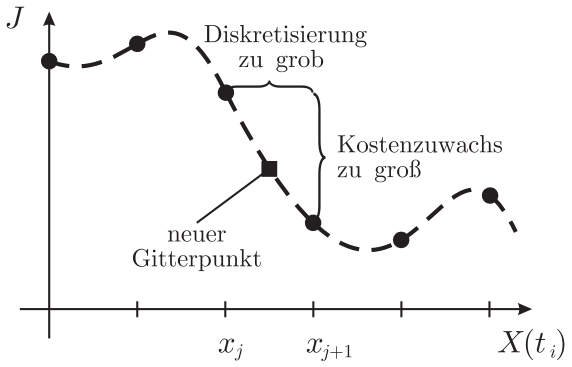


Abbildung 4.13: Adaptives Gitter für die Zustandsdiskretisierung

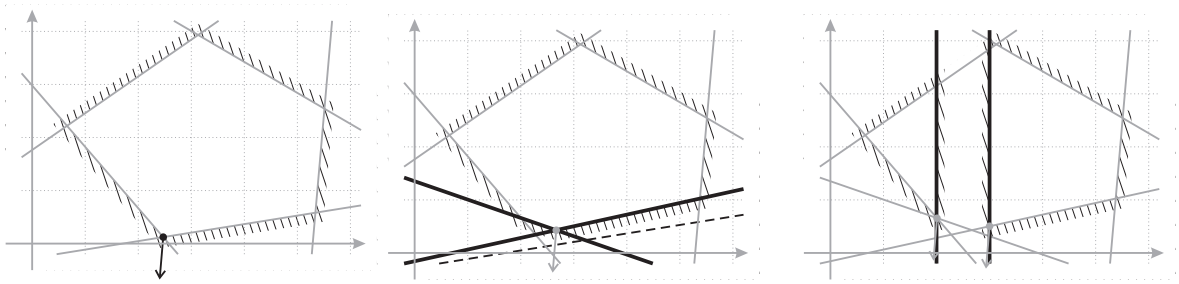
Die Differenz der Kosten relativ zum Abstand zweier Nachbarn ist eine Approximation für die Sensitivität der optimalen Lösung bezüglich des variierten Zustands. Ist die Sensitivität bekannt, so kann sie verwendet werden, um die Kosten in erster Näherung zu bestimmen und daraus den Abstand zum nächsten Diskretisierungspunkt abzuschätzen (vgl. Abbildung 4.13).

4.3.2 Branch-and-Cut

Gemischt ganzzahlige lineare Programme besitzen meist eine enorme Anzahl von Variablen und Nebenbedingungen. Im Lösungspunkt sind im Allgemeinen nur wenige Nebenbedingungen aktiv und viele Variablen gleich null, so dass sie nicht zu den Kosten beitragen. Die inaktiven Nebenbedingungen können daher weggelassen werden. Dieser Umstand wird bei Branch-and-Cut-Algorithmen genutzt.

Algorithmus

Das Branch-and-Cut-Verfahren (siehe z. B. (Mar01)) funktioniert wie ein Branch-and-Bound-Verfahren (vgl. Abschnitt 4.2 auf Seite 71), jedoch wird zur Berechnung unterer Schranken ein Schnittebenenverfahren eingesetzt. Dabei wird zunächst ein Problem mit wenigen Nebenbedingungen



(a) Optimierung (b) Cut (c) Branch

Abbildung 4.14: Branch-and-Cut-Algorithmus

formuliert, dessen konvexer zulässiger Bereich den des eigentlich zu lösenden Problems enthält. Über diesen Bereich wird nun minimiert (Abbildung 4.14(a)), wodurch ein Punkt berechnet wird, dessen Kosten eine untere Schranke für das ursprüngliche Problem, und die beste (d. h. größte), bisher bekannte untere Schranke für den gerade untersuchten Teilbereich darstellen. Nun wird überprüft, ob dieser Punkt eine Lösung des ursprünglichen Problems ist. Falls dies nicht der Fall ist, werden Ungleichungen gesucht, die diesen Punkt abtrennen, jedoch nichts vom zulässigen Bereich des ursprünglichen Problems (Abbildung 4.14(b)). Das Problem, solche Ungleichungen zu finden, wird *Separierungsproblem* genannt. Alte Ungleichungen, die durch diese neuen überflüssig werden, können für den weiteren Verlauf weggelassen werden. Eine erneute Minimierung liefert einen neuen Punkt, für den wiederum überprüft werden muss, ob er eine Lösung darstellt. Damit wird eine Folge wachsender unterer Schranken erzeugt.

Stellt ein gefundener Punkt keine Lösung dar, und es werden keine Nebenbedingungen gefunden, die diesen Punkt separieren, so wird der zulässige Bereich wie bei Branch-and-Bound unterteilt (Abbildung 4.14(c)) und ein Schnittebenenverfahren auf jeden der neu entstandenen Teile angewandt. Anhand der wachsenden Folgen unterer Schranken wird jeweils wie bei Branch-and-Bound entschieden, ob ein Teilbereich einen Lösungskandidaten enthalten kann oder nicht.

Dadurch dass nie alle Nebenbedingungen im aktuell zu lösenden Teilproblem enthalten sind, können diese im Allgemeinen schnell gelöst werden. Der Erfolg des Verfahrens hängt davon ab, wie gut das Separierungsproblem gelöst werden kann. Der beschriebene Algorithmus kann noch dahingehend erweitert werden, dass Variablen, die nicht in den Nebenbedingungen vorkommen, auch nicht als Variablen bei der Optimierung verwendet werden. Algorithmen, die Variablen und Nebenbedingungen dynamisch verwalten, werden *Branch-and-Cut-and-Price* -Verfahren genannt (LRJ01).

Schnittebenen-Verfahren

Im Branch-and-Cut-Verfahren werden eine Vielzahl linearer Programme gelöst. Im Wesentlichen gibt es dazu zwei Methoden: den Simplex-Algorithmus und Innere-Punkt-Verfahren. Innere-Punkt-Methoden basieren darauf, dass im linearen Fall die Lösung des primalen und dualen Problems übereinstimmen, so dass ein Newton-Verfahren verwendet werden kann, um die Optimalitätsbedingungen zu erfüllen, die um die Forderung der Gleichheit der Kosten eines primalen und dualen

Punktes erweitert sind. Weiter verbreitet ist das Simplexverfahren. Hier wird ausgenutzt, dass, wie in Abbildung 4.14(a) dargestellt, bei linearen Programmen eine Ecke des zulässigen Bereichs in der Menge optimaler Lösungen enthalten ist. Eine Ecke kann durch die aktiven Nebenbedingungen in Form eines *Simplextableaus* dargestellt werden. Umformungen dieses Tableaus entsprechen einer Wanderung von einer Ecke des zulässigen Bereichs zu einer anderen. Schnittebenenverfahren starten mit einem Problem, das nicht alle Nebenbedingungen enthält, und somit auch weniger Ecken besitzt und schnell gelöst werden kann. Durch Hinzufügen einer Schnittebene verliert der eben gefundene Punkt seine Zulässigkeit (vgl. Abbildung 4.14(b)). Hier kommt nun ein duales Simplexverfahren zum Einsatz, sprich das duale Problem wird gelöst. Die Optimalität des dualen Problems liefert einen primal zulässigen Punkt. Bei dem dualen Problem handelt es sich um ein Maximierungsproblem. Seine numerische Lösung ergibt eine Folge wachsender unterer Schranken, was im Branch-and-Cut-Verfahren ausgenutzt werden kann, um möglichst früh zu entscheiden, ob ein Knoten weiter untersucht wird oder nicht. Für eine Vertiefung dieser Vorgehensweise sei auf Bücher wie z. B. (Sch86; Sch03) verwiesen.

Algorithmus 4 Dekomposition

```
1: erzeuge adaptives Gitter analog Abschnitt 4.3.1 und berechne dazugehörige Kosten durch
   Lösen der Optimalsteuerungsprobleme
2: erzeuge kleines Teilproblem von (4.14) mit wenigen Nebenbedingungen
3: while noch Knoten da do
4:    wähle Knoten
5:    while es gibt eine verletzte Nebenbedingung do
6:       füge verletzte Nebenbedingung hinzu
7:       entferne überflüssige Nebenbedingungen
8:       berechne neue untere Schranke U
9:    end while
10:   if [U > O] ∨ [es gibt keine Lösung] then
11:      entferne Knoten
12:   else if neue Lösung then
13:      verschiebe Knoten in Lösungsmenge
14:      O = U
15:   else if verzweigbar then
16:      branch: verzweige Knoten
17:   end if
18: end while
```

4.4 Zusammenfassung

Die Verknüpfung eines Branch-and-Bound-Verfahrens mit einem direkten Kollokationsverfahren bildet ein Mehrzweckverfahren für gemischt binär kontinuierliche Optimalsteuerungsprobleme. Allerdings können die Teilprobleme, die durch teilweise Relaxierung der binären Parameter entstehen, äußerst komplex werden, da alle berücksichtigten diskreten Varianten und Verläufe des

betrachteten Systems gleichzeitig in der Formulierung enthalten sein müssen. Dies führt zu hochdimensionalen Optimalsteuerungsproblemen mit vielen Phasen. Bei einer Erweiterung des Problems um zusätzliche Zustände oder Schaltpunkte muss die Baumsuche wieder von der Wurzel starten und bereits berechnete Werte können nicht ohne Weiteres wiederverwendet werden. Die Kopplung der Phasen durch stetige Übergängen oder Transitionsbedingungen stellt hier keine großen Probleme dar.

Die Dekomposition des Problems durch Zustandsdiskretisierung an inneren Punkten hingegen bietet sich nur für Problemstellungen an, bei denen die Phasen durch wenige (≤ 5) stetige Übergänge oder Transitionen der Zustandsvariablen gekoppelt sind. Dafür stellt eine große Anzahl von Schaltungen zwischen wiederkehrenden diskreten Systemzuständen kein Problem dar, da die Berechnungen die zu einer Phase gehören immer wieder verwendet werden können. Bei einer Erweiterung der Problemstellung um zusätzliche diskrete Zustände oder Schaltpunkte können alle bisherigen Berechnungen weiter verwendet werden und der hinzukommende Rechenaufwand hält sich in Grenzen.

Für beide Ansätze muss eine Vielzahl von kontinuierlichen automatisch generierten Teilproblemen gelöst werden. Hierfür sind Kollokationsverfahren aufgrund ihrer Robustheit und Geschwindigkeit besonders gut geeignet.

Kapitel 5

Anwendungen bei Mehrfahrzeugsystemen

5.1 Simultane Reihenfolge- und Flugtrajektorienoptimierung

Bei der Luftverkehrsplanung gehen aktuelle Untersuchungen weg von den bisherigen Luftstraßen hin zur teilweisen oder gar komplett freien Nutzung des gesamten Luftraums (*free flight*). Dadurch ergeben sich zum einen Möglichkeiten der individuellen Optimierung von Flugbahnen durch die Betreiber einzelner Fluglinien, zum anderen wird es, insbesondere in Gebieten hohen Verkehrsaufkommens, schwieriger den Luftraum zu überwachen und die Sicherheit zu gewährleisten. Bereiche mit hohem Verkehrsaufkommen sind sicherlich in der Umgebung von Flughäfen zu finden. Ein interessanter Vorgang ist hier das Einreihen der aus der Freiflugphase kommenden Flugzeuge für die bevorstehende Landung, um den Flughafen optimal auszulasten. Eine optimale Reihenfolge hierfür kann und muss nicht nur unter der Berücksichtigung der dynamischen Eigenschaften der speziellen Flugzeuge, sondern auch unter dem Aspekt kollisionsfreier Flugbahnen gesucht werden.

Das Problem der Berechnung kollisionsfreier Flugtrajektorien wird häufig in der Literatur untersucht (MSS99; TPK$^+$98; PFB02; RGS$^+$04). In (MSS99) werden dafür Optimalsteuerungsprobleme formuliert, bei denen die Flugzeuge als Punktmassen mit realistischer Aerodynamik und Antriebsmodellen modelliert sind. Zur Lösung der Optimalsteuerungsprobleme kommen direkte Verfahren zum Einsatz. Für den Spezialfall von zwei Flugzeugen wird auch ein Regelkreis zur Konfliktvermeidung untersucht.

Auf vier Entscheidungsebenen wird das Kollisionsproblem in (TPK$^+$98) für Punkte in der Ebene gelöst. Die Betrachtung einer Ebene wird durch die Einteilung des Luftraums in einzelne Schichten begründet. Zunächst werden in einem strategischen Planer grobe konfliktfreie Trajektorien in Form von einzelnen Wegpunkten generiert, die je nach Situation angepasst und im taktischen Planer durch Interpolation verfeinert werden. Ein Trajektorienplaner erzeugt daraus schließlich Trajektorien für ein detailliertes dynamisches Modell unter Berücksichtigung von Sensordaten, die auf der Regelungsebene umgesetzt werden. Die Konfliktvermeidung auf oberer Ebene wird unter der Annahme unkooperativer Teilnehmer einmal für festgesetzte lineare Geschwindigkeit und einmal für den Geradeausflug betrachtet. Bei kooperierenden Teilnehmern kommen Potential und Wirbelfelder zur Auflösung der Konflikte zum Einsatz.

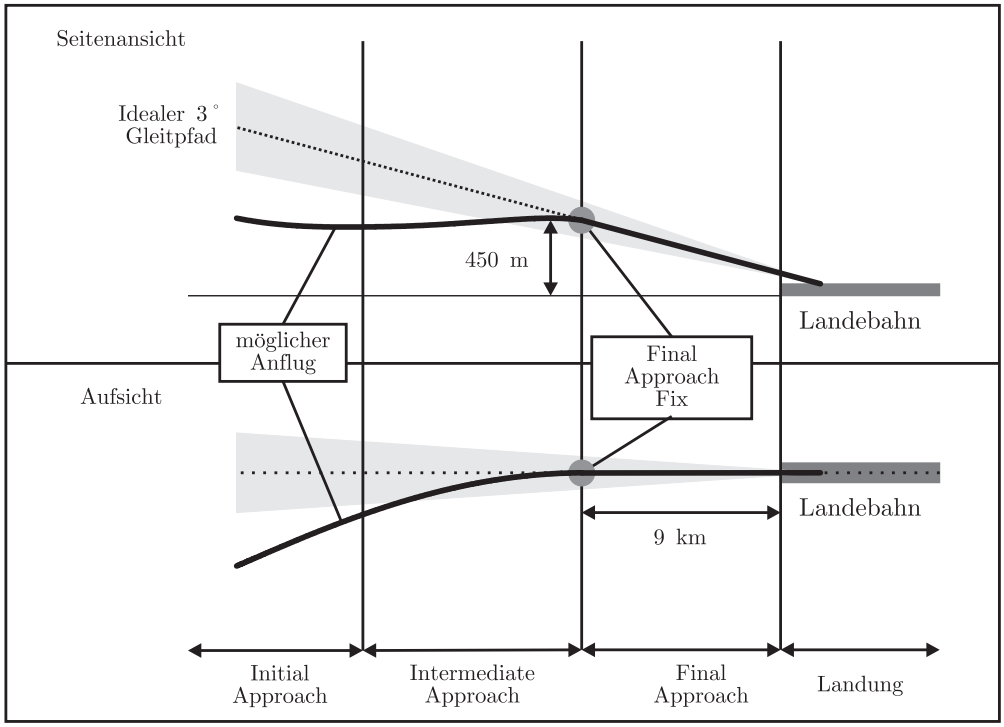


Abbildung 5.1: Phasen eines Landeanflugs mit Instrument Landing System

Ebenfalls in der Ebene wird die Problemstellung in (PFB02) für den Geradeausflug und Flug mit konstanter Geschwindigkeit bei instantanen Richtungänderungen untersucht. Für beide Varianten werden lineare gemischt ganzzahlige Optimierungsprobleme aufgestellt, die mit Standardsoftware gelöst werden.

Ein direktes Kollokationsverfahren wird in (RGS$^+$04) zu Berechnung konfliktfreier Trajektorien von Flugzeugmodellen mit realitätsnaher Dynamik eingesetzt. Um Lösungen für Probleme mit bis zu acht Flugzeugen zu finden, wird der heuristische Ansatz verfolgt, beginnend bei zwei Flugzeugen iterativ ein Flugzeug nach dem anderen dem Problem hinzuzufügen und die bereits berechneten konfliktfreien Trajektorien als fest vorgegeben zu betrachten. In einer anderen Variante dienen konfliktfreie Trajektorien aus einer Modellierung mit einfacher Dynamik als Ausgangspunkt zur Optimierung mit den komplexeren Modellen.

Die Optimierung der Landereihenfolge wird meist ohne gleichzeitige Optimierung der für das Einreihen notwendigen Trajektorien betrachtet. In (FFG$^+$03) wird das Reihenfolgeproblem für Flugzeuge verschiedenen Typs untersucht und als lineares ganzzahliges bzw. gemischt ganzzahliges Problem betrachtet. Zur Lösung der Probleme mit bis zu 123 Flugzeugen werden verschiedene heuristische und exakte Verfahren verglichen.

5.1.1 Landeanflug in der zivilen Luftfahrt

Betrachtet wird der ankommende Luftverkehr in der Umgebung eines Flughafens. Aktuell gibt es verschiedene Verfahren zur Landung von Flugzeugen. Eines der meist verbreiteten ist ein Landeanflug mit ILS (Instrument Landing System, vgl. (KLS03)). Dieser kann in fünf Segmente unterteilt

werden. Er beginnt mit der Einflugstrecke (*Arrival Route*), einer Luftstraße, welche die Flugzeuge zum *Initial Approach Fix* der zugewiesenen Landebahn leitet. Dort beginnt die *Initial Approach* zum *Intermediate Fix*, ab dem das Flugzeug auf der verlängerten Anflugrundlinie für die *Final Approach* stabilisiert wird. Wärend des Final Approach besteht die letzte Möglichkeit den Landeanflug abzubrechen, falls dies nicht geschieht endet der Anflug mit der Landung.

In dieser Arbeit wird von einem Szenario mit freier Luftraumnutzung ausgegangen, bei dem sich die Flugzeuge für den Landeanflug einreihen müssen. Der Flugraum sei bis zum *Final Approach Fix*, der etwa $9\,\text{km}$ von der Landebahn entfernt ist, frei nutzbar. Ab diesem Punkt verläuft die Landung nach strikten Regeln und lässt deshalb keinen Freiraum für Optimierung. Am Final Approach Fix sollten die Flugzeuge einen Abstiegswinkel ca. $3°$ und eine Höhe von etwa $450\,\text{m}$ über dem Boden haben. Bei zu grossen Abweichungen wird der Landeanflug abgebrochen und erneut versucht. Betrachtet werden nur Flugzeuge, die sich zum Anfangszeitpunkt innerhalb eines Radius von $150\,\text{km}$ um den Final Approach Fix befinden. Für diese Flugzeuge werden eine optimale Landereihenfolge und optimale Trajektorien hin zum Final Approach Fix berechnet.

Flugzeugmodellierung

In der zivilen Luftfahrt ist es weit verbreitet, die Luftfahrzeuge durch ein Punktmassemodell zu beschreiben. Unter Vernachlässigung der Rotation der Erde und ihrer Krümmung, kann die Bewegungsgleichung eines Flugzeugs wie folgt beschrieben werden (MSS99; RGS$^+$04).

$$\dot{x}_i = v_i \cos\gamma_i \cos\chi_i, \tag{5.1}$$
$$\dot{y}_i = v_i \cos\gamma_i \sin\chi_i, \tag{5.2}$$
$$\dot{h}_i = v_i \sin\gamma_i, \tag{5.3}$$
$$\dot{v}_i = \frac{(T_i - D_i)}{m_i} - g\sin\gamma_i, \tag{5.4}$$
$$\dot{\gamma}_i = \frac{g}{v_i}\left(\frac{L_i\cos\varphi_i}{g m_i} - \cos\gamma_i\right), \tag{5.5}$$
$$\dot{\chi}_i = \frac{L_i \sin\varphi_i}{m_i v_i \cos\gamma_i}, \tag{5.6}$$
$$\dot{m}_i = -Q_i, \tag{5.7}$$

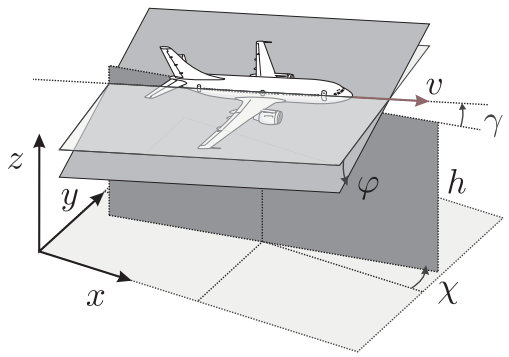


Abbildung 5.2: Flugzeugmodell

mit den Bezeichnungen aus Tabelle 5.1.

Der Vektor (x_i, y_i, h_i) gibt die Lage des Schwerpunkts des i-ten Flugzeugs relativ zum Final Approach Fix, der bei $(0, 0, 450)^T\,\text{m}$ liegt, an. Der Luftraum der in dieser Arbeit von Interesse ist kann eingeschränkt werden auf

$$-200\,\text{km} \leq x_i(t) \leq 200\,\text{km}, \tag{5.8}$$
$$-200\,\text{km} \leq y_i(t) \leq 200\,\text{km}, \tag{5.9}$$
$$0\,\text{m} \leq h_i(t) \leq 1200\,\text{m}, \tag{5.10}$$

Tabelle 5.1: Bezeichnungen bei der Modellierung der Flugzeuge

i	Index des Flugzeugs,	t	Zeit in s,
$x_i(t)$	x-Koordinate in m,	$T_i(t)$	Triebwerksschub in N,
$y_i(t)$	y-Koordinate in m,	$D_i(t)$	Luftwiderstand in N,
$h_i(t)$	Höhe in m über Boden,	$m_i(t)$	Masse des Flugzeugs in kg,
$v_i(t)$	Geschwindigkeit in $\mathrm{m\,s^{-1}}$	g	Fallbeschleunigung in $\mathrm{m\,s^{-2}}$,
	relativ zum Boden,	$L_i(t)$	aerodynamischer Auftrieb in N,
$\gamma_i(t)$	Abstiegswinkel in rad,	$Q_i(t)$	Flussrate des Treibstoffs in $\mathrm{kg\,s^{-1}}$,
$\chi_i(t)$	Flugbahnwinkel in rad,	$\phi_i(t)$	Schräglage in rad.

die weiteren Zustände des Flugzeugs auf

$$200\,\mathrm{km\,h^{-1}} \leq \quad v_i(t) \quad \leq 1000\,\mathrm{km\,h^{-1}}, \tag{5.11}$$

$$-\frac{\pi}{2} \leq \quad \gamma_i(t) \quad \leq \frac{\pi}{2}, \tag{5.12}$$

$$-\frac{\pi}{2} \leq \quad \chi_i(t) \quad \leq \frac{\pi}{2}. \tag{5.13}$$

In der vorliegenden Formulierung wird davon ausgegangen, dass die Geschwindigkeit relativ zum Boden gleich der Luftgeschwindigkeit ist und der Schub in Richtung des Geschwindigkeitsvektors wirkt. Die Flussgeschwindigkeit des Treibstoffs hängt für gewöhnlich von der Flughöhe, der Machzahl und dem Schub T_i ab. Da die vorliegende Betrachtung nur eine kurze Zeitspanne unmittelbar vor der Landung beinhaltet, wird der Treibstoffverbrauch vernachlässigt und die Masse als konstant betrachtet.

Als Steuerungen für das Flugzeug können direkt die Beschleunigung, die Abstiegswinkel- und die Flugbahnwinkelgeschwindigkeit genommen werden, womit sich (5.1) bis (5.6) zu

$$\begin{aligned}
\dot{x}_i(t) &= v_i(t)\cos\gamma_i(t)\cos\chi_i(t), & \dot{v}_i(t) &= u_{v,i}(t),\\
\dot{y}_i(t) &= v_i(t)\cos\gamma_i(t)\sin\chi_i(t), & \dot{\gamma}_i(t) &= u_{\gamma,i}(t),\\
\dot{h}_i(t) &= v_i(t)\sin\gamma_i(t), & \dot{\chi}_i(t) &= u_{\chi,i}(t),
\end{aligned}$$

vereinfachen. Aus diesen Größen können dann die Schräglage $\phi_i(t)$, die durch Quer- und Seitenruder eingestellt wird, der Triebwerksschub $T_i(t)$, der über den Schubhebel geregelt wird, und der aerodynamische Auftrieb $L_i(t)$ durch

$$\begin{aligned}
T_i(u_{v,i}(t), v_i(t), \gamma_i(t)) &= m_i u_{v,i}(t) + m_i g\cos\gamma_i(t) + D_i(v_i(t)),\\
L_i(u_{\chi,i}(t), u_{\gamma,i}(t), v_i(t), \gamma_i(t)) &= m_i\sqrt{(u_{\chi,i}(t)v_i(t)\cos\gamma_i(t))^2 + (u_{\gamma,i}(t)v_i(t) + g\cos\gamma_i(t))^2},\\
\varphi_i(u_{\chi,i}(t), u_{\gamma,i}(t), v_i(t), \gamma_i(t)) &= \arctan\left(\frac{u_{\chi,i}(t)v_i(t)\cos\gamma_i(t)}{u_{\gamma,i}(t)v_i(t) + g\cos\gamma_i(t)}\right),
\end{aligned}$$

Tabelle 5.2: Flugzeugdaten

Typ	Landegewicht m	maximaler Schub T_{max}	Flügelfläche A_i
B737	50t	700kN	105.40m^2
A310	120t	2300kN	219.00m^2
A330	180t	3000kN	361.60m^2
B747	260t	2700kN	510.97m^2
A380	380t	3400kN	845.00m^2

berechnet werden, was als Nebenbedingung in das Steuerungsproblem eingeht. Diese Nebenbedingungen werden durch

$$40\text{kN} \leq \quad T_i \quad \leq T_{i,max}, \tag{5.14}$$

$$0.9g \leq \quad \tfrac{L_i}{m_i} \quad \leq 1.1g, \tag{5.15}$$

$$-\frac{\pi}{2} \leq \quad \phi_i \quad \leq \frac{\pi}{2}, \tag{5.16}$$

$$\tag{5.17}$$

beschränkt. Der Luftwiderstand des Flugzeugs wird hier näherungsweise durch

$$D_i(v_i(t)) = 0.05\rho_l v_i^2(t) A_i$$

berechnet. Dabei wird von einer konstanten Luftdichte $\rho_l = 1\ \text{kg}\,\text{m}^{-3}$ ausgegengen. Je nach Flugzeugtyp unterscheiden sich das Gewicht, der maximale Schub $T_{i,max}$ und die Flügelfläche A_i. Als Gewicht wird hier das Landegewicht verwendet. Die Daten in Tabelle 5.2 sind in Anlehnung an die Flugzeuge A310, A330, A380 von Airbus sowie die B737 und B747 von Boeing gewählt. Die B737 wird als leichtes Flugzeug, die A310 und A330 als mittlere und die B747 und A380 als schwere Flugzeuge eingestuft.

Einreihen

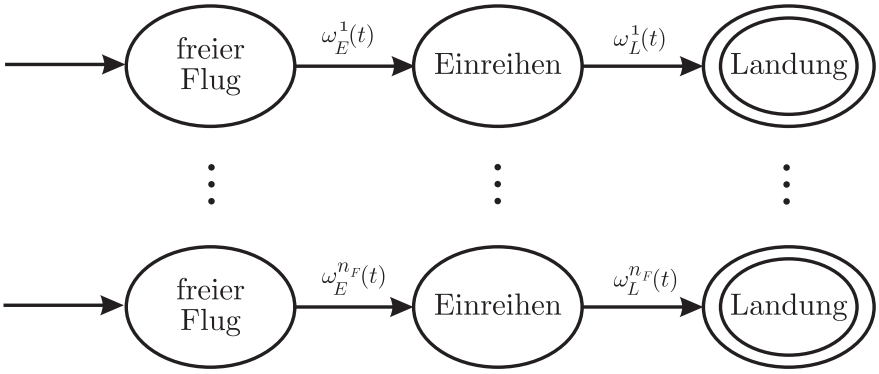


Abbildung 5.3: Hybride Automaten landendender Flugzeuge

Jedes der n_F betrachteten Flugzeug ist für sich ein eigenes System, das sich im Zustand des freien Flugs vor dem Final Approach Fix, dem Einreihen oder danach in dem des Landevorgangs befindet.

Diese Zustände der Automaten sind in Abbildung 5.3 dargestellt und werden mit den Variablen $w_{E,i} \in \{0,1\}$ und $w_{L,i}(t) \in \{0,1\}$ identifiziert, wobei

$$\begin{array}{rcll} w_{E,i}(t) & = & 1 & \leftrightarrow \text{ [Einreihen]} \\ w_{L,i}(t) & = & 1 & \leftrightarrow \text{ [Landung]} \\ w_{E,i}(t) & = & 0 \wedge w_{L,i}(t) = 0 & \leftrightarrow \text{ [freier Flug]} \end{array}$$

gilt. Ein Flieger kann zu jedem Zeitpunkt nur einen seiner Zustände annehmen was gleichbedeutend mit

$$w_{E,i}(t) + w_{L,i}(t) \leq 1, \quad i = 1, \ldots, n_F \tag{5.18}$$

ist. Desweiteren darf sich nur einer der Flieger einreihen, weswegen

$$\sum_{i=1}^{n_F} w_{E,i}(t) = 1 \tag{5.19}$$

gefordert wird. Hat sich der Flieger eingereiht, so folgt die Landung.

$$w_{E,i}(t-) \leq w_{L,i}(t+), \quad i = 1, \ldots, n_F. \tag{5.20}$$

Hier wird der Vorgang nach der Landung nicht weiter untersucht, weswegen der Flieger durch

$$w_{L,i}(t-) \leq w_{L,i}(t+), \quad i = 1, \ldots, n_F \tag{5.21}$$

in dem neutralen Zustand der Landung bis zum Ende der Untersuchung verweilt.

Final Approach Fix

Am Ende einer jeden Phase reiht sich einer der Flieger am Final Approach Fix ein und die Betrachtungen enden, wenn der letzte eingereiht ist. Insgesamt besteht das Problem aus n_F Problemphasen.

Flieger i ist genau dann am Final Approach Fix und bereit zur Landung, wenn er die Nebenbedingungen

$$\begin{array}{rclrcl} x_i(t) & \in & [-10\text{m}, 10\text{m}], & v_i(t) & \in & [280\text{km h}^{-1}, 320\text{km h}^{-1}], \\ y_i(t) & \in & [-10\text{m}, 10\text{m}], & \gamma_i(t) & \in & [\frac{2}{180}\pi, \frac{4}{180}\pi], \\ h_i(t) & \in & [440\text{m}, 460\text{m}], & \chi_i(t) & \in & [-\frac{1}{180}\pi, \frac{1}{180}\pi], \end{array}$$

erfüllt. Ansonsten sind nur die Box-Bedingungen (5.8) bis (5.13) auf Seiten 87–88 gefordert. Unter

Verwendung der binären Variablen $w_{E,i}(t)$ ergeben sich die geschalteten Bedingungen

$$
\begin{aligned}
-10 \;\leq\; & w_{E,i}(t_j)x_i(t_j) && \leq\; 10, && (5.22)\\
-10 \;\leq\; & w_{E,i}(t_j)y_i(t_j) && \leq\; 10, && (5.23)\\
-10 \;\leq\; & w_{E,i}(t_j)(h_i(t_j)-450) && \leq\; 10, && (5.24)\\
-20 \;\leq\; & w_{E,i}(t_j)(v_i(t_j)-300) && \leq\; 20, && (5.25)\\
-\frac{1}{180}\pi \;\leq\; & w_{E,i}(t_j)(\gamma_i(t_j)+\tfrac{3}{180}\pi) && -\leq\; \frac{1}{180}\pi, && (5.26)\\
-\frac{1}{180}\pi \;\leq\; & w_{E,i}(t_j)\chi_i(t_j) && \leq\; \frac{1}{180}\pi. && (5.27)\\
& && && (5.28)
\end{aligned}
$$

Diese stellen sicher, dass sich Flieger i einreiht.

Sicherheitsabstände

Um die Sicherheit im Luftraum zu gewährleisten müssen die Flugzeuge einen gewissen Mindestabstand voneinander halten. Dieser Mindestabstand ist auch der beschränkende Faktor bei der Auslastung von Flughäfen, weswegen er so gering wie aus Sicherheitsaspekten vertretbar gehalten werden sollte. Je nach ihrer Größe verursachen die Flugzeuge unterschiedlich starke Luftverwirbelungen und reagieren unterschiedlich empfindlich auf diese. Fliegt ein kleineres Flugzeug hinter einem größeren, so muss es einen größeren Abstand halten als umgekehrt. Die von der Größe abhängigen Abstände und die Einteilung in drei Klassen (leicht, mittel und schwer) werden wie in (FFG+03) gewählt. Die Abstände für die verschiedenen Kombinationen sind:

Diese Abstände sollen während der gesamten Betrachtung eingehalten werden. Zudem soll über, neben jedem Flugzeug, sowie unterhalb jeden Flugzeugs ein Sicherheitsabstand von $600\,\mathrm{m}$ nicht unterschritten werden. Diese Angaben definieren die Sicherheitszone um die Flugzeuge herum nicht exakt, da keine Form vorgegeben ist. Es besteht die Möglichkeit Zylinder, Quader oder ellipsoide Formen zu verwenden. Hier wird eine ellipsoide Form gewählt.

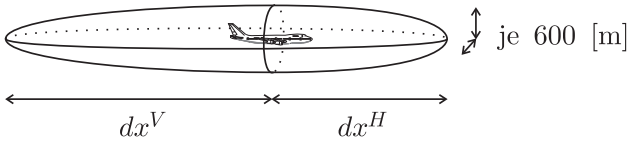


Abbildung 5.4: Sicherheitszone eins Flugzeugs zur Kollisionsvermeidung

Eine Schwierigkeit stellt die mathematische Formulierung der Sicherheitszone für die Kollisionsvermeidung zwischen je zwei Flugzeugen als Nebenbedingung dar, da, wie in Abbildung 5.4 zu

Tabelle 5.3: Abstand zwischen Flugzeugen unterschiedlicher Gewichtsklassen

vorderes Flugzeug	nachfolgendes Flugzeug leicht	mittel	schwer
leicht	5.55km	5.55km	5.55km
mittel	9.25km	7.40km	7.40km
schwer	11.10km	9.25km	7.40km

sehen ist, die Sicherheitszone vor und hinter dem Flugzeug aus unterschiedlichen Ellipsoiden besteht.

Betrachtet werden die Flugzeuge i und j. In jedem Flugzeug sei ein Koordinatensystem S_i bzw. S_j im jeweiligen Schwerpunkt befestigt, so dass die x-Achsen in Flugrichtung und die y-Achse jeweils in Richtung des linken Flügels zeigen. Die Verdrehung von Koordinatensystem S_i bezüglich dem Inertialsystem I wird durch die Rotation

$$ {}^{S_i}\boldsymbol{R}_I = \begin{pmatrix} \cos\chi_i & -\sin\chi_i & 0 \\ \sin\chi_i & \cos\chi_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\gamma_i & 0 & \sin\gamma_i \\ 0 & 1 & 0 \\ -\sin\gamma_i & 0 & \cos\gamma_i \end{pmatrix} $$

beschrieben. Anhand der relativen Position ${}^{S_i}\boldsymbol{d} = (dx_{ij}, dy_{ij}, dh_{ij})$ von Flugzeug j bezüglich Flugzeug i dargestellt im Koordinatensystem S_i lassen sich die Nebenbedingungen

$$ h_{ij}^V(\boldsymbol{x}(t)) := \left(\frac{dx_{ij}}{d_{ij}^V}\right)^2 + \left(\frac{dy_{ij}}{600\,\mathrm{m}}\right)^2 + \left(\frac{dh_{ij}}{600\,\mathrm{m}}\right)^2 - 1 + w_{L,i} + w_{L,j} \geq 0, \quad \leftarrow \quad dx_{ij} < 0, $$

d. h. falls j hinter i fliegt, und

$$ h_{ij}^H(\boldsymbol{x}(t)) := \left(\frac{dx_{ij}}{d_{ij}^H}\right)^2 + \left(\frac{dy_{ij}}{600\,\mathrm{m}}\right)^2 + \left(\frac{dh_{ij}}{600\,\mathrm{m}}\right)^2 - 1 + w_{L,i} + w_{L,j} \geq 0, \quad \leftarrow \quad dx_{ij} < 0, $$

d. h. falls j vor i fliegt, aufstellen. Diese stellen sicher, dass j nicht in die Sicherheitszone von i eindringt. Die Bedingung ist immer erfüllt, wenn sich eines der Fugzeuge im Landeanflug befindet, da dann $w_{L,i} = 1$ und/oder $w_{L,j} = 1$ gilt. Damit hat die Bedingung keinen Einfluss auf den neutralen Zustand.

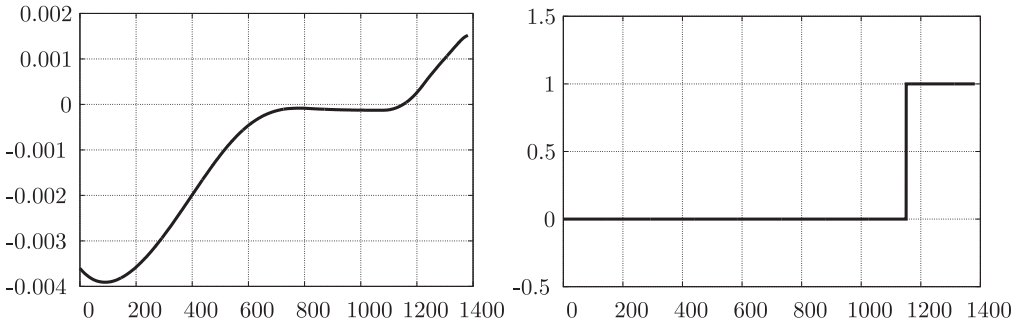
Durch Einführen der Steuerungen $u_{ij}^V(t) \in [0,1]$ und $u_{ij}^H(t) \in [0,1]$ kann die Sicherheitszone in Abhängigkeit der relativen Reihenfolge der Flugzeuge beschrieben werden:

$$ \begin{aligned} 0 &\leq u_{ij}^V(t) h_{ij}^V(\boldsymbol{x}(t)) + u_{ij}^H(t) h_{ij}^H(\boldsymbol{x}(t)), && (5.29) \\ 0 &\geq u_{ij}^V(t) dx_{ij}, && (5.30) \\ 0 &\leq u_{ij}^H(t) dx_{ij}, && (5.31) \\ 1 &= u_{ij}^V(t) + u_{ij}^H(t). && (5.32) \end{aligned} $$

Ist Flugzeug j hinter Flugzeug i, so ist $dx_{ij} < 0$, weswegen nach Ungleichung (5.30) $u_{ij}^V = 0$ gilt, woraus mit (5.32) unmittelbar folgt, dass $h_{ij}^H(\boldsymbol{x}(t)) \geq 0$ erfüllt sein muss. Im umgekehrten Fall gilt entsprechendes. Im Falle $dx_{ij} = 0$ ist (5.29) konstant bezüglich u_{ij}^V, weswegen dieser Fall keine Probleme bereitet. Abbildung 5.5 auf der nächsten Seite zeigt den Abstand zwischen zwei Flugzeugen mit dazugehöriger Schaltfunktion.

Eine weitere Möglichkeit ist die Nebenbedingung stückweise durch

$$ h_{ij}(\boldsymbol{x}(t)) = \begin{cases} h_{ij}^V(\boldsymbol{x}(t)) & \text{falls } 0 \geq dx_{ij} \\ h_{ij}^H(\boldsymbol{x}(t)) & \text{sonst} \end{cases} $$



(a) Relativer Abstand in Flugrichtung (b) Wert der Schaltfunktion

Abbildung 5.5: Schaltung hervorgerufen durch relative Lage der Flugzeuge

zu formulieren. Dies ist hier möglich, da $h_{ij}(\boldsymbol{x}(t))$ an der Stelle $dx_{ij} = 0$ stetig und auch stetig differenzierbar ist. Diese Formulierung benötigt weniger Problemvariablen und Nebenbedingungen. Testbeispiele zeigten, dass sie für das vorliegende Beispiel geeignet ist.

Kosten

Bei der Formulierung der Problemkosten können die unterschiedlichen Wünsche der einzelnen Inhaber der Fluglinien, sowie die der Betreiber des Flughafens berücksichtigt werden.

In dieser Arbeit wird davon ausgegangen, dass die Inhaber der Fluglinien vermutlich an möglichst geringem Treibstoffverbrauch und hohem Komfort für die Passagiere interessiert sind. Die Kosten für den Treibstoff können mit der Länge der Flugstrecke in Verbindung gebracht werden. Die Minimierung der Flugstrecke entspricht der Minimierung des Terms

$$C_{BT,i} = \int_{t_0}^{t_f} \dot{x}_i^2 + \dot{y}_i^2 + 225\dot{h}_i^2 \mathrm{d}\, t.$$

Der Komfort der Passagiere wird durch die Kräfte, die auf das Flugzeug wirken, beeinflusst. Die Kräfte ergeben sich aus der linearen Beschleunigung, den Kreisel- und Fliehkräften beim Kurvenflug. Damit erhöht die Minimierung der linearen Geschwindigkeits- und Winkeländerungen, die durch

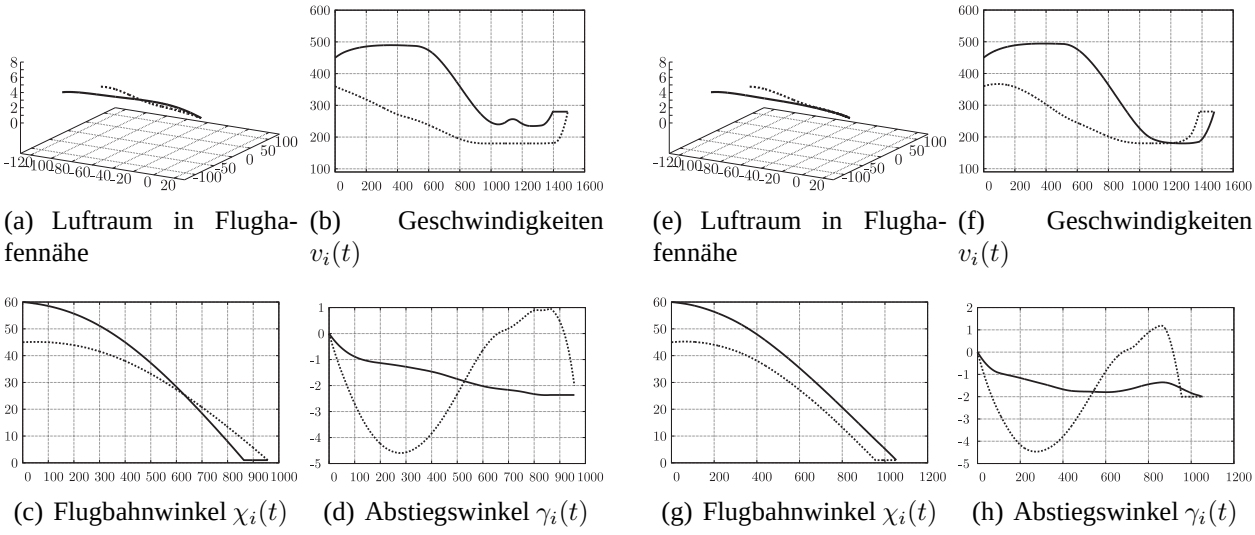
$$C_{KP,i} = \int_{t_0}^{t_f} \dot{v}_i^2 + 25\dot{\gamma}_i^2 + 25\dot{\chi}_i^2 \mathrm{d}\, t$$

gegeben sind, den Komfort der Passagiere.

Weiter wird eine optimale Auslastung der Landebahnen für die Betreiber eines Flughafens als eine wünschenswerte Option gesehen. Diese kann durch Minimierung der Gesamtzeit aller Landevorgänge erreicht werden.

Da die Wünsche der Flughafenbetreiber und Flugzeuginhaber zueinander konträr sind, wird als Zielfunktional eine gewichtete Summe der Einzelkosten

$$\frac{1}{10} t_f + \sum_{i=1}^{n_F} (1 - w_{L,i}(t))(C_{BT,i} + C_{KP,i})$$



(a) Luftraum in Flughafennähe (b) Geschwindigkeiten $v_i(t)$ (e) Luftraum in Flughafennähe (f) Geschwindigkeiten $v_i(t)$

(c) Flugbahnwinkel $\chi_i(t)$ (d) Abstiegswinkel $\gamma_i(t)$ (g) Flugbahnwinkel $\chi_i(t)$ (h) Abstiegswinkel $\gamma_i(t)$

Abbildung 5.6: Mögliche Landereihenfolgen bei zwei Flugzeugen: Flieger 1 (——), Flieger 2 (·····)

verwendet. Ein Flugzeug verursacht nur solange Kosten, wie $w_{L,i} = 0$ gilt, das Flugzeug also noch nicht gelandet ist.

5.1.2 Ergebnisse

In diesem Abschnitt sind die Ergebnisse verschiedener Beispielrechnungen zusammengefasst. Die Anfangszustände der jeweiligen Flugzeuge können Tabelle 5.4 entnommen werden. Als Startwert für die Optimierung wurden Trajektorien verwendet, die durch lineare Interpolation zwischen den Anfangszuständen und dem Endzustand, der durch den Final Approach Fix gegeben ist, gewonnen wurden. Dieser Startwert stellt einen für die Optimierung unzulässigen Punkt dar, da weder die Dynamik noch die geforderten Sicherheitsabstände berücksichtigt wurden. Dennoch ist er ausreichend, um Konvergenz zu erhalten.

Alle Berechnungen wurden auf einem PC mit $2\,\mathrm{GHz}$ Prozessortakt durchgeführt.

Tabelle 5.4: Anfangszustände

Flugzeug Nr.	Typ	x_0	y_0	h_0	v_0	γ_0	χ_0
1	A380	-100 km	-100 km	4 km	450 $\mathrm{km\,h^{-1}}$	0°	60°
2	B737	-80 km	-50 km	4 km	360 $\mathrm{km\,h^{-1}}$	0°	45°
3	A330	-40 km	120 km	5 km	540 $\mathrm{km\,h^{-1}}$	0°	-95°
4	B747	-100 km	70 km	6 km	600 $\mathrm{km\,h^{-1}}$	0°	-45°
5	A310	-80 km	100 km	5 km	600 $\mathrm{km\,h^{-1}}$	0°	frei

Bei nur zwei Fliegern existieren nur zwei Lösungskandidaten, diese sind in 5.6 dargestellt. Zur Berechnung beider Probleme wurden zusammen $14.70\,\mathrm{s}$ benötigt. Es ist deutlich zu sehen, dass der Flieger, der als erstes gelandet ist, in den neutalen Zustand wechselt. Seine Zustandsvariablen bleiben konstant.

Tabelle 5.5: Optimale Landereihenfolge von drei Flugzeugen; ohne Homotopietechniken

Reihenfolge	(1,2,3)	(1,3,2)	(2,1,3)	(2,3,1)	(3,1,2)	(3,2,1)	Branch-and-Bound
Rechenzeit	49.30	43.18	46.08	49.20	41.54	45.70	849.67
Kosten	486.54	497.43	**480.37**	482.51	503.85	493.51	**480.37**

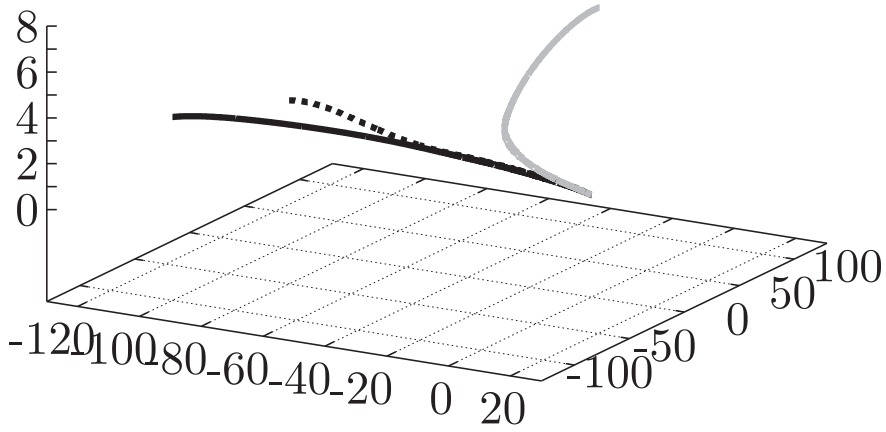
Bei drei Fliegern beläuft sich die Anzahl der Lösungskandidaten auf sechs Reihenfolgen. Für die Berechnung der optimalen Reihenfolge musste nur die Wurzel des Baums untersucht werden. Die binären Variablen ergaben sich automatisch zu null bzw. eins. Zur Verifikation der gefundenen Lösung wurden alle Lösungskandidaten berechnet. Diese sind in Tabelle 5.5 aufgelistet, die optimale Lösung ist duch Fettdruck hervorgehoben. Die zugehörigen Plots der Zustände und Steuerungen auf einem Gitter mit 40 Stützpunkten, können in 5.7 gefunden werden. Flugzeug eins und zwei landen in der gleichen Reihenfolge, wie im vorherigen Beispiel. Der dritte Flieger reiht sich als letztes ein.

Bei den Beispielen mit vier Fliegern treten bereits numerische Probleme auf. Während der Suche im Branch-and-Bound-Baum liefert das Kollokationsverfahren oftmals keine Lösung für die relaxierten Probleme. Die Lösung des Elternknotens als Startwert ist hier bei manchen Knoten eine zu schlechte Startschätzung für zuverlässige Konvergenz. Dies führt dazu, dass die Rechenzeiten enorm ansteigen. Ein Versuch mit dieser Wahl wurde nach 8h abgebrochen.

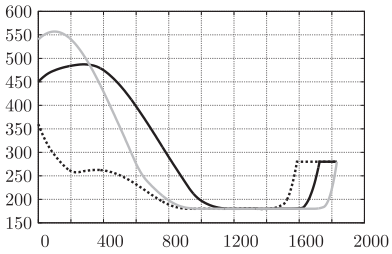
Die Verwendung eines Homotopieverfahrens wärend der Baumsuche reduzierte die Rechenzeit auf $1044.30\,\mathrm{s}$. Die Kosten für diese Reihenfolge (2,4,1,3) belaufen sich auf 541.46. Die Ergebnisplots der dadurch gefundenen Lösung enthält 5.8. Eine Erweiterung auf fünf Flieger konnte mit den gleichen Einstellungen innerhalb $307\,\mathrm{min}\ 43.77\,\mathrm{s}$ gelöst werden.

Die Suchbäume zu den Problemen mit drei, vier und fünf Fliegern sind in Abbildung 5.10 dargestellt. Bei allen Problemen wird eine Tiefensuche mit Pseudokosten und benutzerspezifische Prioritäten verwendet. Die Prioritäten werden so gesetzt, dass die binären Schaltungen, die ein Einreihen hervorrufen entsprechend der Reihenfolge der Phasen bevorzugt behandelt werden. Die Dreiecke in den Bäumen beschreiben Knoten, die aufgrund ihrer zu großen unteren Schranke entfernt wurden. Kreise stellen gewöhnliche Knoten dar, die unterschiedlichen Grautöne geben an, ob eine optimale Lösung gefunden wurde oder ein unzulässiges Problem vorliegt. Bei den vorliegenden Bäumen sind nur Knoten als unzulässig gekennzeichnet, deren lineare Bedingungen an die binären Variablen verletzt waren. Der Diamant bezeichnet jeweils die beste gefundene Lösung.

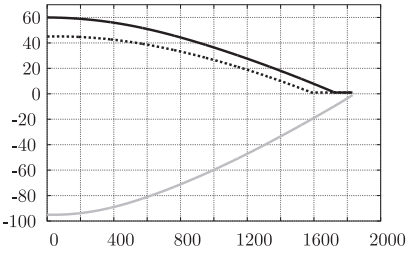
Bei drei Fliegern konnte die optimale Reihenfolge direkt in der Wurzel des Baums gefunden werden. Bei allen Problemen liegen lokale Minima in der Nähe jeder möglichen Reihenfolge. Bei fünf Fliegern liefert eine Starttrajektorie, die sich durch lineare Interpolation ergibt, in der Wurzel ein lokales Minimum, das einen größeren Zielfunktionswert hat, als manche der Endknoten. Zunächst wird der linke Ast von der Wurzel aus durchsucht, und liefert eine Landereihenfolge, deren Kosten niedriger als die der Wurzel sind. Das führt dazu, dass der rechte Ast von der weiteren Suche ausgeschlossen wird. Das Verfahren endet nach $30\,\mathrm{min}$. Die Suche kann nun mit dieser Lösung als Starttrajektorie erneut gestartet werten. Diese Vorgehensweise liefert den Suchbaum aus Abbildung 5.10 und die in Abbildung 5.9 dargestellten Lösungen.



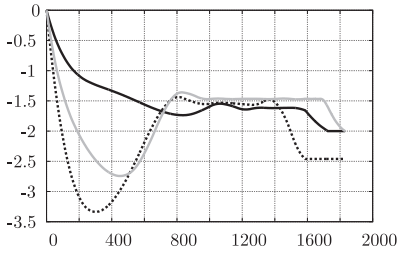
(a) Luftraum in Flughafennähe



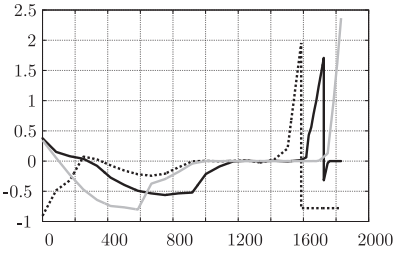
(b) Geschwindigkeiten $v_i(t)$



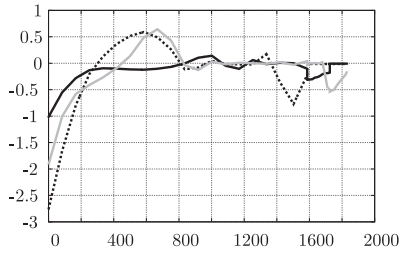
(e) Flugbahnwinkel $\chi_i(t)$



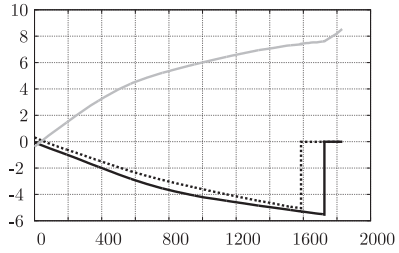
(h) Abstiegswinkel $\gamma_i(t)$



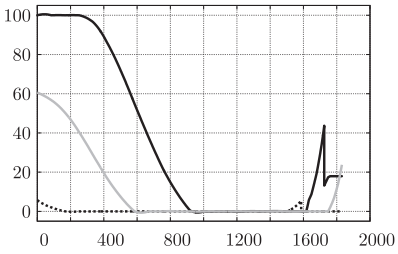
(c) lineare Beschleunigung $u_{v,i}(t)$



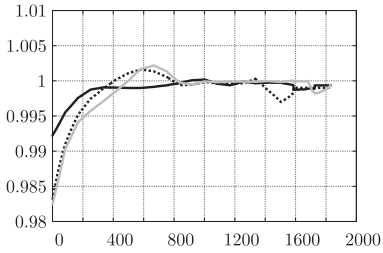
(f) Flugbahnwinkelgeschwindigkeit $u_{\chi,i}$



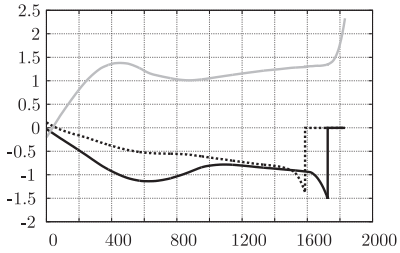
(i) Abstiegswinkelgeschwindigkeit $u_{\gamma,i}(t)$



(d) Triebwerksschub $T_i(t)$

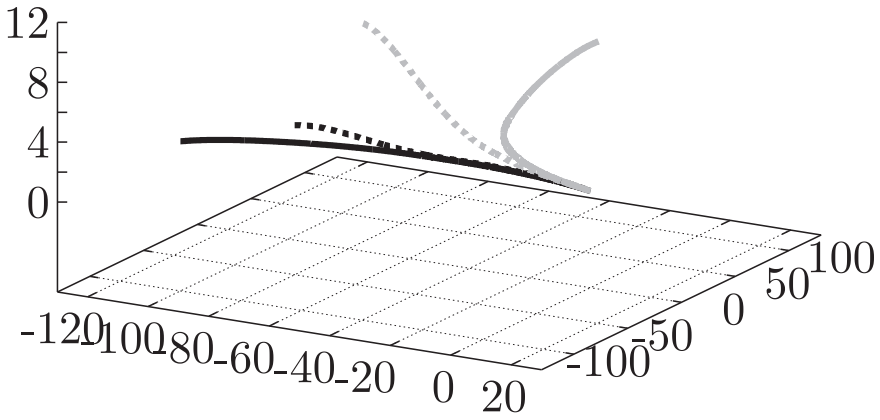


(g) Auftrieb $L_i(t)/m_i$

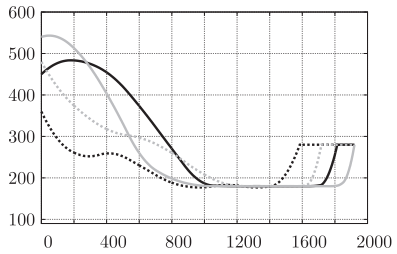


(j) Schräglage $\phi_i(t)$

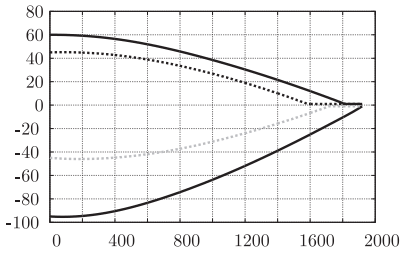
Abbildung 5.7: Optimale Landereihenfolge von drei Flugzeugen: Flieger 1 (——), Flieger 2 (·····), Flieger 3 (——)



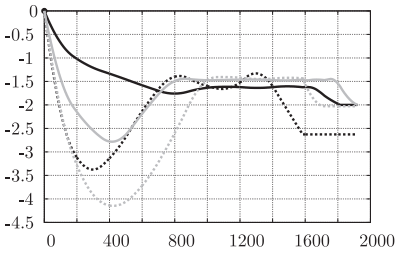
(a) Luftraum in Flughafennähe



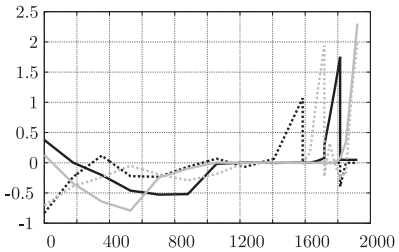
(b) Geschwindigkeiten $v_i(t)$



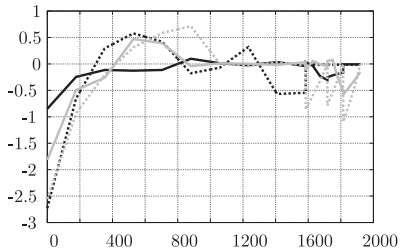
(e) Flugbahnwinkel $\chi_i(t)$



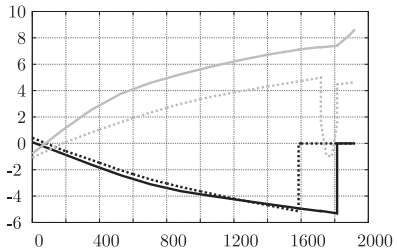
(h) Abstiegswinkel $\gamma_i(t)$



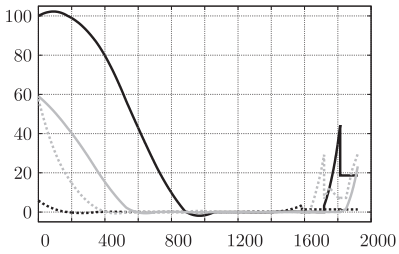
(c) lineare Beschleunigung $u_{v,i}(t)$



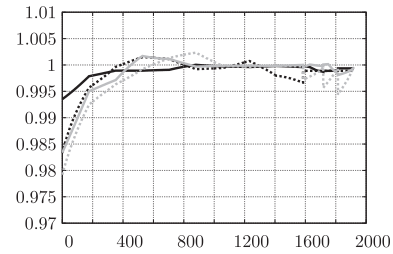
(f) Flugbahnwinkelgeschwindigkeit $u_{\chi,i}$



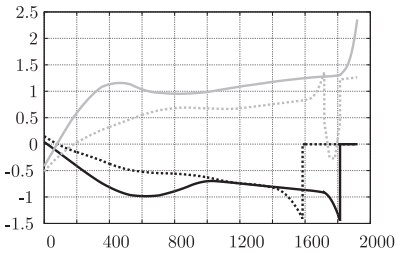
(i) Abstiegswinkelgeschwindigkeit $u_{\gamma,i}(t)$



(d) Triebwerksschub $T_i(t)$

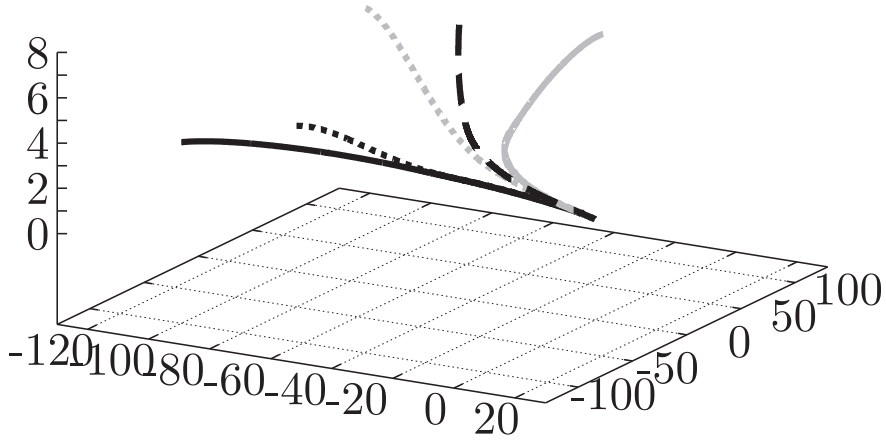


(g) Auftrieb $L_i(t)/m_i$

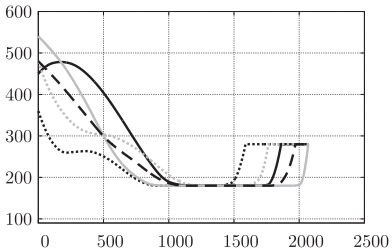


(j) Schräglage $\phi_i(t)$

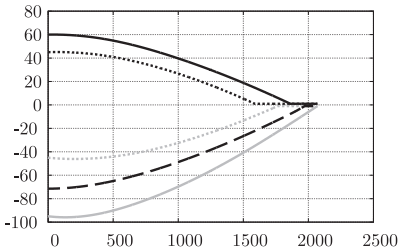
Abbildung 5.8: Optimale Reihung bei vier Flugzeugen: Flieger 1 (——), Flieger 2 (·····), Flieger 3 (——), Flieger 4 (·····)



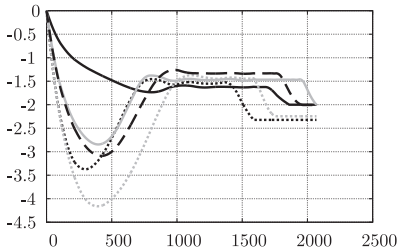
(a) Luftraum in Flughafennähe



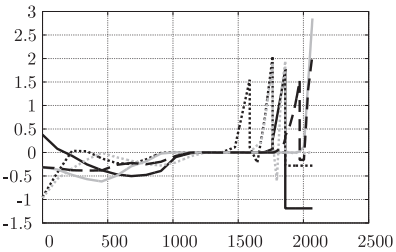
(b) Geschwindigkeiten $v_i(t)$



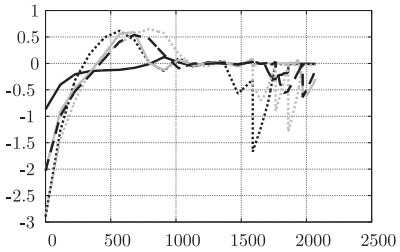
(e) Flugbahnwinkel $\chi_i(t)$



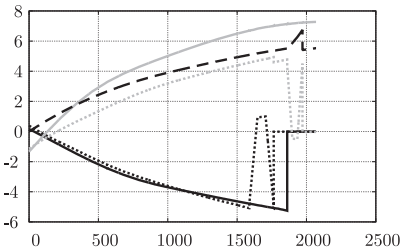
(h) Abstiegswinkel $\gamma_i(t)$



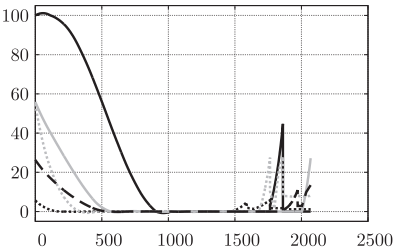
(c) lineare Beschleunigung $u_{v,i}(t)$



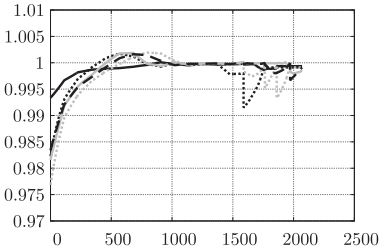
(f) Flugbahnwinkelgeschwindigkeit $u_{\chi,i}$



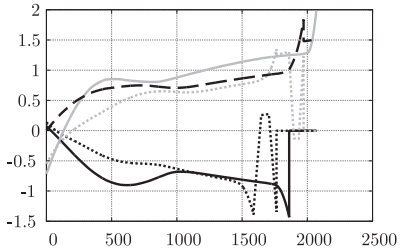
(i) Abstiegswinkelgeschwindigkeit $u_{\gamma,i}(t)$



(d) Triebwerksschub $T_i(t)$



(g) Auftrieb $L_i(t)/m_i$



(j) Schräglage $\phi_i(t)$

Abbildung 5.9: Optimale Reihung bei fünf Flugzeugen: Flieger 1 (——), Flieger 2 (·····), Flieger 3 (——), Flieger 4 (·····), Flieger 5 (– –)

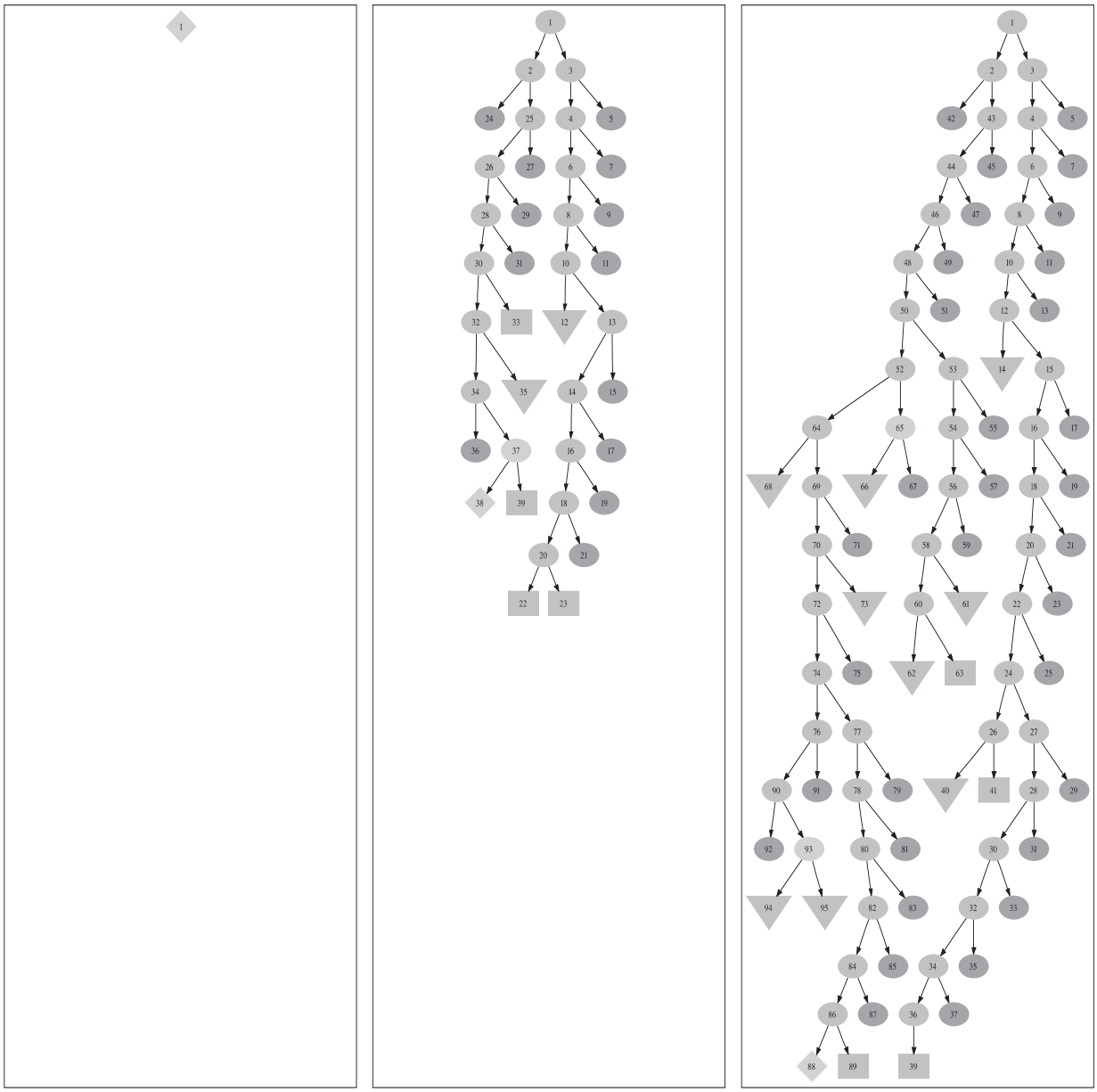


Abbildung 5.10: Branch-and-Bound-Bäume für die Landereihenfolge von drei, vier und fünf Fliegern. z[i] bezeichnet die binären Variablem im verwendeten Programm.

5.2 Aufgabenverteilung und Strategieplanung

Immer weitere Einsatzbereiche werden für die Robotik erschlossen. Der Einsatz von Robotern in der industriellen Fertigung ist mittlerweile ein gewohnter Anblick, aber auch in Bereichen der Pflege, Objektüberwachung, bei Rettungseinsätzen oder auch in der Unterhaltungsindusrie sind autonome und teilautonome Robotiksysteme auf dem Vormarsch. Je komplexer die Einsatzgebiete für Roboter werden, um so mehr ist eine Zusammenarbeit verschiedener Systeme gefragt. Diese muss geplant und koordiniert werden.

Im Bereich der Koordination von Robotiksystemen finden sich verschiedene Ansätze die rein verhaltensbasiert (Mat94) oder marktbasiert (DS01) sind, oder auch andere Architekturen (SSD+02; Par98; Par02) realisieren.

In (Mat94) wird ein verhaltensbasierter Ansatz verfolgt. Als Basis dienen Grundverhalten, die mechanische Eigenschaften, Sensoreigenschaften und die Eigenschaften der Umgebung, wie z. B. Interaktion und Sensorinformation kombinieren und geeignet abstrahieren. Diese werden benutzt um verschiedene robuste Gruppenverhalten zu generieren, die sicheres Herumfahren, Verfolgen, Sammeln, Verteilen und Zielsuchen beinhalten. Auf diesem Grundstock aufbauend werden die Roboter durch *Reinforcement Lernen* in die Lage versetzt, komplexe Aufgaben wie z. B. Nahrungssuche zu erlernen. Das gesamte Konzept wird mit einer Gruppe homogener Robotersysteme validiert.

Ein weiterer verhaltensbasierter Ansatz ist in der *Alliance* Architektur (Par98) realisiert. Sie stellt eine Softwarearchitektur für heterogene Multi-Roboterkooperation dar. Die einzelnen Roboter werden dabei mit Beweggründen, wie Ungeduld oder Einverständnis ausgestattet. Diese Beweggründe sind so gestaltet, dass ein Roboter jederzeit eine Aufgabe beenden kann, falls sie nicht mehr die Möglichkeit zeigt, den gewünschten Effekt zu erzielen. Je nachdem steigen und fallen die Beweggründe, womit in dieser Architektur eine adaptive Aufgabenwahl erreicht wird, die fehlertolerant gegenüber Ausfällen bei den Teammitgliedern, Änderungen der Teamaufgabe, Wechsel der Teamzusammenstellung oder auch Ausfällen und Störungen bei der Kommunikation ist. Eine explizite Koordination der Roboter ist hier nicht vorgesehen. Basierend auf dieser Philosophie wird z. B. das Problem *gemeinsamer Beobachtung mehrerer sich bewegender Ziele* gelöst (Par02).

Zu den marktbasierten Ansätzen gehört die Architektur *Murdoch*. Diese stellt einen komplett verteilten Mechanismus zur Rollen- bzw. Aufgabenzuordnung dar, der auf einer Publish/Subscribe-Architektur basiert, die Auktionen verwendet, um die einzelnen Aufgaben zuzuweisen.

Dazu versenden einzelne Roboter – im Gegensatz zu direkter Kommunikation – Nachrichten mit Informationen über ihre Ressourcen und ihren Bedarf an alle anderen, womit eine gewisse Fehlertoleranz in der Kommunikation erreicht wird. Auf der Basis dieses kommunizierten Wissens über die anderen Roboter wird nun entschieden, ob eine Aufgabe, die hierarchisch als Baum von Teilaufgaben strukturiert ist, übernommen wird oder nicht. Das Ziel dieses Ansatzes ist nicht, die Ressourcen in einer global optimalen Weise zu verteilen, sondern vielmehr eine effiziente Methode zur Verteilung von Aufgaben innerhalb von Gruppen autonomer, heterogener Roboter zu realisieren. In einem verwandten, marktbasierten Ansatz erfolgt die Rollenverteilung zur Maximierung des individuellen ‚Profits‘ der Roboter (DS01).

Eine Erweiterung der traditionellen Drei-Ebenen-Architektur (Verhaltensebene, Ausführungsebene und Planungsebene) auf Anwendungen mit mehreren Robotern wird in (SSD+02) beschrieben. Die einzelnen Ebenen werden dahingehend erweitert, dass jede Ebene eines Roboters mit den entsprechenden der anderen Roboter direkt kommunizieren kann, um die Möglichkeit einer Koordination der Roboter auf verschiedenen Abstraktionsniveaus untereinander zu ermöglichen.

Auch die dynamische Verteilung von Rollen wird benutzt, um Teams mehrerer Roboter zu koordinieren (GM03; BK01; KJ03; SV99; DFL+04).

Bei dem Aufgabenverteilungsproblem (task assignment problem) in (GM03), wie es aus dem Operations Reasearch bekannt ist, wird versucht m Aufgaben optimal auf n Arbeiter zu verteilen, wobei jeder Arbeiter einen Job sucht und jeder Job einen Arbeiter benötigt. Bei der Roboterkooperation handelt es sich allerdings um einen dynamischen Prozess, der eine sich wiederholende Neuverteilung aufgrund von eventuellen Wechsel der Teamzusammenstellung etc. notwendig macht. Um dies zu berücksichtigen werden die Aufgaben durch Gewichtung priorisiert und das Zuweisungsproblem fortwährend erneut gelöst.

Ein weiteres Testfeld zur Entwicklung und Validierung von Verhaltensstrategien, in dem sich viele Gruppen engagieren, bietet der Roboterfußball (PBBY04).

In (SV99) werden Umgebungen betrachtet, bei denen die Mitglieder eines Roboterteams autonom mit nur geringer Kommunikation handeln, jedoch in regelmäßigen Abständen die Möglichkeit haben, sich beliebig auszutauschen. Dieser Aspekt ist gerade im Fußball interessant, wo während des Spiels nur eine eingeschränkte Kommunikation stattfinden kann und der komplette Austausch der Informationen in der Kabine stattfindet, wo das Team wieder synchronisiert und das Ziel für die nächste Halbzeit festgelegt wird.

Eine kooperative Verhaltenssteuerung des Teams ist in der Regel so aufgebaut, dass die einzelnen Roboter verschiedene Rollen annehmen und zwischen ihnen wechseln können. Eine Sammlung von Rollen gehört dabei zu einer Formation, in der mehrere Roboter agieren. Im Verlauf der Aufgabenbewältigung werden nun situationsspezifisch verschiedene Teampläne ausgeführt.

Neuste Versuche gehen nun dahin, eine Formulierung qualitativer Strategien und Taktiken im Fußball zu schaffen, die unabhängig von den Eigenheiten der RoboCup-Ligen sind (DFL+04).

In Anlehnung an das auf Taktik ausgerichtete Lehrbuch (Luc01) wird eine Fußballstrategie als Tupel formuliert, zusammengesetzt aus einem Satz von Rollenbeschreibungen und einem Satz komplexer Verhaltensmuster. Die Rollen sind auf die Verhaltensmuster abgestimmt und die Verhaltensmuster selbst wiederum auf die Strategie. Das Problem der Erreichbarkeit beim Passspiel wird hier über Voronoi-Diagramme gelöst.

Diese aus dem menschlichen Fußball abgeleitete Formalisierung von taktischem Verhalten ist unabhängig von den Fähigkeiten und Eigenschaften der Spieler und der Liga im RoboCup, womit es ermöglicht werden soll, dass prinzipiell Spieler unterschiedlicher Teams mit dem Wissen über diese Formalismen zusammen spielen können.

Da die verschiedenen Taktiken auf den Fähigkeiten und Eigenschaften menschlicher Spieler basieren, ist hier noch zu untersuchen, ob diese ohne weitere Änderung im Roboterfußball erfolgreich

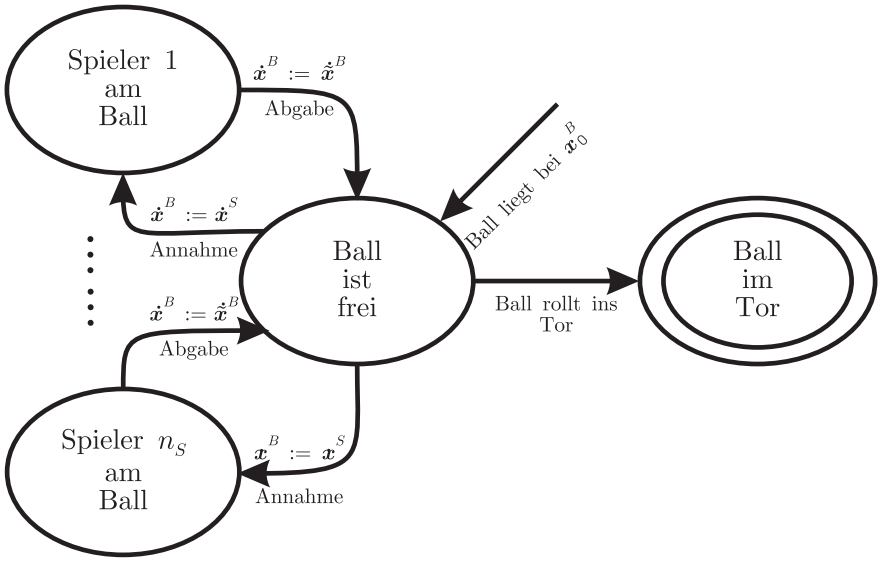


Abbildung 5.11: Hybrider Automat des Balls in einem Fussballspiel

Anwendung finden können. Eine Taktik kann nur dann optimal sein, wenn sie auf die Fähigkeiten der Spieler abgestimmt ist. Deshalb wird in dieser Arbeit, untersucht welches Verhalten sich bei einem Fussballszenario aus der Optimierung unter Berücksichtigung der Dynamik der Roboter ergibt.

5.2.1 Planungsmodell für Teams Fußball spielender Roboter

Betrachtet wird der Angriff eines Fußballroboterteams auf das gegnerische Tor. Ein einzelner Spieler, der im Ballbesitz ist, kann diesen dribbeln, vorlegen, abspielen, oder auf das Tor schießen. Ist ein Spieler nicht im Ballbesitz, kann er sich freilaufen oder, falls der Ball frei ist, zu diesem laufen und ihn annehmen. Demnach kann ein Ball im Besitz eines der Spieler sein oder sich frei bewegen. Abbildung 5.11 zeigt den hybriden Automaten eines Balls, der diese Systemzustände beinhaltet. Als Anfangswert dienen feste Position von Spielern und Ball, wie es z. B. bei Anstoß oder Freistoß realisiert werden kann. Die letzte Phase besteht darin, dass der Ball auf das Tor geschossen wird. Vorab kann keine Aussage darüber getroffen werden wie viele Phasen bis zur nächsten Spielunterbrechung ablaufen können.

Modellierung des Angriffspiels

Die einzelnen Spielphasen werden mit binären Schaltungen identifiziert, die genau dann gleich eins sind, wenn die entsprechende Phase aktiv ist. Insgesamt gibt es die möglichen Phasen

- $w_{f,i}(t) = 1 \leftrightarrow$ Ball ist frei und wird von Spieler i angenommen,
- $w_{i,f}(t) = 1 \leftrightarrow$ Spieler i ist im Ballbesitz und spielt diesen ab,
- $w_{f,T}(t) = 1 \leftrightarrow$ der Ball rollt ins Tor,

die zu verschiedenen Zeitpunkten angenommen werden können, wobei aber nur genau eine aktiv sein kann. Diese wird durch die Bedingung

$$1 = \sum_{i=1}^{n_s} (w_{f,i}(t) + w_{i,f}(t) + w_{f,T}(t))$$

ausgewählt. Weitere Einschränkung ergeben sich aus der zeitlichen Abfolge der Phasen. Der Ball kann z. B. am Anfang einer Phase nur frei sein, wenn er zuvor frei war, oder von einem Spieler freigegeben wurde. So werden Endbedingungen der aktuellen Phase mit den Anfangsbedingungen der nächsten verknüpft

$$\begin{aligned} 0 &= \sum_{i=1}^{n_s} (w_{i,f}(t^-) - w_{f,i}(t^+)) - w_{f,T}(t) \\ 0 &= w_{f,i}(t^-) - w_{i,f}(t^+), \quad i = 1, \ldots n_s. \end{aligned}$$

Erreicht ein Spieler den Ball, nimmt er ihn an, so ist die Ballposition an die des Spielers gebunden. Sowie er den Ball abgibt ist die Position des Balls wieder frei.

$$w_{i,f} = 1 \rightarrow \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_B) = 0.$$

Dieser Sachverhalt wird mit der Big-M Methode dargestellt durch

$$0 \leq (1 - w_{i,f}(t)) M_i^2 - \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_B)^2,$$

wobei dist$(\boldsymbol{x}_i, \boldsymbol{x}_B)$ den euklidischen Abstand zwischen zwei Punkten $\boldsymbol{x}_i$ und $\boldsymbol{x}_B$ darstellt und M_i die maximal mögliche Distanz zwischen Ball und Spieler i ist.

Der neutrale Zustand wird angenommen, wenn ein Tor geschossen wurde und bleibt dann bis zum Ende aktiv.

$$w_{f,T}(t^-) \leq w_{null}(t^+), \quad w_{null}(t^-) \leq w_{null}(t^+).$$

Damit kann das Durchlaufen des hybriden Automaten durch Schalten der binärwertigen Funktionen angegeben werden.

Spielermodellierung

Die Spieler im Robocup haben je nach Liga unterschiedliche dynamische Eigenschaften. In der Small-Size- und Mid-Size-Liga werden häufig omnidirektionale Antriebe eingesetzt, womit sich der Roboter unabhängig von seiner Orientierung in jede Richtung bewegen kann. Ein Punktmassemodell für Spieler i, mit Position $\boldsymbol{x}_i = (x_{i,1}, x_{i,2})^T$ und Steuerung $\boldsymbol{u}_i = (u_{i,1}, u_{i,2})^T$, das diese Eigenschaften widerspiegelt lautet

$$\ddot{\boldsymbol{x}}_i = \boldsymbol{f}_{L,i}(\dot{\boldsymbol{x}}_i(t), \boldsymbol{u}_i(t)) + \boldsymbol{f}_{r,i}(\dot{\boldsymbol{x}}_i(t)).$$

Die rechte Seite beschreibt die auf den Spieler einwirkenden Kräfte aufgrund eigener Leistung $\boldsymbol{f}_{L,i}$ und den Laufwiderstand $\boldsymbol{f}_{r,i}$. Unter der Annahme gleicher Bodenbeschaffenheit und Windstille, sind diese Terme unabhängig von der momentanen Position des Spielers. Eine Ermüdung der Spieler wird im Fall des Roboterfußballs ausgeschlossen, womit keine explizite Abhängigkeit von der Zeit vorliegt.

Sobald der Spieler in Ballbesitz kommt, ändert sich seine Bewegungsfreiheit. Diese Einschränkung wird durch Ersetzen des Terms $\boldsymbol{f}_{L,i}$ mit einem Term $\boldsymbol{f}_{LB,i}$ erreicht, womit die Bewegungsgleichung

$$\ddot{\boldsymbol{x}}_i = \boldsymbol{f}_{LB,i}(\dot{\boldsymbol{x}}_i(t), \boldsymbol{u}_i(t)) + \boldsymbol{f}_{r,i}(\dot{\boldsymbol{x}}_i(t))$$

lautet.

Bei der Annahme von Punktmassen kann für die eigene Leistung direkt die Beschleunigung des Spielers als Steuerung verwendet werden. $\boldsymbol{f}_{L,i}(\dot{\boldsymbol{x}}_i(t), \boldsymbol{u}_i(t)) = \boldsymbol{u}_i(t)$. Durch eine geschwindigkeitsproportionale Kraft $\boldsymbol{f}_{r,i}(\dot{\boldsymbol{x}}_i(t)) = -\rho_i \dot{\boldsymbol{x}}_i(t)$, die entgegengesetzt der Laufrichtung wirkt, werden Reibung und Luftwiderstand mit Widerstandskoeffizienten $\rho_i \in \mathbb{R}^+$ zusammengefasst. Durch $\boldsymbol{f}_{LB,i}(\dot{\boldsymbol{x}}_i(t), \boldsymbol{u}_i(t)) = c_i \boldsymbol{u}_i(t)$ mit einem Koeffizienten $0 < c_i < 1$, was einer geringeren Beschleunigung gleichkommt, wird die eingeschränkte Beweglichkeit eines Spieler während des Ballführens modelliert.

Spieler i ist genau dann im Ballbesitz, wenn $w_{i,f}(t) = 1$ gilt, und verharrt, wenn der neutrale Zustand eintritt. Damit lautet die Bewegungsgleichung für Spieler i

$$\ddot{\boldsymbol{x}}_i = w_{i,f}(t)\boldsymbol{f}_{LB,i}(\dot{\boldsymbol{x}}_i(t), \boldsymbol{u}_i(t)) + (1 - w_{i,f}(t) + w_{null}(t))\boldsymbol{f}_{L,i}(\dot{\boldsymbol{x}}_i(t), \boldsymbol{u}_i(t)) + \boldsymbol{f}_{r,i}(\dot{\boldsymbol{x}}_i(t)).$$

Ballmodellierung

Solange ein Spieler im Ballbesitz ist kann das System bestehend aus Spieler und Ball als Einheit betrachtet werden, so dass der Ball in diesem Fall keine Sonderbehandlung erfährt. Ist der Ball frei, so bewegt er sich im einfachsten Fall geradlinig und wird durch Reibung oder Luftwiderstand mit Koeffizienten $\rho_B \in \mathbb{R}^+$ abgebremst. Diese Bremskraft wird als geschwindigkeitsproportional angenommen,

$$\ddot{\boldsymbol{x}}_B(t) = \boldsymbol{f}_B(\boldsymbol{x}_B(t), \dot{\boldsymbol{x}}_B(t)) = -\rho_B \dot{\boldsymbol{x}}_B(t)$$

mit der Ballposition $\boldsymbol{x}_B = (\boldsymbol{x}_{B,1}, \boldsymbol{x}_{B,2})$.

Der Ball ist frei wenn $w_{f,*}(t) := \sum_{i=1}^{n_s} w_{f,i}(t) + w_{f,T}(t) = 1$ erfüllt ist. Die Bewegungsgleichung des gespielten Balls ergibt sich zu

$$\ddot{\boldsymbol{x}}_B(t) = \sum_{i=1}^{n_s} w_{i,f}(t)\boldsymbol{f}_{LB,i}(\dot{\boldsymbol{x}}_i(t), \boldsymbol{u}_i(t)) + w_{f,*}(t)\boldsymbol{f}_B(\boldsymbol{x}_B(t), \dot{\boldsymbol{x}}_B(t)),$$

was $\ddot{\boldsymbol{x}}_B(t) = 0$ impliziert falls $w_{null}(t) = 1$ gesetzt ist.

Schussmodellierung

Im Spiel wird angenommen, dass Position und Geschwindigkeit der Spieler stetig sind. Bei dem Ball ist nur die Position stetig, die Geschwindigkeit nicht immer. Hier wird angenommen, dass die Ballgeschwindigkeit springt, falls ein Spieler den Ball schießt oder annimmt. Dies ergibt sich aus der Translationsbedingung

$$\dot{\boldsymbol{x}}_B(t_k^+) = \sum_{i=1}^{n_s}(w_{f,i}(t_k^-)\dot{\boldsymbol{x}}_i(t_k^-) + w_{f,i}(t_k^-)p_{i,k}\dot{\boldsymbol{x}}_i(t_k^-)) + \left(1 - \sum_{i=1}^{n_s}(w_{f,i}(t_k^-) + w_{f,i}(t_k^-))\right)\dot{\boldsymbol{x}}_B(t_k^-).$$

Bei der Ballannahme erhält der Ball die Geschwindigkeit des Spielers, bei einem Schuss wird der Ball mit dem Parameter $p_{i,k}$ in Laufrichtung des ballführenden Spielers beschleunigt. In allen anderen Fällen bleibt die Ballgeschwindigkeit stetig.

Gegnermodellierung

Die einfachste Annahme für ein gegnerisches Verhalten ist, dass der Aufenthaltsort für jede Zeit t durch eine Funktion $\boldsymbol{x}_G = \boldsymbol{P}(t)$ gegeben ist oder sich durch eine gegeben Steuerung $\boldsymbol{u}_G(t)$ aus der Bewegungsgleichung $\ddot{\boldsymbol{x}}_G(t) = \boldsymbol{f}_G(\dot{\boldsymbol{x}}_G(t), \boldsymbol{u}_G(t))$ berechnen lässt.

Auch ein bekanntes oder angenommenes gegnerisches Verhalten lässt sich verwenden. Dieses wird als Reaktion auf die aktuellen Zustände aller Spieler und des Balls angenommen

$$\dot{\boldsymbol{x}}_G = \boldsymbol{f}_G(\boldsymbol{x}_1(t), \ldots, \boldsymbol{x}_{n_s}(t), \boldsymbol{x}_B(t), t)$$

Damit nun der Ball nicht von dem Gegner abgenommen wird, kann dem Problem eine zusätzliche Nebenbedingung

$$(\boldsymbol{x}_G - \boldsymbol{x}_B)^T(\boldsymbol{x}_G - \boldsymbol{x}_B) \geq d_{min}^2$$

hinzugefügt werden, die einen Mindestabstand d_{min} zwischen Gegner und Ball verlangt.

Eine konservative Lösung wäre auch, für den Gegner optimales Verhalten anzunehmen. Dazu müsste das Problem als dynamisches Spiel mit gemeinsamen Zielfunktional formuliert werden. Die eigenen Spieler versuchen das Zielfunktional zu minimieren, während die Gegner ihrerseits das Funktional maximieren wollen. Die Menge der so berechneten Lösungen wird paretooptimaler Bereich genannt und all diese Lösungen liefern mindestens den Erwartungswert des Spiels. Aus dieser Menge von Lösungen würde dann diejenige ausgewählt, die den größten Zugewinn verspricht, falls der Gegner von seiner optimalen Strategie abweicht. Die Frage ist allerdings, ob sich unter den paretooptimalen Lösungen eine befindet bei der überhaupt ein Torschuss versucht wird.

5.2.2 Optimierung von Szenarien aus dem Roboterfußball

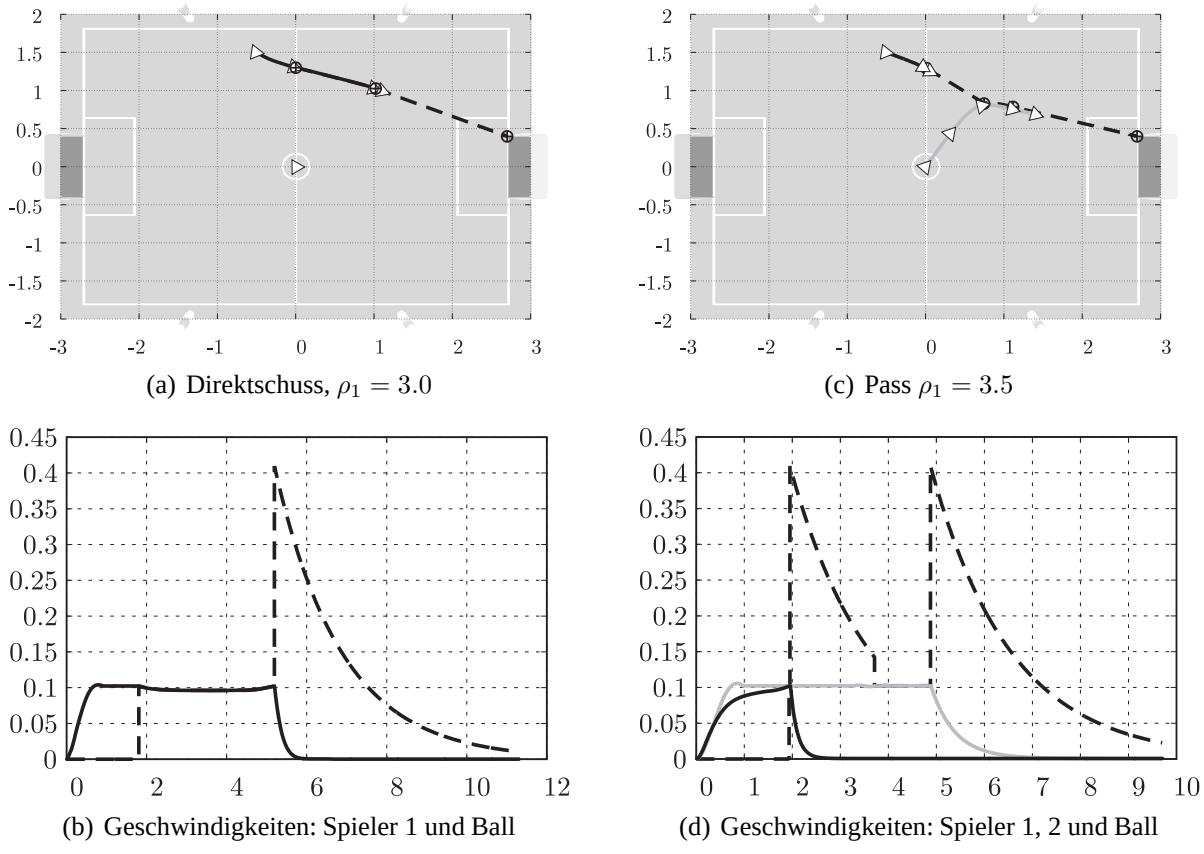
Die folgenden Abschnitte beschreiben die Ergebnisse verschiedener Problemstellungen aus dem Roboterfußball. Wie bereits im letzten Abschnitt angeführt, sind die Spieler durch einfache Punkt-

massenmodelle mit geschwindigkeitsproportionalem Laufwiderstand gegeben. Der nicht geführte Ball wird ebenfalls geschwindigkeitsproportional abgebremst. Bei Schüssen wird die Geschwindigkeit mit dem festen Parameter $p_{i,k} = 3$ multipliziert. Als Zielfunktional wird jeweils die gewichtete Summe von Laufkosten und verstrichener Zeit bis zum Torschuß verwendet.

$$\min c_1 t_f + c_2 \int_{t_0}^{t_f} u_1^T u_1 + u_2^T u_2 \,\mathrm{d}\, t.$$

Das erste Beispiel behandelt zwei Szenarien mit je zwei Spielern. Diese unterscheiden sich in der Dynamik der Spieler und zeigen, dass dieser Unterschied zu verschiedenen Spielstrategien führt. Die beiden weiteren Beispiele erläutern mögliche Varianten zur Modellierung von Gegenspielern.

Abhängigkeit zwischen Strategie und Dynamik



(a) Direktschuss, $\rho_1 = 3.0$

(c) Pass $\rho_1 = 3.5$

(b) Geschwindigkeiten: Spieler 1 und Ball

(d) Geschwindigkeiten: Spieler 1, 2 und Ball

Abbildung 5.12: Spielzüge ohne Gegner in Abhängigkeit von der Dynamik der Spieler; Spieler 1 (——), Spieler 2 (——), Ball (– –)

Eine optimale Spielstrategie hängt immer von den Eigenschaften der einzelnen Spieler ab. Betrachtet wird eine Spielsituation, aus der möglichst schnell ein Torschuß erfolgen soll, wobei keine gegnerischen Spieler berücksichtigt werden. Aus dem eigenen Team werden zwei Spieler betrachtet, von denen keiner im Ballbesitz ist. Jeder der Spieler kann zum Ball laufen, ihn direkt auf das Tor schießen, dribbeln oder abgeben. Mehr als eine mögliche Ballabgabe wird nicht erlaubt, womit

Tabelle 5.6: Problemdimensionen

	Startphase	mittlere Phasen (3x)	letzte Phase	gesamt
Zustandsvariablen	13	13	13	13
Steuervariablen	4	4	4	4
binäre Variablen	7	7	7	35
nichtlin. Nebenbedingungen	4	4	4	4
Übergangsbedingungen	12	12	0	48
Randbedingungen	0	0	1	1
logische Bedingungen	7	10	7	44

Tabelle 5.7: Zusammenfassung der Rechenergebnisse dynamikabhängiger Strategie

	Pass (P4, 2.6GHz)	Direkter Torschuss (XP 1700, 1.4GHz)
Rechenzeit	813s 33s	1091s
untersuchte Knoten	119	129
NLP Variablen	1045	1045
NLP Nebenbedingungen	1117	1117
Berechnete Kosten	25.347	23.909

das Optimalsteuerungsproblem aus fünf Phasen besteht. Die Anzahl der Variablen und Nebenbedingungen für das gemischt diskret-kontinuierliche Optimalsteuerungsproblem sind in Tabelle 5.6 zusammengefasst.

Der Widerstandskoeffizient für das Laufen von Spieler eins beträgt $\rho_1 = 3$, der von Spieler zwei $\rho_2 = 1$.

Abbildung 5.12(a) zeigt den optimalen Spielzug für diese Parameter. Spieler eins läuft zum Ball, führt ihn und schießt dann direkt auf das Tor, während Spieler zwei auf seiner Position verharrt.

Anders gestaltet sich die Situation, wenn der Widerstandskoeffizient von Spieler eins etwas höher, hier $\rho_1 = 3.5$, angesetzt wird. Der von Spieler zwei bleibt unverändert. Abbildung 5.12(c) zeigt das Ergebnis zu diesen Einstellungen. Spieler eins läuft nun zum Ball, führt ihn nur kurz und passt ihn zu Spieler zwei, der nach wenigen Schritten in das Tor schießt.

Beide Beispiele wurden mit Kollokation an Gausspunkten berechnet. Dabei wurden 25 Diskretisierungspunkte verwendet. Die Nebenbedingungen wurden auch in den sich daraus ergebenden Intervallmitten überprüft. Tabelle 5.7 zeigt eine Zusammenfassung der Rechenergebnisse. Die Suchbäume sind in Abbildung 5.13 dargestellt.

Ein Spieler mit Gegner

Realistischer werden die Spielsituationen, wenn Gegner zugelassen werden. Dabei kann das Verhalten des Gegners z. B. aus früheren Spielsituationen geschätzt und als fest vorgegeben integriert werden. Dazu wird eine Spielsituation mit einem angreifenden Spieler und dem verteidigten Torwart betrachtet. Zu Beginn befindet sich der Angreifer auf Höhe der Mittellinie $\boldsymbol{x}_1(t_0) = (0, -0.75)^T$, der Ball liegt ein kleines Stück vor ihm $\boldsymbol{x}_B(t_0) = (0.5, -0.75)^T$ und der Torwart

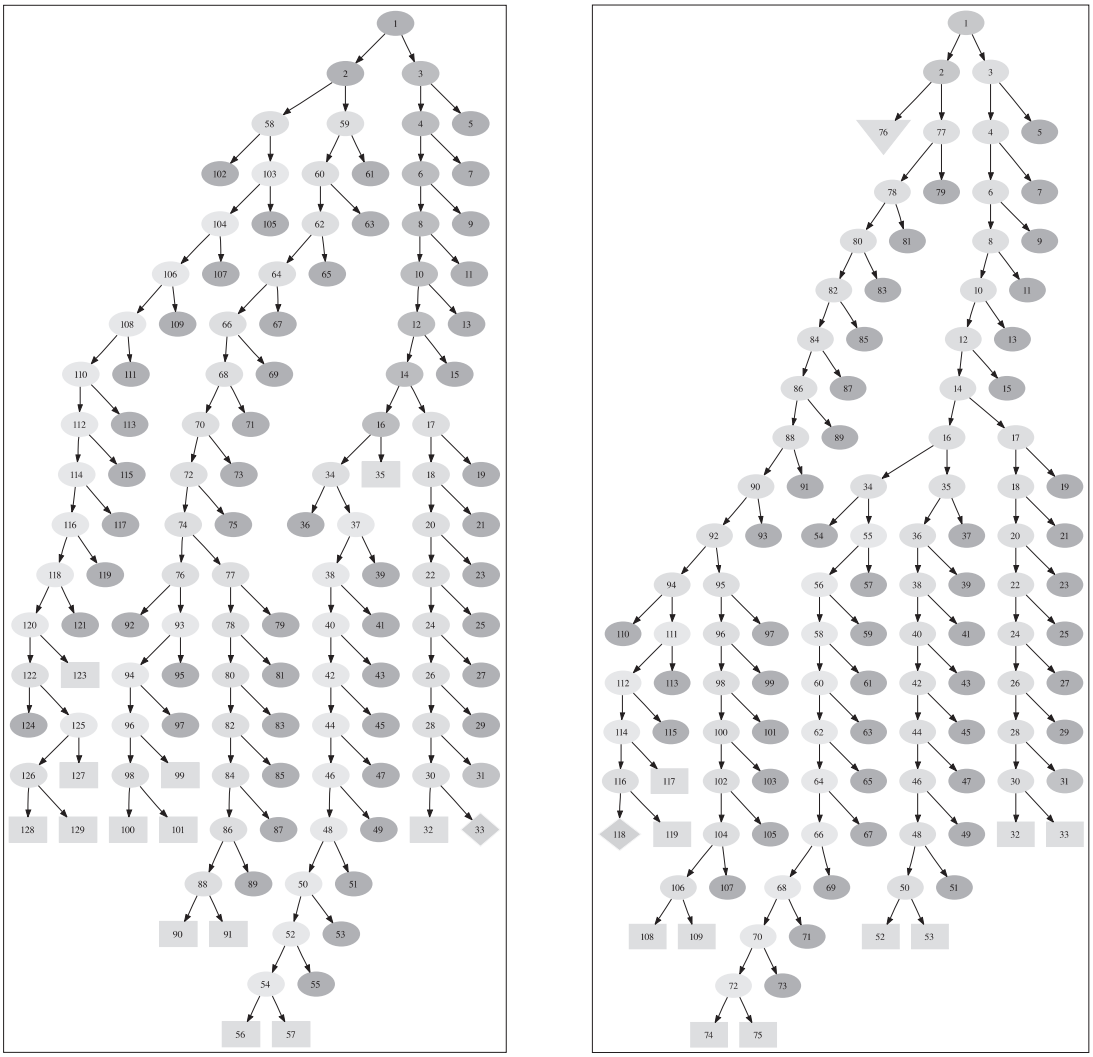


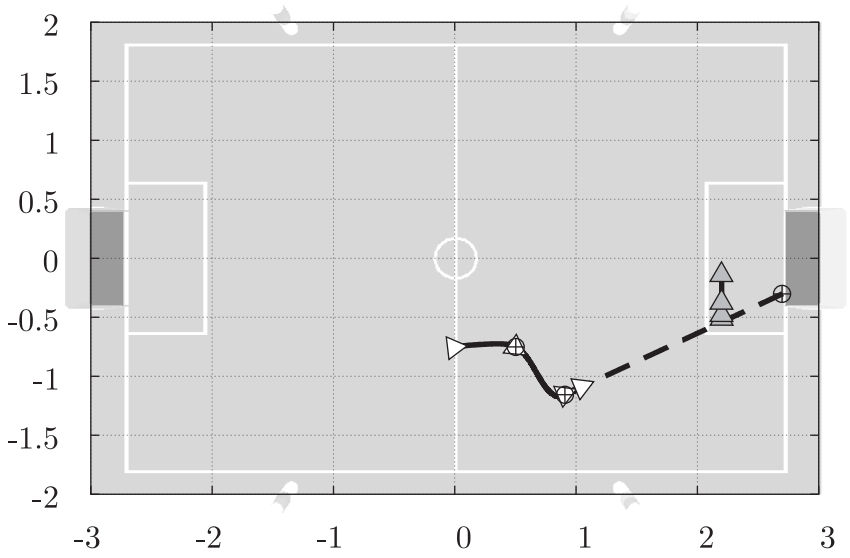
Abbildung 5.13: Branch-and-Bound-Bäume für die Berechnungen dynamikabhängiger Strategie; links Direktschuss $\rho_1 = 3.0$, rechts Pass $\rho_1 = 3.5$

steht in der, von ihm aus gesehen, linken Ecke des Tors $\boldsymbol{x}_G(t_0) = (2.2, -0.5)^T$. Für den Fall, dass sich der Torwart nicht bewegt, ist eine freie Schussbahn in die andere Ecke des Tors gegeben. Es wird angenommen, dass der Torwart vor dem Tor auf- und abläuft. Dies ist durch die Bewegungsgleichung

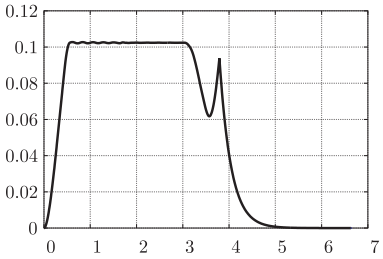
$$\begin{aligned} \dot{x}_{G,1}(t) &= 0.1 \sin(0.2t) \\ x_{G,2}(t) &= 2.2 \end{aligned}$$

berücksichtigt. Weiter wird ein Mindestabstand zwischen Ball und gegnerischem Torwart gefordert. Als Zielfunktion wird eine gewichtete Summe aus Zeit und Bewegungsaufwand verwendet.

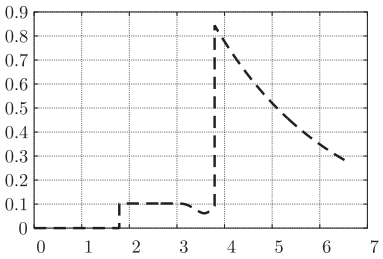
Die optimale Lösung, die in Abbildung 5.14 zu sehen ist, zeigt wie der Angreifer den Ball annimmt, in eine freie Schussbahn hinter dem Torwart bringt und ein Tor erzielt. Die dazu gehörigen Geschwindigkeiten von Angreifer und Ball, sowie der Abstand zwischen Ball und Gegner sind ebenfalls Abbildung 5.14 zu entnehmen.



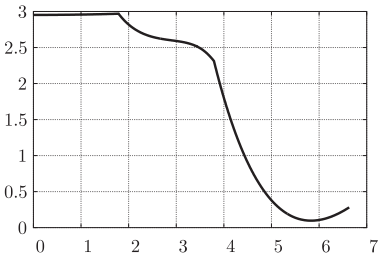
(a) Ein Spieler mit Gegner



(b) Spielergeschwindigkeit



(c) Ballgeschwindigkeit



(d) Abstand zwischen Ball und Gegner

Abbildung 5.14: Angriff eines Stürmers bei fest vorgegebenen Verhalten des Verteidigers; Spieler (——), Ball (– –)

Bei einer Diskretisierung mit 27 Punkten besitzt das Problem 627 Variablen und 736 Nebenbedingungen. Zur Lösung bei vorgegebenen diskreten Variablen wurden ca. $40\,\mathrm{s}$. auf einem PC mit $2\,\mathrm{GHz}$ Prozessortakt benötigt.

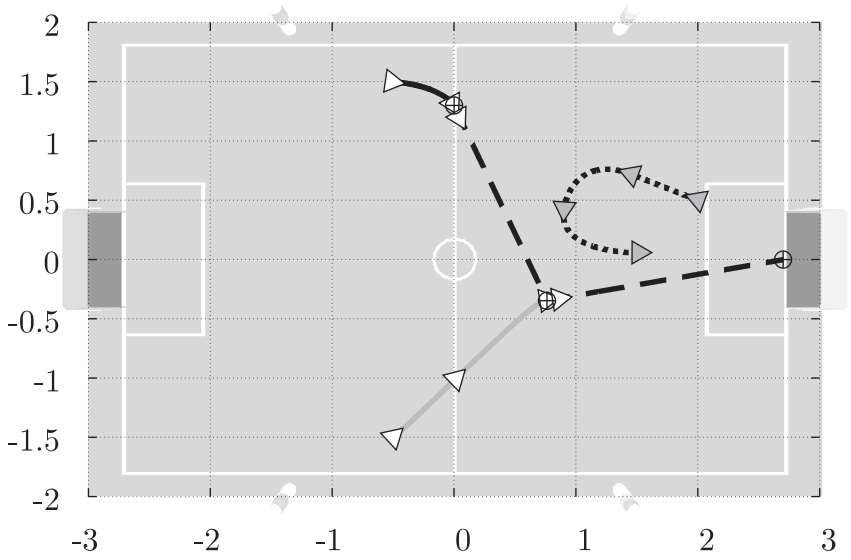
Zwei Spieler mit Gegner

Eine einfache Variante eines gegnerischen Verhaltens ist, dass der gegnerische Spieler dem Ball, der unter dem Winkel α_{BG} gesehen wird, mit der Geschwindigkeit v_G hinterherläuft. Dies wird durch

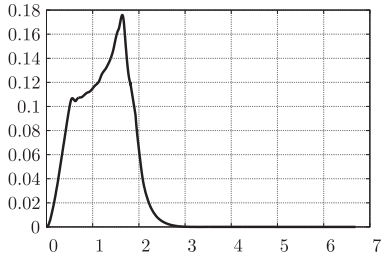
$$\begin{aligned} \dot{x}_{G,1}(t) &= v_G \cos(\alpha_{BG}(t)) = v_G \frac{x_{B,1}(t)-x_{G,1}(t)}{\sqrt{(x_{B,1}(t)-x_{G,1}(t))^2+(x_{B,2}(t)-x_{G,2}(t))^2}}, \\ \dot{x}_{G,2}(t) &= v_G \sin(\alpha_{BG}(t)) = v_G \frac{x_{B,2}(t)-x_{G,2}(t)}{\sqrt{(x_{B,1}(t)-x_{G,1}(t))^2+(x_{B,2}(t)-x_{G,2}(t))^2}} \end{aligned}$$

erreicht.

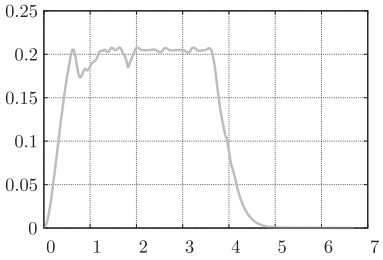
Unter Berücksichtigung dieses gegnerischen Verhaltens wird die folgende Spielsituation untersucht. Der Ball liegt zu Beginn des Spielzugs auf der Mittellinie in der vom Angreifer aus gesehen linken Spielhälfte $\boldsymbol{x}_B(t_0) = (0, 1.3)^T$. Einer der Angreifer ist in nahe dem Ball platziert



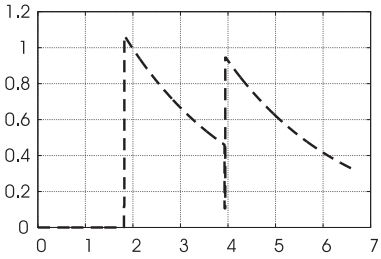
(a) Zwei Spieler mit Gegner



(b) Geschwindigkeit Spieler 1



(c) Geschwindigkeit Spieler 2



(d) Ballgeschwindigkeit

Abbildung 5.15: Angriff zweier Stürmer bei ballabhängigen Verhalten des Verteidigers; Spieler 1 (——), Spieler 2 (——), Ball (– –)

$\boldsymbol{x}_1(t_0) = (-0.5, 1.5)^T$, der andere befindet sich auf der rechten Seite der eigenen Spielhälfte $\boldsymbol{x}_2(t_0) = (-0.5, -1.5)^T$. Der Verteidiger befindet sich am äußeren Rand des Strafraumes $\boldsymbol{x}_G(t_0) = (2, 0.5)^T$ und verhindert einen direkten Torschuss.

Wiederum werden hier die Bewegungskosten und die Zeit bis zum Torschuss minimiert. Die optimale Lösung zeigt einen Pass zwischen den angreifenden Spielern mit abschließendem Schuss auf das Tor. Damit ist es dem Verteidiger nicht möglich den Ball abzufangen. Die Ergebnisse sind in Abbildung 5.15 zusammengefasst.

Die Problemstellung wurde auf einem Gitter mit 75 Punkten gelöst. Damit ergab sich ein nicht lineares Optimierungsproblem mit 1789 Variablen und 1437 Nebenbedingungen. Zur Lösung bei gegebenen diskreten Variablen wurden auf einem PC mit 2GHz Prozessortakt ca. 50s benötigt.

5.3 Rundreiseprobleme

Mit zu den bekanntesten kombinatorischen Optimierungsproblemen gehört das Handlungsreisendenproblem.

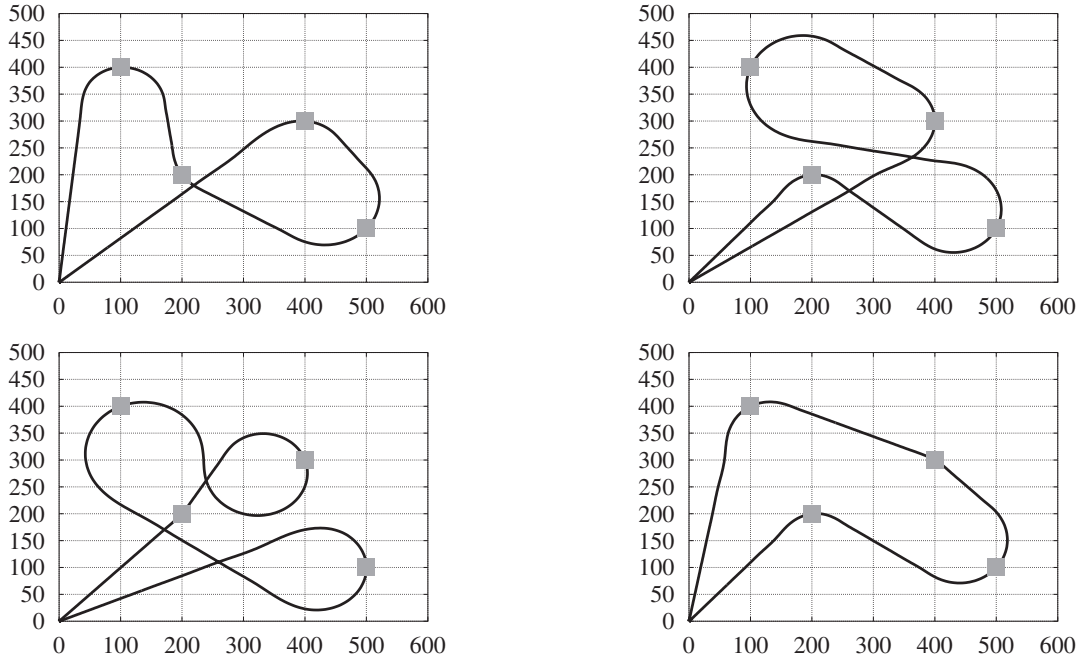


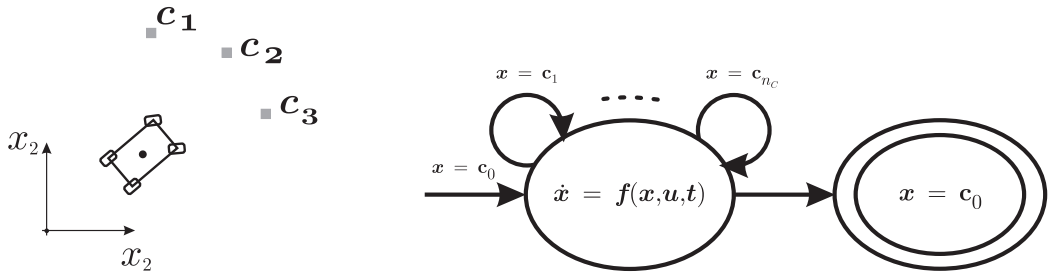
Abbildung 5.16: Vergleich von Rundreisen mit gegebener Reihenfolge der Städte

Ein Handlungsreisender verbringt seine Zeit damit n_c verschiedene Städte zu besuchen, jede davon genau einmal. Am Ende seiner Rundreise kehrt er zum Ausgangspunkt zurück. In welcher Reihenfolge soll er die Städte besuchen, um die Reisezeit zu minimieren?

Laut einem Artikel über die Geschichte der kombinatorischen Optimierung (Sch05) wurde das Problem des Handlungsreisenden bereits in *„Der Handlungsreisende – wie er sein soll und was er zu thun hat,um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein – Von einem alten Commis-Voyageur, B.Fr. Voigt, Ilmenau, 1832“* erwähnt. Es wird darauf hingewiesen, dass das Buch keine mathematische Behandlung des Problems enthält, aber eine der 5 enthaltenen Touren durch 45 Städte Deutschlands kann unter Berücksichtigung der damaligen Gegebenheiten durchaus optimal gewesen sein, auch wenn dies nicht unter Verwendung geodätischer Distanzen zutrifft.

Aus mathematischer Sicht, ist eine Rundreise ein *Hamiltonkreis* (vgl. Definition 18 auf Seite 127) in einem Graphen, der die Städte als Knoten besitzt. Derartige Problemstellungen wurden Mitte des 19. Jahrhunderts hauptsächlich von Sir William Rowan Hamilton und Thomas Penyngton Kirkman untersucht.

Karl Menger studierte in den 20er Jahren des 20. Jahrhunderts eine allgemeine Form des Handlungsreisendenproblems in Wien und Harvard. In den 30er Jahren tauchte die Problemstellung in Princeton wieder auf und wurde in den 40ern von Statistikern untersucht (Mahalanobis (1940), Jessen (1942), Gosh (1948), Marks (1948)). Ein Handlungsreisendenproblem mit 49 Städten lösten George Dantzig, Ray Fulkerson und Selmer Johnson 1954. Seither werden alle paar Jahre größere Beispiele untersucht und beweisbar optimal gelöst. In den 70ern und 80ern des letzten Jahrhunderts lieferten Grötschel, Padberg, Holland und Rinaldi die größten Touren, seit den 90ern sind



(a) Fahrzeug (b) Automat

Abbildung 5.17: Modellierung des motorisierten Handlungsreisenden

dies Applegate, Bixby, Chvátal und Cook (Coo05).

Unter Verwendung der *Lin-Kernighan*-Methode (Hel98) wurde von K. Helsgaun eine Rundreise mit 24,978 Städten in Schweden berechnet. Diese ist ca. 72.500 km lang und im Mai 2004 wurde unter Verwendung von Branch-and-Cut-Techniken implementiert in Concorde (ABCC05) bewiesen, dass es keine kürzere gibt. Damit ist die Tour durch Schweden das größte bisher gelöste Handlungsreisendenproblem. Eine Welttour mit 1.904.711 geographischen Orten wird aktuell untersucht. Eine bewiesen optimale Tour gibt es zum aktuellen Zeitpunkt noch nicht. Zur Berechnung von Touren dieser Größe sind nicht nur entsprechende Algorithmen notwendig, sondern auch die Nutzung von Großrechenanlagen.

Entscheidend für die Untersuchung des Handlungsreisendenproblems ist, dass die Länge oder auch Kosten der Verbindungen zwischen je zwei Städten unabhängig von der restlichen Rundreise angegeben werden können und die Bewertung kaum Zeit in Anspruch nimmt. Eine Erweiterung der Problemstellung liefert die Forderung der Glattheit der Rundreise und die Beschränkung der Ableitungen (SG00). Anders gesagt soll der Handlungsreisende nun entlang einer glatten Kurve durch die Städte fahren, ohne dort anzuhalten oder an Straßen gebunden zu sein. Daher hat das Problem des motorisierten Handlungsreisenden auch seinen Namen. Abbildung 5.16 zeigt optimale Rundreisen zu verschiedenen vorgegeben Reihenfolgen der Städte. Ein Vergleich gleicher Städteverbindungen unterschiedlicher Rundreisen verdeutlicht die Abhängigkeit einer einzelnen Verbindung von der gesamten Reise.

Das Problem des motorisierten Handlungsreisenden tritt immer dann auf, wenn die Zustände eines Prozesses durch Parameter auswählbare, innere Punktbedingungen erfüllen müssen, und das Problem an diesen Punkten aus Stetigkeitsgründen nicht entkoppelt werden kann.

5.3.1 Modell des motorisierten Handlungsreisenden

Das Fahrzeug des Handlungsreisenden (vgl. Abbildung5.17(a)) wird als einfacher Punkt in der (x_1, x_2) Ebene mit den Koordinaten $x_1(t)$ und $x_2(t)$ betrachtet. Sein Zustand ist durch die Gleichungen

$$\dot{x}_i(t) = f_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \quad i = 1, \ldots, 4$$

beschrieben. Die Zustände $x_{3,min} \leq x_3(t) \leq x_{3,max}$ und $x_{4,min} \leq x_4(t) \leq x_{4,max}$ können dabei entweder die Geschwindigkeit in x_1 und x_2 Richtung, oder Fahrtrichtung und Geschwindigkeit in diese darstellen.

Der Automat in Abbildung 5.17(b), der die Rundreise darstellt, ist sehr einfach aufgebaut. Er besitzt zwei Knoten. Einer der Knoten stellt die Bewegung des Fahrzeugs in der Ebene dar und ist mit der Dynamik des Fahrzeugs und eventuellen Beschränkungen an die Zustände markiert. Der andere stellt den Endzustand des Systems dar. Ein Phasenübergang findet genau dann statt, wenn der Handlungsreisende eine der n_c Städte erreicht. Die Städte werden durch ihre Koordinaten $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n \in \mathbb{R}^2$ angegeben. Bei einem Übergang bleiben die Zustände stetig und die Stadt wird als besucht markiert. Es können nur solche Städte besucht werden, deren Marken noch nicht gesetzt sind. Sind alle Städte mit Marken versehen, kann das Fahrzeug in den Endzustand, dem Verharren in dem Ursprung des Koordinatensystems, übergehen. Als Startwert wird die Position des Fahrzeugs im Ursprung des Koordinatensystems festgesetzt.

Eine Phase des Systems entspricht der Fahrt von einer Stadt $\boldsymbol{c}_i$ zu einer Stadt $\boldsymbol{c}_j$ mit $\boldsymbol{c}_i \neq \boldsymbol{c}_j$. Da nicht klar ist, in welcher Reihenfolge die Städte besucht werden sollen, kann zu jedem Zeitpunkt jede Verbindung zwischen je zwei Städten möglich sein. Da jede Stadt genau einmal besucht werden soll, besitzt das Problem n_c Zeitspannen mit je $n_c(n_c - 1)$ möglichen Phasen und eine abschließende Phase mit n_c Möglichkeiten, in der das Fahrzeug zu seinem Ausgangspunkt zurückkehrt.

Zunächst werden die parallel möglichen Phasen logisch verknüpft. Die Startbedingung einer Phase bzw. Endbedingung der vorhergehenden lautet

$$\bigvee_{j=1}^{n_c} \left[\begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} - \boldsymbol{c}_j = 0 \right]$$

was sich unter Verwendung der binären Variablen $\boldsymbol{q}_{bi}$, $i, j = 1, \ldots, n_c$ äquivalent ist zu

$$\begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix} - \boldsymbol{C}\boldsymbol{q}_{bi} = 0, \quad i = 1, \ldots, n_c$$

mit der Matrix $\boldsymbol{C} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{n_c})$, die die Koordinaten der Städte enthält. Die Dynamik ist in allen Phasen identisch.

Als Kosten der Rundreise können die Fahrzeit, der Steueraufwand oder eine Kombination beider betrachtet werden durch

$$J[\boldsymbol{u}, \boldsymbol{q}_b] = a_1 t_f + a_2 \int_{t_0}^{t_f} \boldsymbol{u}(t)^T \boldsymbol{u}(t) \mathrm{d}\, t.$$

5.3.2 Ergebnisse

Erste obere Schranke mit genetischem Algorithmus

In diesem Abschnitt wird für das Fahrzeug des Handlungsreisenden ein Punktmassemodell betrachtet, das durch die Beschleunigungen $u_1(t)$ in x_1 Richtung und $u_2(t)$ in x_2 Richtung gesteuert wird. Insgesamt lautet das Modell

$$\begin{aligned}
\dot{x}_1(t) &= v_1(t), \quad x_1(0) = 0 = x_1(t_f), \\
\dot{x}_2(t) &= v_2(t), \quad x_2(0) = 0 = x_2(t_f), \\
\dot{v}_1(t) &= u_1(t), \quad v_1(0) = 0 = v_1(t_f), \\
\dot{v}_2(t) &= u_2(t), \quad v_2(0) = 0 = v_2(t_f).
\end{aligned}$$

Im Zielfunktional werden die Konstanten $a_1 = 1$ und $a_2 = 2 \cdot 10^{-3}$ gewählt. Da im Zielfunktional die Zeit linear auftritt, wird die maximale Beschleunigung des Fahrzeugs durch $u_1^2 + u_2^2 \leq 7$ beschränkt.

Um das Branch-and-Bound-Verfahren zu starten, wird ein genetischer Algorithmus zur Berechnung einer ersten oberen Schranke eingesetzt. Dazu wird zunächst eine Anfangspopulation von Rundreisen durch Vorgabe der Reihenfolge der Städte festgelegt. Alle Individuen der Population werden durch Lösen der dazugehörigen Optimalsteuerungsprobleme bewertet.

Zur Erzeugung neuer Rundreisen werden zwei Individuen aus der Population durch eine *Turnierselektion* ausgewählt. Bei einer Turnierselektion werden zufällig k Elemente der Population gewählt und aus diesen die beiden besten für die Kreuzung verwendet. Die gewählten Reisen seien A und B, wobei die Rundreisen durch die Indizes der Städte $\boldsymbol{c}_i$, in der Reihenfolge, wie sie besucht werden, charakterisiert sind. Ein Beispiel für zwei gewählte Rundreisen und deren Kreuzung ist in der Tabelle 5.8 auf der nächsten Seite zu sehen.

Zunächst wird aus dem Elternteil A ein zufälliger Bereich, der *Crossover Bereich* genannt wird und hier schwarz hervorgehoben ist, ausgewählt. Dieser Bereich beschreibt einen Teil der Rundreise und wird in die entsprechenden Positionen der Rundreise des Kindes $\tilde{A}$ übertragen. Aus dem Elternteil B wird, soweit noch Platz ist, elementweise Stadt für Stadt in das Kind kopiert (dunkelgrauer Bereich), wobei die Elemente des Crossover Bereichs von A ausgelassen werden. Die noch nicht verwendeten Städte (mittelgrau) werden in der Reihenfolge, in die sie in B liegen auf die noch leeren Felder von $\tilde{A}$ geschrieben. Das Kind hat einen Teil der Tour von Elternteil A (schwarz) und einen Teil von B (dunkelgrau) geerbt. Um die Population vor Stagnation zu bewahren wird eine neue Information eingebracht, indem ein zufälliger Bereich des Kindes durch seine umgekehrte Reihenfolge ersetzt wird. Das neu entstandene Kind wird durch Lösen des dazugehörigen Optimalsteuerungsproblems bewertet und der Population hinzugefügt. Dieser Prozess wird solange wiederholt, bis die Population eine vorgegebene Größe erreicht. Das beste bis dahin bekannte Individuum wird als erste obere Schranke für den Branch-and-Bound verwendet.

Tabelle 5.9 auf Seite 116 zeigt Ergebnisse zu 5, 6 und 7 Städten. Der obere Teil der Tabelle bezieht sich auf die Berechnung einer oberen Schranke mit dem beschriebenen genetischen Algorithmus. Es sind jeweils die Größe der Anfangspopulation und die maximale Populationsgröße, bei deren

Tabelle 5.8: Genetischer Algorithmus

A	2	6	**10**	**3**	**8**	5	1	7	4	9	Zufällig gewählter Crossover Bereich
B	**7**	3	**6**	**9**	**1**	8	10	**5**	**2**	**4**	Hervorgehobene Elemente werden vererbt
$\tilde{A}$	x	x	**10**	**3**	**8**	x	x	x	x	x	Kopiere schwarzen Bereich von A
$\tilde{A}$	**7**	**6**	**10**	**3**	**8**	**9**	**1**	**5**	**2**	**4**	Kopiere dunkelgrauen Bereich von B mit mittelgrauen Elementen auffüllen

Erreichen das Verfahren abgebrochen wird, angegeben. In der Zeile Auswahlbereich ist die Anzahl der Elemente für die Turnierselektion notiert, und schließlich ist die beste gefundene obere Schranke angegeben, mit der das Branch-and-Bound-Verfahren startet. Falls im oberen Tabellenbereich keine Einträge vorhanden sind, wurde das Branch-and-Bound-Verfahren mit der notierten, vorgegebenen Schranke gestartet.

Im unteren Abschnitt ist zuerst die nächste vom Branch-and-Bound-Verfahren gefundene obere Schranke und die Anzahl der bis dahin untersuchten Knoten notiert. In der folgenden Zeile ist die Anzahl der insgesamt vom Branch-and-Bound-Verfahren untersuchten Knoten und in der letzten Zeile das dabei gefundene Ergebnis zu finden. Bei dem Branch-and-Bound-Verfahren wird eine kleinste Schrankenstrategie verwendet, und jeweils die Variable verzweigt, die am nächsten bei $1/2$ liegt. Als Startwert der Optimierung an einem der Knoten des Baums wird die Lösung des darüberliegenden Knotens verwendet. Die Rechenzeiten variieren zwischen wenigen Minuten für 5 Städte und mehreren Stunden für 7 Städte auf einem Pentium III 900 MHz PC.

Die verschiedenen Berechnungen, die in Tabelle 5.9 auf der nächsten Seite dargestellt sind, zeigen, dass sich die Anzahl der insgesamt zu berechnenden Optimalsteuerungsprobleme durch den Einsatz heuristischer Verfahren erheblich reduzieren lässt. Für ein heuristisches Verfahren, das die Reihenfolge der Städte vorgibt, muss allerdings für die Bewertung eine geeignete Startschätzung der Trajektorie vorliegen, da sonst nicht unbedingt Konvergenz erreicht werden kann.

Ein interessantes Resultat zeigt die letzte Spalte der Tabelle 5.9. Hier wurden annähernd die minimalen Kosten als erste obere Schranke vorgegeben, und dennoch wurden ca. 1000 Knoten untersucht, um die optimale Lösung zu finden. Dies zeigt, dass die unteren Schranken zu niedrig sind, als dass Knoten früher von der Untersuchung ausgeschlossen werden könnten. Dies verstärkt sich sogar noch, wenn an den Knoten konvexe Unterschätzer zur Schrankenberechnung verwendet werden, da hier mit noch niedrigeren Schranken gerechnet werden muss.

Zustandsdiskretisierung

Mit einem leicht geändertem Fahrzeugmodell beschrieben durch die Gleichungen

$$\begin{aligned}
\dot{x}_1(t) &= v(t)\sin(\alpha(t)), & x_1(0) &= 0 = x_1(t_f),\\
\dot{x}_2(t) &= v(t)\cos(\alpha(t)), & x_2(0) &= 0 = x_2(t_f),\\
\dot{v}(t) &= u_v(t), & v(0) &= v(t_f),\\
\dot{\alpha}(t) &= u_\alpha(t), & \alpha(0) &= \alpha(t_f),
\end{aligned}$$

Tabelle 5.9: B&B mit und ohne GA zur Schrankengenerierung

Anzahl der Städte	5	5	6	6	6	7	7	7
GA								
Anfangspopulation	-	10	-	20	20	30	50	-
max. Population	-	20	-	40	60	100	200	-
Aswahlbereich	-	8	-	15	15	25	40	-
Beste obere Schranke	∞	109.05	∞	116.57	112.88	127.96	117.92	112.86
B&B								
nächste obere Schranke	109.05	97.92	140.65	113.79	111.07	124.15	115.0	112.85
berechnet an Knoten	59	101	85	223	279	213	949	725
insgesamt getestete Knoten	195	129	11331	825	647	1931	1397	1058
Lösung	97.92	97.92	111.07	111.07	111.07	112.85	112.85	112.85

wobei die Geschwindigkeit durch $0 \leq v(t) \leq 10$ und die Steuerungen durch $|u_v| \leq 2$ und $|u_\alpha(t)| \leq 1$ beschränkt sind, wird das Handlungsreisendenproblem mit Zustandsdiskretisierung gelöst. Wegen der zyklischen Randbedingungen wird die Rundreise auch im Ursprung glatt, und der Ursprung kann wie jede andere Stadt behandelt werden.

Die einzelnen Phasen des Problems des motorisierten Handlungsreisenden stellen je eine Fahrt von einer Stadt zu einer anderen dar. Bei dem Übergang in die jeweils darauffolgende Phase sind alle Zustände stetig und müssen zusätzlich die Bedingungen

$$\begin{pmatrix} x_1(t_i) \\ x_2(t_i) \end{pmatrix} - \boldsymbol{C}\boldsymbol{q}_{bi} = 0, \quad i = 1, \ldots, n_c$$

mit $\boldsymbol{C} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{n_c})$, der Matrix mit den Koordinaten aller Städte, erfüllen. Damit lautet der zulässige Bereich für die Zustände an einem der Zeitpunkte $t_0, \ldots, t_{n_c}$, zu denen der Handlungsreisende in einer der Städte oder im Ursprung $\boldsymbol{c}_0$ des Koordinatensystems ist

$$\mathbb{R}_x^i = \left\{ (x_1, x_2, v, \alpha)^T \;\middle|\; (x_1, x_2)^T \in \{\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{n_c}\},\ v \in (0, 10),\ \alpha \in (0, 2\pi) \right\}.$$

Diesen Bereich zu diskretisierten entspricht einer Diskretisierung der Durchfahrtsgeschwindigkeit und des Durchfahrtswinkels in der jeweiligen Stadt. Damit ergibt sich

$$\mathbb{D}_x^{i+} = \left\{ (x_1, x_2, v, \alpha)^T \;\middle|\; (x_1, x_2)^T \in \{\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{n_c}\},\ v \in \{v_0, \ldots, v_{n_v}\},\ \alpha \in \{\alpha_0, \ldots, \alpha_{n_\alpha}\} \right\}.$$

Da die Diskretisierung unabhängig von t_i ist und die Zustände stetig in die nächste Phase übergehen, ist $\mathbb{D}_x := \mathbb{D}_x^{i-} = \mathbb{D}_x^{i+}$ mit $i = 0, \ldots, n_c$.

Zu der gegebenen Diskretisierung kann die Familie von Optimalsteuerungsproblemen

$$\min \quad J[\boldsymbol{u}, \boldsymbol{q}_b] = a_1(t_{k+1} - t_k) + a_2 \int_{t_k}^{t_{k+1}} \boldsymbol{u}(t)^T \boldsymbol{u}(t) \mathrm{d}\, t$$

unter den Nebenbedingungen

$$\begin{aligned}
\dot{x}_1(t) &= v(t)\sin(\alpha(t)), \quad \dot{x}_2(t) = v(t)\cos(\alpha(t)),\\
\dot{v}(t) &= u_v(t), \qquad\qquad \dot{\alpha}(t) = u_\alpha(t),\\
0 \le v(t) &\le 10, \quad |u_v| \le 2, \quad |u_\alpha(t)| \le 1,\\
\boldsymbol{x}(t_k) &= \boldsymbol{x}_i \in \mathbb{D}_x \quad \boldsymbol{x}(t_{k+1}) = \boldsymbol{x}_j \in \mathbb{D}_x
\end{aligned}$$

berechnet werden und es ergeben sich jeweils die Kosten c_{ij}. Einige Lösungen zu einer Diskretisierung in $n_v = 5$ und $n_\alpha = 6$ ist in Abbildung 5.18 zu sehen. Da in diesem Beispiel das Fahrzeug so schnell bremsen wie beschleunigen kann, ist eine berechnete Bahn auch optimal, wenn sie in der entgegengesetzten Richtung abgefahren wird. Allerdings gilt nicht $c_{ij} = c_{ji}$, da wegen der Fahrtrichtung eine Hinfahrt mit Anfangswinkel α nur einer Rückfahrt mit Endwinkel $\alpha + (2k+1)\pi$, $k \in \mathbb{N}$ entsprechen kann.

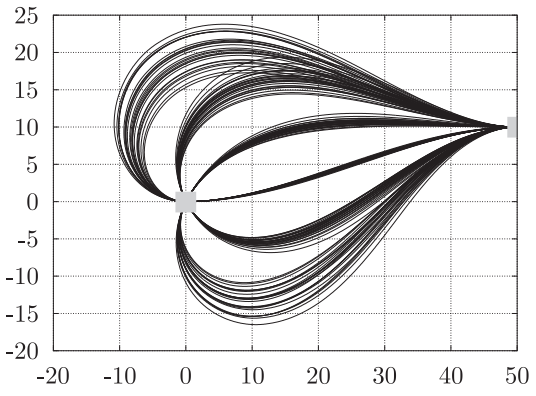


Abbildung 5.18: Verbindungen diskreter Zustände zweier Städte in der (x_1, x_2)-Ebene

Alle Knoten und möglichen Kanten können zu einem Graphen $G = (\mathbb{D}_x, E)$ zusammengefasst werden. Die Kanten $ij \in E$ sind mit binären Variablen ω_{ij} identifiziert und mit den Gewichten c_{ij} ausgestattet. Kanten innerhalb einer Stadt $\mathbb{S}_k = \{(c_{k1}, c_{k2}, v, \alpha)^T \mid v \in \{v_0, \ldots, v_{n_v}\},\ \alpha \in \{\alpha_0, \ldots, \alpha_{n_\alpha}\}\}$ erhalten Gewichte $c_{ij} = 0$.

In diesem Graphen wird durch Lösen eines linearen ganzzahligen Programms ein zulässiger Weg gesucht, der jede Stadt besucht. Zur kompakten Darstellung helfen die Mengen

$$E(Q) := \{ij \in E \mid i, j \in Q\}, \tag{5.33}$$

$$\eta(Q,R)^+ := \{ij \in E \mid i \in Q,\ j \in \mathbb{D}_x \setminus R\}, \tag{5.34}$$

$$\eta(Q,R)^- := \{ij \in E \mid i \in \mathbb{D}_x \setminus R,\ j \in Q\}. \tag{5.35}$$

Die Menge (5.33) beschreibt alle Kanten, die Knoten innerhalb von Q verbinden. (5.34) enthält diejenigen Kanten die von Knoten in Q zu Knoten, die nicht in R liegen, gehen und (5.35) die entgegengesetzter Richtung.

Damit kann ein lineares ganzzahliges Optimierungsproblem

$$\min_{\omega_{ij}} \quad \sum_{ij \in E} \omega_{ij} c_{ij} \tag{5.36}$$

unter den Nebenbedingungen

$$\sum_{ij \in E(\mathbb{S}_k)} \omega_{ij} = 0, \tag{5.37}$$

$$\sum_{ij \in E} \omega_{ij} = n_c + 1, \tag{5.38}$$

$$\sum_{ij \in \eta^+(\{l\}, \mathbb{S}_k)} \omega_{ij} = \sum_{ij \in \eta^-(\{l\}, \mathbb{S}_k)} \omega_{ij}, \quad \forall\, l \in \mathbb{S}_k,\ k \in \mathbb{N}_c, \tag{5.39}$$

$$\sum_{ij \in E(Q)} \omega_{ij} \leq |I| - 1, \quad \forall\, Q = \bigcup_{k \in I} \mathbb{S}_k,\ I \subset \mathbb{N}_c, \tag{5.40}$$

$$\omega_{ij} \in \{0, 1\}, \quad \forall ij \in E, \tag{5.41}$$

formuliert werden. Dieses Optimierungsproblem stellt ein *asymmetrisches Gruppen-Handlungsreisendenproblem* dar, dessen Lösung eine Näherung der gesuchten Rundreise liefert.

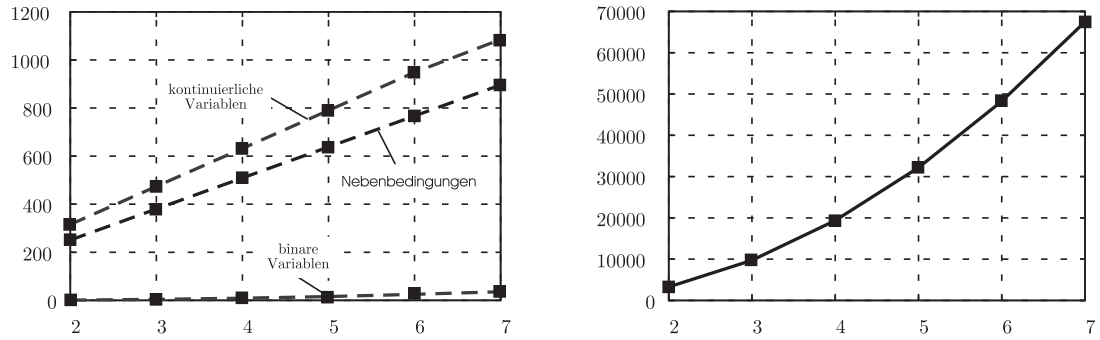
Gleichung (5.37) versichert, dass keine Verbindungen zwischen Knoten $i, j \in \mathbb{S}_k$, die zu einer Stadt gehören, in der Rundreise vorhanden sind. Sind diese Bedingungen erfüllt, bilden die verbleibenden Kanten einen n_c-partiten Graphen, aus dem für eine Rundreise genau $n_c + 1$ Kanten ausgewählt werden müssen (vgl. (5.38)). Die Verknüpfung von eingehenden und ausgehenden Kanten innerhalb der Städte bedingen die Gleichungen (5.39). Mit den bisherigen Gleichungen ist der Übergang innerhalb der Städte gesichert, und es muss nur noch durch Gleichung (5.40) und die Ganzzahligkeitsforderung (5.41) eine Rundreise durch die Städte, d. h. durch die Mengen $\mathbb{S}_k$ gefordert werden. Gleichung (5.40) bedeutet, dass zwischen Knoten jeder echten Teilmenge von $|I|$ Städten maximal $|I| - 1$ Verbindungen vorhanden sein dürfen.

Vergleich Zustandsdiskretisierung und Branch-and-Bound

Das im letzten Abschnitt beschriebene Rundreiseproblem wird nun mit einem Branch-and-Bound-Verfahren gelöst. Zur Traversion des Baums wird eine Tiefensuche verwendet und jeweils die nächste relaxierte Variable verzweigt. Zur Berechnung der Optimalsteuerungsprobleme an den Knoten im Baum wurde als Startwert für ein Homotopieverfahren die Lösung des darüberliegenden Problems genutzt.

Abbildung 5.19(a) auf der nächsten Seite zeigt die Anzahl der Variablen und Nebenbedingungen bei der Lösung des Problems mit einem Branch-and-Bound-Verfahren. Es ist deutlich zu erkennen, dass die Anzahl der binären Variablen im Vergleich mit den kontinuierlichen verhältnismäßig gering ist, jedoch mit steigender Anzahl von Städten schneller anwächst. Die Anzahl der kontinuierlichen Variablen spiegelt die Komplexität der einzelnen Optimalsteuerungsprobleme an den Knoten des Baums wider, die der binären Variablen die Baumgröße und damit die Anzahl der zu lösenden Probleme, bis der Baum durchsucht ist. Beides wächst mit steigender Anzahl der Städte.

Bei einem Ansatz über Zustandsdiskretisierung ist die Anzahl der kontinuierlichen Variablen und



(a) Variablen und Nebenbedingungen bei B&B (b) Variablen des linearen Programms

Abbildung 5.19: Problemdimension bei Branch-and-Bound und Zustandsdiskretisierung

Nebenbedingungen unabhängig von der Anzahl der zu besuchenden Städte, da immer nur einphasige Stadt zu Stad Verbindungen berechnet werden müssen. Kommt eine neue Stadt hinzu, müssen auch nur die Verbindungen von den bisherigen Städten zu dieser und wieder zurück berechnet werden. Alle anderen können wieder verwendet werden. Desweiteren ist die Bahn zwischen zwei Städten in der vorliegenden Formulierung nur abhängig von der relativen Lage der Städte zueinander, nicht aber von deren absoluter Position, weswegen hier gegebenenfalls Einsparungen im Rechenaufwand möglich sind. Zudem könnte die Symmetrie der Bahnen, die sich durch die gleiche maximale Beschleunigung und Verzögerung ergibt, benutzt werden um den Rechenaufwand weiter zu senken. All dies wurde hier nicht berücksichtigt, sondern jede mögliche Trajektorie einzeln berechnet. Lediglich wurde jeweils eine Nachbartrajektorie als Startschätzung einer neuen Berechnung verwendet, um die automatische Berechnung der Gewichte robuster zu gestalten. Abbildung 5.19(b) zeigt die Anzahl der binären Variablen des linearen Problems. Diese wächst mit Anzahl der Städte. Auf die Darstellung der Anzahl der Nebenbedingungen wurde verzichtet, da diese bei einem Branch-and-Cut Algorithmus dynamisch ist.

In Abbildung 5.20 sind die Gesamtrechenzeiten beider Ansätze im Vergleich abgebildet. Es wird deutlich, dass sich bei geringer ganzzahliger Komplexität der Aufwand der Diskretisierung nicht rechtfertigen lässt. Bei steigender diskreter Komplexität und damit auch steigender Komplexität der kontinuierlichen Probleme bei dem Branch-and-Bound-Verfahren ist der Ansatz mit Zustandsdiskretisierung performanter. Hier darf aber nicht vergessen werden, dass es sich im einen Fall um eine exaktere Lösung des kontinuierlichen und diskreten Teils handelt, im anderen Fall nur um eine gröbere Näherung. Es ist auch unklar, ob bei zu schlechter Diskretisierung überhaupt die richtige Kombination diskreter Werte gefunden werden kann. Hier kann ein Vergleich der Approximation durch Zustandsdiskretisierung mit der kontinuierlichen Lösung zu vorgegebener Reihenfolge Aufschluss darüber geben, ob eine Verfeinerung der

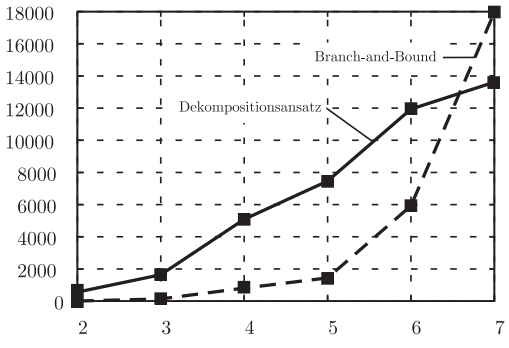
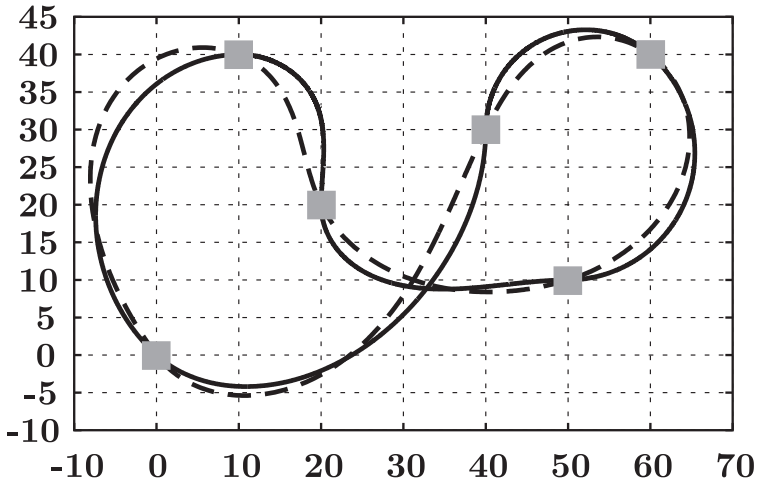
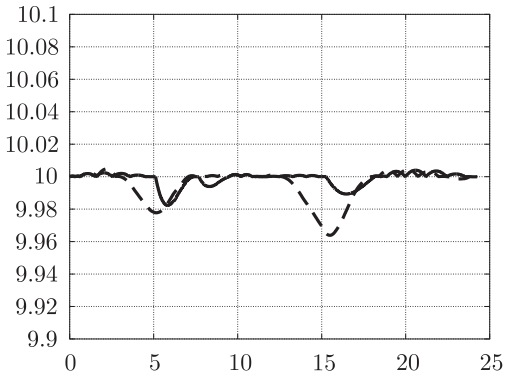


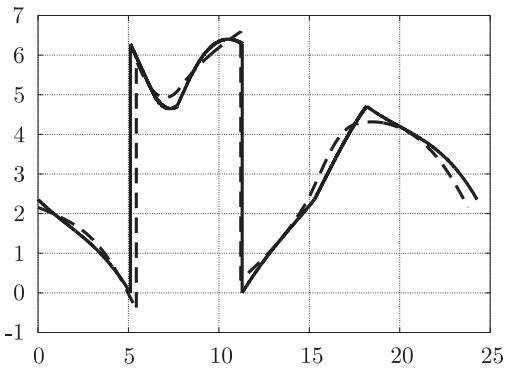
Abbildung 5.20: Rechenzeitvergleich bei Branch-and-Bound und Zustandsdiskretisierung



(a) Rundreisen in (x_1, x_2) Ebene



(b) Geschwindigkeit



(c) Winkel

Abbildung 5.21: Lösungskurven des Branch-and-Bound-Verfahren und des Ansatzes über Zustandsdiskretisierung im Vergleich; Zustandsdiskretisierung (——), Branch-and-Bound-Verfahren (– –)

Diskretisierung mit komplett neuer Suche über dem diskreten Bereich sinnvoll erscheint oder nicht. Ein Vergleich von Lösungen beider Verfahren anhand eines Beispiels ist in Abbildung 5.21(a) zu sehen, die dazugehörigen Geschwindigkeiten und Steuerungen in Abbildung 5.21(b) und Abbildung 5.21(c). Die mit dem Branch-and-Bound-Verfahren minimierten Kosten sind dabei um 7,59% niedriger als die durch Zustandsdiskretisierung optimierten, was eine akzeptable Näherung darstellt. Insbesondere reicht die Näherung aus, um die gleiche Reihenfolge der Städte zu ermitteln.

Kooperatives Team von Handlungsreisenden

Das Problem des motorisierten Handlungsreisenden lässt sich noch erweitern, in dem weiterere Handlungsreisende hinzugenommen werden. Nun stellt sich zum einen die Frage, wie die Hand-

lungsreisenden die Städte untereinander aufteilen sollen, zum anderen, in welcher Reihenfolge ein jeder die ihm zugeteilten Städte besuchen soll.

Das Problem wurde bereits in (Glo00) für zwei Handlungsreisende und sechs Städte untersucht. Die Fahrzeuge werden, wie in Abschnitt 5.3.2 modelliert. Die Beträge der Beschleunigungen und Geschwindigkeiten beider Fahrzeuge sind durch

$$u_{j,1}^2 + u_{j,2}^2 \leq 7 \quad v_{j,1}^2 + v_{j,2}^2 \leq 10 \quad j = I, II$$

beschränkt. Als Zielfunktional wurde die Endzeit verwendet. Damit ergeben sich keine eindeutigen Lösungen, da diese durch den langsameren der beiden Handlungsreisenden festgelegt ist. Wie der schnellere die Zeitdifferenz bis zur Ankunft des langsameren überbrückt, ist offen. Eindeutigkeit kann hier durch weitere Kriterien, wie Energieminimalität erreicht werden.

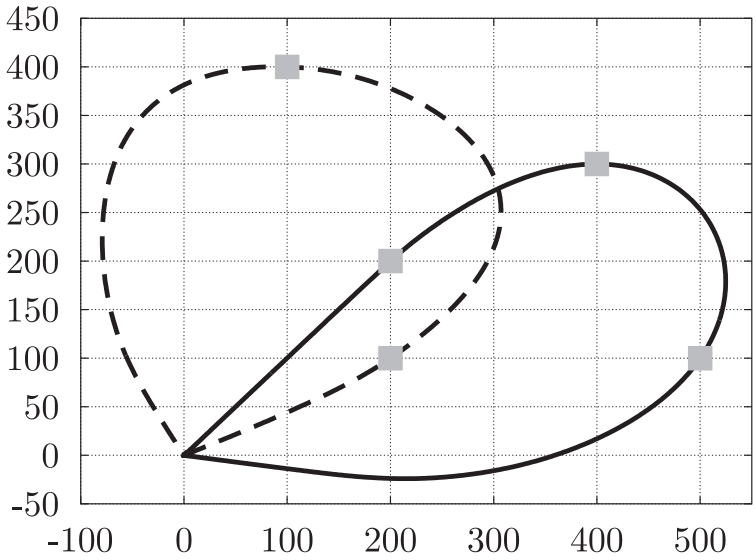
Da die nun zu besuchenden Städte auf zwei Handlungsreisende aufgeteilt werden, wird für jede Stadt eine weitere binäre Variable eingeführt, die entscheidet, welcher der beiden Handlungsreisenden die jeweilige Stadt besucht.

$$\hat{q}_{bi} \begin{pmatrix} x_{I,1}(t_i) \\ x_{I,2}(t_i) \end{pmatrix} + (1 - \hat{q}_{bi}) \begin{pmatrix} x_{II,1}(t_i) \\ x_{II,2}(t_i) \end{pmatrix} - \boldsymbol{C}\boldsymbol{q}_{bi} = 0, \quad i = 1, \ldots, n_c.$$

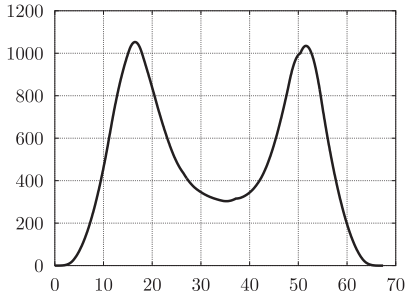
Bei sechs Städten besitzt das Problem $2(5!+4!+3!+2!+1!) = 306$ mögliche Lösungen. Darin sind auch jene inbegriffen, bei denen eines der beiden Fahrzeuge alle Städte besucht. Diese Lösungen sind zwar nicht optimal, werden aber dennoch erlaubt. Abbildung 5.22 zeigt die optimale Verteilung und Abfolge der Städte, sowie die Quadratsummen der Geschwindigkeiten und Beschleunigungen beider Fahrzeuge. Ein Vergleich von Berechnungen mit unterschiedlichen Suchstrategien ist in Tabelle 5.10 aufgelistet. Die kleinste Schrankenstrategie liefert am schnellsten die optimale Lösung, die Anzahl der untersuchten Knoten unterscheidet sich nur wenig von der Breiten- und Tiefensuche. Weitere Ergebnisse können in (Glo00) gefunden werden.

Tabelle 5.10: Teamerweiterung des motorisierten Handlungsreisendenproblem mit verschiedenen Suchtrategien berechnet

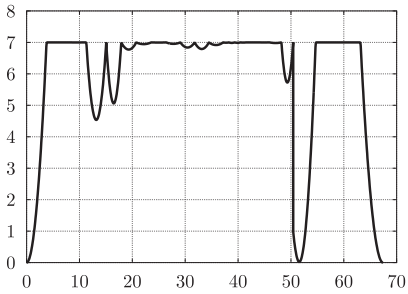
Strategie	Untersuchte Knoten	Ungelöste Teilprobleme	Dauer der Reise
Breitensuche	147	**9**	**67.394**
Tiefensuche	145	10	**67.394**
Kleinste Schranken	**138**	**9**	**67.394**



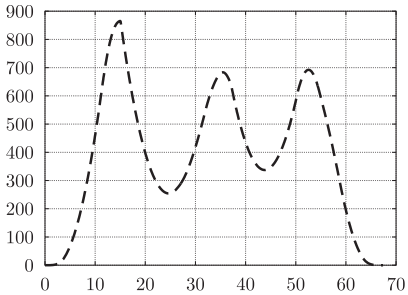
(a) Teamerweiterung



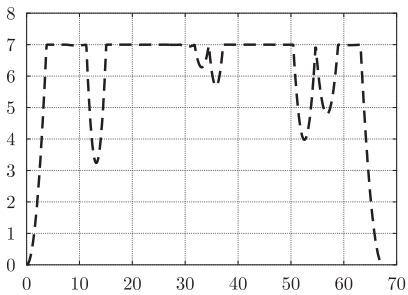
(b) Geschwindigkeit Reisender eins



(c) Beschleunigung Reisender eins



(d) Geschwindigkeit Reisender zwei



(e) Beschleunigung Reisender zwei

Abbildung 5.22: Lösung für ein Team von kooperierenden motorisierten Hundlungsreisenden; Reisender 1 (——), Reisender 2 (– –)

Kapitel 6

Zusammenfassung

Anhand relevanter exemplarischer Problemstellungen wurde eine allgemeine mathematische Formulierung für gemischt diskret-kontinuierliche Optimalsteuerungsprobleme hergeleitet. Basierend auf dieser Formulierung, wurden wichtige Problemklassen definiert, die für die Entwicklung spezieller Verfahren sinnvoll erscheinen. Die Problemklassen sind auf oberster Ebene geschaltete Systeme, die sich in intern und extern geschaltete Systeme weiter untergliedern, und allgemeine hybride Systeme, die entsprechend der Aufgabenstellung, ähnlich wie in der kombinatorischen Optimierung, in weiteren Unterklassen, wie z. B. den Rundreise- oder Maschinenbelegungsproblemen, spezifiziert werden. Die darin enthaltenen Spezialfälle linear quadratischer und zeitdiskreter Probleme wurden gesondert behandelt.

Die Untersuchung der notwendigen Bedingungen bietet zum einen Möglichkeiten der a posteriori Validierung, zum anderen liefert sie Hinweise zur numerisch sinnvollen Formulierung und Entwicklung von Lösungsverfahren. So kann das Maximumprinzip auch auf Steuerungen mit diskretem Wertebereich angewandt werden, womit diese in Spezialfällen durch kontinuierliche Steuerungen ersetzt werden können. Die Hamilton-Jacobi-Bellman-Gleichung dient hingegen als Basis für den in dieser Arbeit untersuchten Dekompositionsansatz.

Die strikte Anwendung des Konzepts der hybriden Automaten mit Steuerungen und Kosten liefert eine geschlossene Herangehensweise zur Modellierung, bis hin zur gemischt binär-kontinuierlichen Problemformulierung unter Einbeziehung von Methoden der kombinatorischen Optimierung.

Zur Lösung der gemischt binär-kontinuierlichen Probleme wurde eine Variante der Branch-and-Bound-Verfahren und ein neuer Dekompositionsansatz vorgeschlagen. Das Branch-and-Bound-Verfahren stellt eine Lösungsmöglichkeit dar, die auf eine Vielzahl von Problemen angewandt werden kann, während das Dekompositionsverfahren besonders bei Rundreiseproblemen mit wenigen stetigen Zustandsvariablen seine Vorzüge zeigt.

Das Branch-and-Bound-Verfahren wurde für verschiedene wichtige Problemstellungen eingesetzt. Erstmalig wurde ein Schedulingproblem für den Landeanflug verschiedener Verkehrsflugzeuge modelliert und berechnet, wobei die Reihenfolge und die Landetrajektorien unter Berücksichtigung einer, für die zivile Luftfahrt gebräuchlichen Aerodynamik numerisch gelöst wurden. Das

Problem der Landereihenfolge zeichnet sich durch einen moderaten diskreten, aber komplexen kontinuierlichen Teil aus, was dazu führt, dass aufgrund des hohen Rechenzeitbedarfs momentan nur eine kleine Anzahl von Flugzeugen betrachtet werden kann. Zudem erschweren lokale Minima eine automatische Lösung des Problems.

Eine weitere interessante Problemstellung, die in dieser Arbeit diskutiert wurde, ist der Roboterfußball. Hier wurden verschiedene Spielsituationen untersucht, die eine Abhängigkeit der Spielstrategie von den dynamischen Bewegungseigenschaften einzelner Spieler zeigen. Echtzeitlösungen eines kompletten Spiels sind bis dato noch nicht möglich, aber offline Betrachtungen zur Planung einzelner Spielzüge sind realistisch.

Beide vorgeschlagenen Verfahren wurden anhand des Problems des motorisierten Handlungsreisenden verglichen. Hier zeigte der Dekompositionsansatz gerade für eine größere Anzahl von Städten, die besucht werden sollen, seine Stärken. Das Rundreiseproblem besitzt eine hohe kombinatorische Komplexität, in der hier untersuchten Variante jedoch einen einfachen kontinuierlichen Anteil. Hier ist die kombinatorische Komplexität der beschränkende Faktor für Probleme mit einer größeren Anzahl von Städten.

Bei den Untersuchungen zu dieser Arbeit hat sich gezeigt, dass die Erzeugung von Berechnungsmodellen aufwendig und fehleranfällig ist. Hier können geeignete Benutzerschnittstellen zur Erzeugung der hybriden Automaten zur Verfügung gestellt werden. Anschließend bietet sich die Benutzung von Werkzeugen der Automatentheorie an, um die Modelle zu verifizieren. Die logischen Zusammenhänge der einzelnen Systemzustände können dann mit einer geeigneten Modellierungssprache beschrieben werden. Anschließend kann das gesamte Problem automatisch in Quellcode übersetzt werden.

Zur Weiterentwicklung von numerischen Verfahren ist es notwendig die Konvexitätstheorie für Optimalsteuerungsprobleme weiter voranzutreiben, damit globale Optimierungsverfahren z. B. innerhalb des vorgeschlagenen Branch-and-Bound-Verfahren und des Dekompositionsansatzes eingesetzt werden können. Für den Dekompositionsansatz bietet sich eine lineare Interpolation des diskretisierten Problems an, womit ein lineares gemischt ganzzahliges Optimierungsproblem zur Berechnung einer Näherung entsteht. Der Vergleich einer Lösung dieses Problems mit einer Lösung zu vorgegebenen diskreten Variablen, kann Aufschluss über die Genauigkeit geben und zur Verfeinerung der Diskretisierung bzw. als Abbruchkriterium genutzt werden. Hier kann auch eine engere Verknüpfung der Verfahren zur Lösung der gemischt ganzzahligen Probleme mit denen zur Lösung von Optimalsteuerungsproblemen in Betracht gezogen werden.

Für die Verfahren zur Lösung von Optimalsteuerungsproblemen sind weitere Unersuchungen zur automatischen Erzeugung geeigneter Starttrajektorien sinnvoll, da gute Startwerte entscheidend für das schnelle und robuste Auffinden von Lösungen sind. Weiter wäre es interessant zu untersuchen, ob Diskretisierungen existieren, die bei Verfeinerung der Diskretisierung wachsende untere Schranken für die Kosten eines Optimalsteuerungsproblems liefern.

Anhang A

Hilfsmittel

A.1 Mathematische Grundlagen

Relationen und Rechenregeln

Die drei grundlegenden logischen Operatoren sind

Konjunktion	‚A und B‘	$A \wedge B$,
Disjunktion	‚A oder B‘	$A \vee B$,
Negation	‚nicht A‘	$\overline{A}$.

Mit ihnen können weitere Operatoren, wie

Implikation	‚aus A folgt B‘	$A \rightarrow B = \overline{A} \vee B$,
Äquivalenz	‚A genau dann, wenn B‘	$A \leftrightarrow B = (A \wedge B) \vee (\overline{A} \wedge \overline{B})$,
Antivalenz	‚A und nicht B oder B und nicht A‘	$A \oplus B = (A \overline{\wedge} B) \vee (\overline{A} \wedge B)$,

zusammengesetzt werden. Für Konjunktion und Disjunktion gelten *Assoziativ-* und *Kommutativgesetz*. Das Distributivgesetz gilt mit der Konjunktion als ‚Mal‘ und der Disjunktion als ‚Plus‘. Weiter gelten die Regeln von *DeMorgan*

$$\begin{array}{rcl} \overline{A \wedge B} & = & \overline{A} \vee \overline{B}, \\ \overline{A \vee B} & = & \overline{A} \wedge \overline{B}. \end{array} \tag{A.1}$$

Konvexität

Definition 10 (Konvexe Menge)**.**
Eine Menge $\mathbb{C} \in \mathbb{R}^n$ heißt *konvexe Menge*, falls für beliebige Paare $x, y \in \mathbb{C}$ auch die Verbindungsgerade $\{z \mid z = (1-\lambda)x + \lambda y,\ \lambda \in [0,1]\}$ in $\mathbb{C}$ enthalten ist.

Bemerkung 16.
Die Schnittmenge zweier konvexer Mengen ist konvex.

Definition 11 (Epigraph)**.**
Gegeben sei eine Funktion $f : \mathbb{C} \to \mathbb{R},\ \mathbb{C} \in \mathbb{R}^n$. Die Menge $\{(x, y) \mid x \in \mathbb{C},\ y \geq f(x)\}$ heißt *Epigraph* von f.

Definition 12 (Konvexe Funktion)**.**
Eine Funktion $f : \mathbb{C} \to \mathbb{R},\ \mathbb{C} \in \mathbb{R}^n$ heißt *konvexe Funktion*, falls ihr Epigraph eine konvexe Menge des $\mathbb{R}^{n+1}$ ist.

Bemerkung 17.
Die *Niveaumenge* einer konvexen Funktion ist konvex.

Graphentheorie

Graphen können als geeignete Mittel zur Darstellung und Untersuchung von Problemstrukturen und diskreten Abläufen verwendet werden. Damit bilden sie die Grundlage für die Untersuchung von Automaten und Netzen. Einen guten Überblick und weitere Informationen zur Graphentheorie kann z. B. in (Die00) oder in Büchern zur kombinatorischen Optimierung wie z. B. (Sch03) gefunden werden.

Definition 13 (Graph)**.**
Ein Graph ist ein geordnetes Paar zweier Mengen $G = (V, E)$. Die Menge V beinhaltet die Ecken oder Knoten des Graphen und die Menge $E \subset (V \times V)$ die Kanten. Die Ecken $i \in V$ und $j \in V$ können mit einer Kante $ij \in E$ verbunden sein.

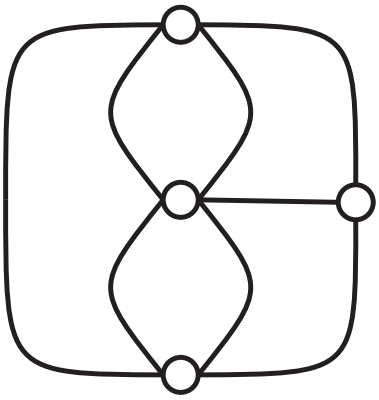


Abbildung A.1: Beispiel eines Graphen

Häufig werden Graphen auch als Tripel (V, E, ψ) beschrieben, mit der *Inzidenzfunktion* $\psi : E \to V \times V$, die den Kanten Knoten zuordnet. Definition 13 verwendet hierfür eine Abkürzung durch die Beschreibung der Kante über ihre Knoten.

Ein Graph heißt *schlicht*, falls er *ungerichtet* ist und keine *Mehrfachkanten* besitzt. Ungerichtet meint dabei im Gegensatz zu *gerichtet*, dass den Kanten keine Richtung zugeordnet ist, also entlang einer Kante von einem zum anderen Knoten gegangen werden kann und wieder zurück. Bei gerichteten Kanten können Schleifen enthalten sein und auch Knoten, die nie erreicht werden. Ein Graph mit Mehrfachkanten bzw. mehreren Kanten zwischen zwei Knoten wird auch *Multigraph* genannt. In Hypergraphen können mehrere Knoten mit einer Kante verbunden werden, diese werden aber hier nicht benötigt.

Ein Graph heißt *zusammenhängend*, falls es zu je zwei Knoten e_i und e_j einen Weg zwischen ihnen entlang der Kanten gibt. Graphen können auch in Teilgraphen zerlegt werden. Ein Teilgraph ist dabei ein Graph $G' = (V', E')$ mit $V' \in V$ und $E' \in E$. Zwei Teilgraphen G' und G'' heißen *disjunkt* falls $V' \cap V'' = \emptyset$ und $E' \cap E'' = \emptyset$. Eine *trennende Kantenmenge* teilt einen Graphen in disjunkte Teilgraphen. Die Menge der Kanten zwischen disjunkten Teilgraphen heißt trennende Kantenmenge. Zwei wichtige trennende Kantenmengen sind Schnitte und speziell Brücken.

Definition 14 (Schnitt)**.**
Ein Schnitt ist eine trennende Kantenmenge, die keine weiter trennende Kantenmenge enthält.

Definition 15 (Brücke)**.**
Ein Schnitt der nur eine Kante enthält heißt Brücke.

Falls sich aus den Ecken des Graphen zwei disjunkte Teilgraphen bilden lassen, so dass keine Verbindungen zwischen den Knoten innerhalb jeder Teilmenge vorhanden sind, so heißt er *bipartit*. Sind alle möglichen Verbindungen zwischen Knoten dieser beiden Teilmengen in E, so heißt er vollständig bipartit. Bei k derartigen Teilmengen wird von *k-partiten* Graphen gesprochen.

Definition 16 (Weg)**.**
Es sei $G = (V, E)$ ein Graph. Ein Teilgraph $G' = (V', E') \subset G = (V, E)$ mit nichtleerer Menge $V' = \{i_0, i_1, \ldots, i_n\}$ paarweise verschiedener Knoten und Kanten $E' = \{i_0i_1, i_1i_2, \ldots, i_{n-1}, i_n\}$ heißt Weg. Oft werden Wege kurz durch die Folge ihrer Knoten beschrieben $G' = i_0i_1i_2 \ldots i_n$.

Definition 17 (Kreis)**.**
Es sei $G' = i_0i_1i_2 \ldots i_{n-1}$ ein Weg. Wird G' um die Kante $i_{n-1}i_0$ zu einem Graphen $K = (V', E' \cup i_{n-1}i_0)$ erweitert, so wird K wird Kreis genannt.

Definition 18 (Hamiltonkreis)**.**
Ein Kreis der alle Ecken eines Graphen enthält, heißt Hamiltonkreis. Graphen, die einen Hamiltonkreis enthalten heißen hamiltonsch.

Komplexität

Definition 19 ($\mathcal{P}$)**.**
$\mathcal{P}$ bezeichnet die Menge aller Probleme, die ein deterministischer Algorithmus in Polynomialzeit löst.

Definition 20 ($\mathcal{NP}$)**.**
$\mathcal{NP}$ bezeichnet die Menge aller Probleme, die ein nichtdeterministischer Algorithmus in Polynomialzeit löst.

Bemerkung 18.
Es gilt dass $\mathcal{P} \subset \mathcal{NP}$, ob allerdings Gleichheit gilt, ist bisher nicht bewiesen.

Bemerkung 19.
Weiter ist zu bemerken, dass es auch weitere Problemklassen oberhalb von $\mathcal{NP}$ gibt (z.B. Probleme, die exponentielle Zeit benötigen).

Definition 21 ($\mathcal{NP}$-hart)**.**
Eine Problem P heißt $\mathcal{NP}$-hart, wenn jedes Problem in $\mathcal{NP}$ darauf reduziert werden kann.

Definition 22 ($\mathcal{NP}$-vollständig)**.**
Eine Problem P heißt $\mathcal{NP}$-vollständig, falls $P \in \mathcal{NP}$ und P $\mathcal{NP}$-hart ist.

Definition 23 (PSPACE)**.**
PSPACE bezeichnet die Klasse der allgemeinen Entscheidungsprobleme, die von einer Turing-Maschine mit polynomialem Speicherplatzbedarf in unbegrenzter Zeit lösbar sind. Diese Definition ist unabhängig davon, ob die Turing-Maschine deterministisch ist oder nicht.

Variationsrechnung

Zum besseren Verständnis der Bedingungen aus Abschnitt 2.3 auf Seite 27 wird an dieser Stelle kurz auf deren Herleitung eingegangen.

Grundlagen

Betrachtet wird die Aufgabe, ein Zielfunktional $J : \mathbb{X} \to \mathbb{R}$ zu minimieren, wobei die Elemente $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{H}$ aus einer Teilmenge $\mathbb{X}$ eines *Hilbertraums* $\mathbb{H}$ stammen. Die Elemente $\boldsymbol{x}$, die auch Variablen genannt werden, sind selbst Funktionen $\boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^{n_x}$, die von einem unabhängigen Parameter abhängen. Hier wird als unabhängiger Parameter die Zeit t verwendet.

Wird nun angenommen, dass $\boldsymbol{x}^* \in \mathbb{X}$ die gesuchte Lösung der Minimierungsaufgabe ist, und eine Schar von Vergleichsfunktionen $\boldsymbol{\xi}(t, \epsilon) \in \mathbb{X},\ \forall\ \epsilon \in \mathbb{R}$ in Abhängigkeit des *Variationsparameters* ϵ mit $\boldsymbol{\xi}(t, 0) = \boldsymbol{x}^*$ gegeben ist, dann besitzt auch die Funktion $F(\epsilon) := J[\boldsymbol{\xi}(t, \epsilon)]$ ein Minimum im Punkt $\epsilon = 0$. Damit $F(\epsilon)$ ein Minimum in $\epsilon = 0$ besitzen kann, muss

$$F_\epsilon(0) = J_{\boldsymbol{x}}[\boldsymbol{x}^*(t)]^T \xi_\epsilon(t, 0) = 0 \tag{A.2}$$

gelten.

Definition 24 (n-te Variation)**.**
Sei eine Funktionenschar $\xi(t, \epsilon)$ mit $\xi(t, 0) = x(t)$ gegeben, die n-mal nach ϵ differenzierbar ist, so heißt

$$\delta^n x := \left.\frac{\partial^n \xi(t, \epsilon)}{\partial \epsilon^n}\right|_{\epsilon=0} \epsilon^n$$

die n*-te Variation* von $x(t)$.

Bemerkung 20.
Wird $\xi(t, \epsilon)$ um $\epsilon = 0$ in ein Taylorreihe

$$\begin{aligned} \xi(t, \epsilon) &= \xi(t, 0) + \left.\tfrac{\partial \xi(t,\epsilon)}{\partial \epsilon}\right|_{\epsilon=0} \epsilon + \tfrac{1}{2} \left.\tfrac{\partial^2 \xi(t,\epsilon)}{\partial \epsilon^2}\right|_{\epsilon=0} \epsilon^2 + \cdots + \tfrac{1}{n!} \left.\tfrac{\partial^n \xi(t,\epsilon)}{\partial \epsilon^n}\right|_{\epsilon=0} \epsilon^n + \ldots \\ &= x(t) + \delta x + \delta^2 x + \cdots + \delta^n x + \ldots \end{aligned}$$

entwickelt und nach dem 2. Term abgebrochen, so ergeben sich spezielle Vergleichsfunktionen. δx entspricht hier den aus der Mechanik bekannten *virtuellen Verrückungen*.

Bemerkung 21.
Vertauschbarkeit von δ-Prozess und Differentiation

$$\frac{\partial}{\partial t}(\delta x) = \frac{\partial}{\partial t}\left(\left.\frac{\partial \xi(t,\epsilon)}{\partial \epsilon}\right|_{\epsilon=0}\epsilon\right) = \frac{\partial}{\partial \epsilon}\left(\frac{\partial \xi(t,\epsilon)}{\partial t}\right)\Bigg|_{\epsilon=0}\epsilon = \delta\dot{x}$$

Bemerkung 22.
Seien $x_a = x(t_a)$ sowie t_a selbst frei und wird als Vergleichsfunktion $\xi(\theta(t,\epsilon),\epsilon)$ gewählt, so gilt

$$\delta x_a = \left.\frac{\partial \xi(\theta(t,\epsilon),\epsilon)}{\partial \epsilon}\right|_{\epsilon=0}\epsilon = \left.\frac{\partial \xi(t_a,\epsilon)}{\partial \epsilon}\right|_{\epsilon=0}\epsilon + \left.\frac{\partial \xi(t_a,\epsilon)}{\partial t}\frac{\partial \theta(t,\epsilon)}{\partial \epsilon}\right|_{\epsilon=0}\epsilon = \delta x(t_a) + \dot{x}(t_a)\delta t_a$$

Herleitung der Bedingungen 1. Ordnung

Im Folgenden wird angenommen, dass das zu minimierende Zielfunktional die Form

$$J[\boldsymbol{x}, t_0, \ldots, t_f] = \Phi_0(\boldsymbol{x}(t_0), t_0) + \sum_{i=1}^{n_c}\left(\Phi_i(\boldsymbol{x}(t_i), t_i) + \int_{t_{i-1}}^{t_i} L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\mathrm{d}\,t\right)$$

mit freien Zeitpunkten t_i besitzt. Um notwendige Bedingungen für ein Minimum zu bestimmen soll zunächst nur ein Intervall $[t_i, t_{i+1}]$ betrachtet werden. Der Lesbarkeit wegen wird t_i mit t_a und t_{i+1} mit t_b bezeichnet. Für die Minimierung von

$$J[\boldsymbol{x}, t_a, t_b] = \Phi(\boldsymbol{x}(t_b), t_b) + \int_{t_a}^{t_b} L(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t)\mathrm{d}\,t$$

werden

$$J[\boldsymbol{x}+\delta\boldsymbol{x}, t_a+\delta t_a, t_b+\delta t_b] = \Phi(\boldsymbol{x}(t_b,\delta t_b)+\delta\boldsymbol{x}(t_b), t_b+\delta t_b) + \int_{t_a+\delta t_a}^{t_b+\delta t_b} L(\boldsymbol{x}(t)+\delta\boldsymbol{x}, \dot{\boldsymbol{x}}(t)+\delta\dot{\boldsymbol{x}}, t)\mathrm{d}\,t$$

$$\delta J = J_{t_a}\delta t_a + J_{t_b}\delta t_b + J_{\boldsymbol{x}}\delta\boldsymbol{x} + J_{\dot{\boldsymbol{x}}}\delta\dot{\boldsymbol{x}} = 0. \tag{A.3}$$

Für die Variation der Anfangszeit und des Anfangszustands ergibt sich

$$J_{t_a}\delta t_a = -L(\boldsymbol{x}(t_a), \dot{\boldsymbol{x}}(t_a), t_a)\delta t_a, \tag{A.4}$$
$$J_{\boldsymbol{x}(t_a)}\delta\boldsymbol{x}(t_a) = 0, \tag{A.5}$$

für die der Endzeit und des Endzustands

$$J_{t_b}\delta t_b = \left(\Phi_{t_b}(\boldsymbol{x}(t_b), t_b) + \Phi_{\boldsymbol{x}(t_b)}(\boldsymbol{x}(t_b), t_b)\dot{\boldsymbol{x}}(t_b) + L(\boldsymbol{x}(t_b), \dot{\boldsymbol{x}}(t_b), t_b)\right)\delta t_b, \tag{A.6}$$
$$J_{\boldsymbol{x}(t_b)}\delta\boldsymbol{x}(t_b) = \Phi_{\boldsymbol{x}(t_b)}(\boldsymbol{x}(t_b), t_b)\delta\boldsymbol{x}(t_b) \tag{A.7}$$

und für die des Zustands im Intervallinneren

$$J_{\boldsymbol{x}}\delta\boldsymbol{x} = \int_{t_a}^{t_b} L_{\boldsymbol{x}}(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t)\delta\boldsymbol{x}\,\mathrm{d}\,t, \tag{A.8}$$

$$\begin{aligned} J_{\dot{\boldsymbol{x}}}\delta\dot{\boldsymbol{x}} &= \int_{t_a}^{t_b} L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t)\delta\dot{\boldsymbol{x}}\,\mathrm{d}\,t \\ &= L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t_b),\dot{\boldsymbol{x}}(t_b),t_b)\delta\boldsymbol{x}(t_b) - L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t_a),\dot{\boldsymbol{x}}(t_a),t_a)\delta\boldsymbol{x}(t_a) - \\ &\quad \int_{t_a}^{t_b} \frac{d}{dt}\left(L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t)\right)\delta\boldsymbol{x}\,\mathrm{d}\,t. \end{aligned} \tag{A.9}$$

Da nun die Variationen im Rahmen der Zulässigkeit beliebig und unabhängig voneinander gewählt werden können, ist es möglich aus Gleichung (A.3) auf der vorherigen Seite durch Nullsetzen bestimmter Variationen ein System von Gleichungen zu gewinnen.

Aus (A.8) und (A.9) ergibt sich für Variationen $\delta\boldsymbol{x}$

$$\int_{t_a}^{t_b} \left(L_{\boldsymbol{x}}(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t) - \frac{d}{dt}\left(L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t)\right)\right)\delta\boldsymbol{x}\,\mathrm{d}\,t = 0.$$

Da auch Variationen mit lokalem Träger zulässig sind, kann gefolgert werden, dass der Integrand selbst gleich Null sein muss. Damit ergeben sich die wohl bekannten Euler-Lagrange-Gleichungen

$$L_{\boldsymbol{x}}(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t) - \frac{d}{dt}\left(L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t),\dot{\boldsymbol{x}}(t),t)\right) = 0. \tag{A.10}$$

Unter Verwendung von Bemerkung 22 auf der vorherigen Seite folgt für Variationen $\delta\boldsymbol{x}_a$ und δt_a aus den Gleichungen (A.4), (A.5) auf der vorherigen Seite und (A.9)

$$\left(-L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t_a),\dot{\boldsymbol{x}}(t_a),t_a)\right)\delta\boldsymbol{x}_a = 0 \tag{A.11}$$

$$\left(-L(\boldsymbol{x}(t_a),\dot{\boldsymbol{x}}(t_a),t_a) + L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t_a),\dot{\boldsymbol{x}}(t_a),t_a)\dot{\boldsymbol{x}}(t_a)\right)\delta t_a = 0 \tag{A.12}$$

und für Variationen $\delta\boldsymbol{x}_b$ und δt_b aus den Gleichungen (A.6), (A.7) und (A.9)

$$\left(\Phi_{\boldsymbol{x}(t_b)}(\boldsymbol{x}(t_b),t_b) + L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t_b),\dot{\boldsymbol{x}}(t_b),t_b)\right)\delta\boldsymbol{x}_b = 0 \tag{A.13}$$

$$\left(\Phi_{t_b}(\boldsymbol{x}(t_b),t_b) + L(\boldsymbol{x}(t_b),\dot{\boldsymbol{x}}(t_b),t_b) - L_{\dot{\boldsymbol{x}}}(\boldsymbol{x}(t_b),\dot{\boldsymbol{x}}(t_b),t_b)\dot{\boldsymbol{x}}(t_b)\right)\delta t_b = 0 \tag{A.14}$$

Bisher wurde der Term Φ_0 noch nicht berücksichtigt. Wird er hinzugefügt, ergibt sich für den Anfangspunkt $t_a = t_0$

$$\left(\Phi_{0\,\boldsymbol{x}(t_0)} - L_{\dot{\boldsymbol{x}}}\right)\Big|_{t=t_0}\delta\boldsymbol{x}_0 = 0 \tag{A.15}$$

$$\left(\Phi_{0\,t_0} - L + L_{\dot{\boldsymbol{x}}}\dot{\boldsymbol{x}}\right)\Big|_{t=t_0}\delta t_0 = 0 \tag{A.16}$$

und für die Endzeit $t_b = t_f$

$$\left(\Phi_{n_c}\, \boldsymbol{x}_{(t_f)} + L_{\dot{\boldsymbol{x}}}\right)\Big|_{t=t_f} \delta \boldsymbol{x}_{n_c} = 0 \tag{A.17}$$

$$\left(\Phi_{n_c\ t_f} + L - L_{\dot{\boldsymbol{x}}}\dot{\boldsymbol{x}}\right)\Big|_{t=t_f} \delta t_f = 0. \tag{A.18}$$

Für einen inneren Punkt t_i muss noch berücksichtigt werden, dass sowohl einmal $t_a = t_i^+$ als auch einmal $t_b = t_i^-$ auftritt und an dieser Stelle $\delta \boldsymbol{x}_a = \delta \boldsymbol{x}_b = \delta \boldsymbol{x}_i$ gilt, womit sich aus (A.11) bis (A.14) auf der vorherigen Seite

$$\left(\Phi_i\, \boldsymbol{x}_{(t_i)}|_{t=t_i} + L_{\dot{\boldsymbol{x}}}|_{t=t_i^+} - L_{\dot{\boldsymbol{x}}}|_{t=t_i^-}\right) \delta \boldsymbol{x}_i = 0 \tag{A.19}$$

$$\left(\Phi_{i\ t_i}|_{t=t_i} + (L - L_{\dot{\boldsymbol{x}}}\dot{\boldsymbol{x}})|_{t=t_i^+} - (L - L_{\dot{\boldsymbol{x}}}\dot{\boldsymbol{x}})|_{t=t_i^-}\right) \delta t_i = 0. \tag{A.20}$$

ergibt. Diese Gleichungen können nur erfüllt werden, falls $\dot{\boldsymbol{x}}(t)$ an der Stelle t_i unstetig sein darf. Die Euler-Lagrange-Gleichungen (A.10) auf der vorherigen Seite bilden zusammen mit den Transversalitätsbedingungen (A.15) bis (A.20) ein Mehrpunkt-Randwertproblem, wobei sich die Randwerte zusammen mit den gegebenen Randbedingungen aus den Transversalitätsbedingungen ergeben.

Die Gleichungen (A.19) und (A.20) müssen ebenfalls an Stellen $\hat{t}_i$ gelten, an denen $\boldsymbol{x}(t)$ eine Ecke aufweist, also nicht stetig differenzierbar ist. In diesem speziellen Fall taucht der Term Φ nicht auf und da sowohl $\hat{t}_i$ als auch $\boldsymbol{x}(\hat{t}_i)$ frei sind, ergeben sich die Gleichungen

$$L_{\dot{\boldsymbol{x}}}|_{t=t_i^-} = L_{\dot{\boldsymbol{x}}}|_{t=t_i^+} \tag{A.21}$$

$$(L - L_{\dot{\boldsymbol{x}}}\dot{\boldsymbol{x}})|_{t=t_i^-} = (L - L_{\dot{\boldsymbol{x}}}\dot{\boldsymbol{x}})|_{t=t_i^+}\,, \tag{A.22}$$

die Weierstraß-Erdmann-Eckbedingungen genannt werden.

A.2 Software

In diesem Abschnitt werden die Softwarepakete beschrieben, die in dieser Arbeit verwendet wurden. Für die Lösung gemischt diskret-kontinuierlicher Optimalsteuerungsprobleme wurde begleitend das Programm MINOCS (mixed integer nonlinear optimal control problem solver) entwickelt. Dieses verwendet zur Lösung von Teilproblemen der nichtlinearen Optimierung SNOPT. Für die Lösung der kombinatorischen Optimierungsprobleme, die bei dem Dekompositionsansatz entstehen, wurde das Branch-and-Cut-and-Price-Verfahren SYMPHONY verwendet.

MINOCS

MINOCS ist eine C++ Implementierung eines Branch-and-Bound-Verfahrens, das an den Knoten des Suchbaums kontinuierliche Optimalsteuerungsprobleme durch direkte Kollokation löst. Es

wurde begleitend zu dieser Arbeit entwickelt und beinhaltet die in Abschnitt 4.1 auf Seite 54 beschriebenen Diskretisierungen. Die Lösung, die mit einer dieser Diskretisierungen berechnet wurde, kann als Startwert für eine Optimierung mit anderer Diskretisierung wieder verwendet werden. Des Weiteren bietet es Möglichkeiten der automatischen Gitterverfeinerung, automatischer Skalierung, sowie der Benutzung des Homotopieverfahrens SICOM (solver independent continuation method).

Die Beschreibung der Struktur des gemischt diskret-kontinuierlichen Optimalsteuerungsproblems ist XML-basiert. Dabei muss jede mögliche Phase mit ihren Variablen und Nebenbedingungen nur einmal angegeben werden. Die Abfolge der Phasen wird unabhängig davon angegeben. Die Verwendung von speziellen XML-Editoren erleichtert dabei die Problembeschreibung maßgeblich.

Für die Implementierung der Nebenbedingungen steht dem Benutzer eine C++ Schnittstelle zur Verfügung. Hier kann auch die Struktur der Jacobimatrix, sowie die Jacobimatrix selbst, angegeben werden.

Die Ergebnisse werden in Dateien ausgegeben und für Gnuplot aufbereitet.

SNOPT

Für die Berechnung der nichtlinearen Optimierungsprobleme, die durch die direkte Kollokation entstehen, kommt SNOPT (GMS97) zur Anwendung. Hierbei handelt es sich um ein vielseitig einsetzbares Programmpaket zur Optimierung nichtlinearer Probleme mit vielen Variablen und Nebenbedingungen. Dabei wird die volle Struktur der dünn besetzten Jacobimatrix ausgenutzt. SNOPT berechnet lokale Lösungen, wobei es robust gegenüber Unstetigkeiten in den Gradienten der Funktionen ist, solange diese nicht zu nahe bei der optimalen Lösung liegen. Es ist dem Benutzer freigestellt, Ableitungen selbst bereitzustellen oder diese durch finite Differenzen von SNOPT approximieren zu lasssen. Falls der Benutzer die Ableitungen selbst bereitstellt, besteht die Möglichkeit, diese von SNOPT mittels der finiten Differenzenapproximation auf Richtigkeit überprüfen zu lassen.

SNOPT verwendet einen SQP-Algorithmus (sequentielle quadratische Programmierung), wie er in Abschnitt 4.1.2 auf Seite 63 beschrieben ist. Dabei ist SNOPT besonders effizient, falls die Anzahl der aktiven Nebenbedingungen nahezu so groß wie die Anzahl der Variablen ist. Da SNOPT verhältnismäßig wenig Funktionsauswertungen zum Auffinden einer Lösung benötigt, ist es besonders geeignet für rechenintensive Probleme.

Da SNOPT in FORTRAN implementiert ist, wurde eine C++ Schnittstelle geschaffen, die die Anbindung an MINOCS erlaubt. Um dem Benutzer eine einheitliche Schnittstelle für die Verwendung von MINOCS und SNOPT zu ermöglichen, wurde auch die Eingabe der Parameter von SNOPT in XML realisiert.

SYMPHONY

SYMPHONY (Ral04) stellt einen in C implementierten Rahmen zur Verfügung, um benutzerspezifische Branch-and-Cut-and-Price-Verfahren (vgl. Abschnitt 4.3.2 auf Seite 81) zu implementieren. Dabei lassen sich über das OSI (open solver interface) lineare Optimierer anbinden. SYMPHONY ist Thread-sicher, was mehrere parallele Berechnungsläufe innerhalb eines Programms ermöglicht. Zu dembesteht die Möglichkeit der Parallelisierung.

Literaturverzeichnis

AA04a ALAMIR, M. ; ATTIA, S. A.: An efficient Algorithm to Solve Optimal Control Problems for Nonlinear Switched Hybrid Systems. In: *Proceedings of the 2004 NOLCOS Symposium on Nonlinear Control Systems*. Stuttgart, Germany, 2004

AA04b ALAMIR, M. ; ATTIA, S. A.: On solving optimal control problems for switched hybrid nonlinear systems by strong variations algorithms. In: *Submitted to: IEEE Transactions On Automatic Control* (2004)

AAF00 ADJIMAN, C.S. ; ANDROULAKIS, I.P. ; FLOUDAS, C.A.: Global optimization of mixed-integer nonlinear problems. In: *AIChE Journal* 46 (2000), Nr. 9, S. 1769 – 1797

AAW05 ATTIA, S. A. ; ALAMIR, M. ; WIT, C. C.: Sub Optimal Ccontrol Of Switched Nonlinear Systems Under Location And Switching Constraints. In: *16th IFAC World Congress, Czech Republic, Accepted for publication*, 2005. – Accepted for publication

AB97 ALLGOR, R.J. ; BARTON, P.I.: Mixed integer dynamic optimization. In: *Computational Chemical Engineering* 21 (1997), S. 451–456

AB03 ALAMIR, M. ; BOYER, F.: Fast Generation of Attractive Trajectories for an Under-Actuated Satellite Application to Feedback Control Design. In: *Optimization and Engineering* 4 (2003), S. 215 – 230

ABCC05 APPLEGATE, D. ; BIXBY, R. ; CHVATAL, V. ; COOK, W.: *CONCORDE TSP Solver*. Website. `http://www.tsp.gatech.edu/concorde.html`. Version: January 2005

ACHH92 ALUR, Rajeev ; COURCOUBETIS, Costas ; HENZINGER, Thomas A. ; HO, Pei-Hsin: Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In: *Hybrid Systems*, 1992, S. 209–229

ADFN98a ADJIMAN, C.S. ; DALLWIG, S. ; FLOUDAS, C.A. ; NEUMAIER, A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs - I. Theoretical advances. In: *Computers and Chemical Engineering* 22 (1998), S. 1137–1158

ADFN98b ADJIMAN, C.S. ; DALLWIG, S. ; FLOUDAS, C.A. ; NEUMAIER, A.: A global optimization method, αBB, for general twice-differentiable constrained NLPs - II. Implementation and Computational Results / Princeton University. 1998. – Forschungsbericht

AG04 AGARWAL, Ashish ; GROSSMANN, I.E.: Modeling for Optimization of Hybrid Dynamic Networks. In: *Submittet to: Computers & Chemical Engineering* (2004), December

Alu99 ALUR, R.: Timed Automata. In: *11th International Conference on Computer-Aided Verification*, Springer-Verlag, 1999, S. 8 – 22

ASF99 ADJIMAN, C.S. ; SCHWEIGER, C.A. ; FLOUDAS, C.A.: Mixed-Integer Nonlinear Optimization in Process Synthesis. In: DU, D.-Z. (Hrsg.) ; PARDALOS, P.M. (Hrsg.): *Handbook of Combinatorial Optimization* Bd. 1, Kluwer Academic Publishers, 1999, S. 1–76

BBM00 BEMPORAD, A. ; BORRELLI, F. ; MORARI, M.: Optimal controllers for hybrid systems: Stability and piecewise linear explicit form. In: *Proceedings of the 39th IEEE Conference on Decision and Control* Bd. 2, 2000, S. 1810 – 1815

Bem04 BEMPORAD, A.: Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. In: *IEEE Transactions on Automatic Control* 49 (2004), May, Nr. 5, S. 832– 838

Bet98 BETTS, J.T.: Survey of numerical methods for trajectory optimization. In: *AIAA J. Guidance, Control, and Dynamics* 21(2) (1998), S. 193–207

BG03 BEMPORAD, A. ; GIORGETTI, N.: Logic-Based Hybrid Solvers for Optimal Control of Hybrid Systems. In: *42th IEEE Conf. on Decision and Control, Maui, Hawaii, USA*, 2003, S. 640–645

BGH+02 BUSS, M. ; GLOCKER, M. ; HARDT, M. ; STRYK, O. von ; BULIRSCH, R. ; SCHMIDT, G.: Nonlinear hybrid dynamical systems: modeling, optimal control, and applications. In: ENGELL, S. (Hrsg.) ; FREHSE, G. (Hrsg.) ; SCHNIEDER, E. (Hrsg.): *Modelling, Analysis and Design of Hybrid Systems* Bd. 279. Berlin, Heidelberg : Springer-Verlag, 2002, S. 311–335

BGKH02 BEMPORAD, A. ; GIORGETTI, N. ; KOLMANOVSKY, I.V. ; HROVAT, D.: A Hybrid System Approach to Modeling and Optimal Control of DISC Engines. In: *41th IEEE Conference on Decision and Control*, 2002, S. 1582–1587

BHS02 BUSS, M. ; HARDT, M. ; STRYK, O. von: Numerical solution of hybrid optimal control problems with applications in robotics. In: *15th IFAC World Congress on Automatic Control, Barcelona, Spain, July 21-26*, Elsevier Science, 2002, S. 2077–2082

BK01 BREDENFELD, Ansgar ; KOBIALKA, Hans-Ulrich: Team Cooperation Using Dual Dynamics. In: HANNEBAUER, Markus (Hrsg.) ; WENDLER, Jan (Hrsg.) ; PAGELLO, Enrico (Hrsg.): *Balancing Reactivity and Social Deliberation in Multi-Agent Systems* Bd. 2103, Hannebauer, Markus, 2001, S. 111 – 124

BL02 BARTON, P.I. ; LEE, C.K.: Modeling, Simulation, Sensitivity Analysis, and Optimization of Hybrid Systems. In: *ACM Transactions on Modeling and Computer Simulation* 12 (2002), (October, Nr. 4, S. 256 – 289

BL04 BARTON, P.I. ; LEE, C.K.: Design of process operations using hybrid dynamic optimization. In: *Computers and Chemical Engineering* 28 (2004), June, Nr. 6-7, S. 955–969

BM99 BEMPORAD, A. ; MORARI, M.: Control of systems integrating logic, dynamics, and constraints. In: *Automatica* 35 (1999), Nr. 3, S. 407–427

BMM99 BEMPORAD, A. ; MIGNONE, D. ; MORARI, M.: An efficient branch and bound algorithm for state estimation and control of hybrid systems. In: *Proceedings of the European Control Conference*. Karlsruhe, Germany, 1999

BP03 BUSSIECK, M.R. ; PRUESSNER, A.: Mixed-integer nonlinear programming. In: *SIAG/OPT Views and News — Newsletter of the SIAM Activity Group on Optimization* 14 (2003), April, Nr. 1, S. 19–22

BSBS00 BUSS, M. ; STRYK, O. von ; BULIRSCH, R. ; SCHMIDT, G.: Towards hybrid optimal control. In: *at – Automatisierungstechnik* 48 (2000), Nr. 9, S. 448–459

BSN+99 BOEL, R.K. ; SCHUTTER, B. D. ; NIJSSE, G. ; SCHUMACHER, J.M. ; SCHUPPEN, J.H. van: Approaches to modelling, analysis, and control of hybrid systems / Delft University of Technology. 1999. – Forschungsbericht

CM01 CUZZOLA, Francesco A. ; MORARI, Manfred: A Generalized Approach for Analysis and Control of Discrete-Time Piecewise Affine and Hybrid Systems. In: *Lecture Notes in Computer Science* 2034 (2001), January, S. 189

Con02 CONWAY, B. A.: Efficient methods for the determination of optimal low-thrust trajectories. In: *2nd Int. Symp, on Low-Thrust Trajectories*. Toulouse, France, Juni 2002

Coo05 COOK, W.: *Traveling Salesman Problem*. Website. `http://www.tsp.gatech.edu/index.html`. Version: March 2005

DFL+04 DYLLA, F. ; FERREIN, A. ; LAKEMEYER, G. ; MURRAY, J. ; OBST, O. ; RÖFER, Th. ; STOLZENBURG, F. ; VISSER, U. ; WAGNER, Th.: Towards a League-Independent Qualitative Soccer Theory for RoboCup. In: *Proceedings of the workshop on Methods and Technology for Empirical Evaluation of Multi-agent Systems and Multi-robot Teams during KI 2004*, 2004

Die00 DIESTEL, Reinhard: *Graphentheorie*. Heidelberg : Springer-Verlag, 2000

DS01 DIAS, M.B. ; STENTZ, A.: A Market Approach to Multirobot Coordination / Robotics Institute, Carnegie Mellon University. 2001. – Forschungsbericht

ED04 EARL, M. G. ; D'ANDREA, R.: Multi-vehicle Cooperative Control Using Mixed Integer Linear Programming. In: *SUBMITTED TO IEEE TRANSACTIONS ON ROBOTICS, NOVEMBER 14, 2004* (2004)

ED05 EARL, M. G. ; D'ANDREA, R.: A Decomposition Approach to Multi-vehicle Cooperative Control. In: *SUBMITTED TO THE IEEE TRANSACTIONS ON ROBOTICS* (2005)

EF00 ESPOSITO, W.R. ; FLOUDAS, C.A.: Global Optimization of Nonconvex Problems with Differential-Algebraic Constraints. In: PIERUCCI, S. (Hrsg.): *European Symposium on Computer Aided Process Engineering* Bd. 10, Elsevier, 2000, S. 73–78

FFG+03 FAHLE, T. ; FELDMANN, R. ; GÖTZ, S. ; GROTHKLAGS, S. ; MONIEN, B.: The Aircraft Sequencing Problem. In: R. KLEIN, L. W. (Hrsg.): *Lecture Notes in Computer Science: Computer Science in Perspective: Essays Dedicated to Thomas Ottmann*. Springer-Verlag GmbH, July 2003, S. 152 – 166

FL94 FLETCHER, R. ; LEYFFER, S.: Solving mixed integer nonlinear programs by outer approximation. In: *Mathematical Programming* 66 (1994), S. 327–349

Flo95 FLOUDAS, C.A.: *Nonlinear and Mixed Integer Optimization*. Oxford University Press, 1995

Flo02 FLOUDAS, C. A.: Mixed-Integer Nonlinear Optimization. In: PARDALOS, P.M. (Hrsg.) ; RESENDE, M. (Hrsg.): *Handbook of Applied Optimization*. Oxford University Press Inc., 2002, Kapitel 9.4, S. 451 ff.

Gam99 GAMKRELIDZE, R.V.: Discovery of the Maximum Principle. In: *Journal of Dynamical and Control Systems* 5 (1999), Nr. 4, S. 437–451

GB98 GALÁN, S. ; BARTON, P.I.: Dynamic Optimization of Hybrid Systems. In: *Computational Chemical Engineering* 22 (1998), S. 183–190

Geo72 GEOFFRION, A.M.: Generalized Benders decomposition. In: *Journal of Optimization Theory and Applications* 10 (1972), Nr. 4, S. 237–260

Ger05 GERDTS, M.: Solving mixed-integer optimal control problems by branch&bound: a case study from automobile test-driving with gear shift. In: *Optimal Control Applications and Methods* 26 (2005), Jan., Nr. 1, S. 1 – 18

GL02 GOUX, J.-P. ; LEYFFER, S.: Solving Large MINLPs on Computational Grids. In: *Optimization and Engineering* 3 (2002), S. 327–346

GL03 GROSSMANN, I.E. ; LEE, S.: Generalized Convex Disjunctive Programming: Nonlinear Convex Hull Relaxation. In: *Computational Optimization and Applications* 26 (2003), Nr. 1, S. 83–100. – ISSN 0926–6003

Glo00 GLOCKER, M.: *Modellierung und Numerik Gemischt-Ganzzahliger Optimalsteuerungsprobleme*, Technische Universität München, Diplomarbeit, 2000

Glo04 GLOCKER, M.: A decomposition approach for optimal control problems with integer inner point state constraints. In: *Proc. Appl. Math. Mech. (PAMM)* Bd. 4, 2004, S. 608–609

GM03 GERKEY, B. ; MATARIĆ, M. J.: A Framework for Studying Multi-Robot Task Allocation. In: *Proceedings of the Second International Naval Research Laboratory Workshop on Multi-Robot Systems*, 2003

GMS97 GILL, P.E. ; MURRAY, W. ; SAUNDERS, M.A.: *SNOPT: An SQP algorithm for large-scale constrained optimization*. University of California, San Diego: Math Dept., 1997. – Report NA 97-2

Gri97 GRIGAT, E.: *Berechnung von Optimalflugbahnen in Fallwindgebieten mittels eines Homotopieverfahrens variabler Ordnung*, Technische Universität München, Dissertation, 1997

Gro02 GROSSMANN, I.: Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques. In: *Optimization and Engineering* 3 (2002), S. 227–252

Gro04 *Kapitel* Advances in Logic-Based Optimization Approaches to Process Integration and Supply Chain Management. In: GROSSMANN, I.E.: *Advances in Chemical Engineering*. Elsevier, 2004, S. to appear

GS02 GLOCKER, M. ; STRYK, O. von: Hybrid optimal control of motorized traveling salesmen and beyond. In: *15th IFAC World Congress on Automatic Control, Barcelona, Spain, July 21-26*, Elsevier Science, 2002, S. 987–992

GVS00 GLOCKER, M. ; VÖGEL, M. ; STRYK, O. von: Trajectory optimization of a shuttle mounted robot. In: *Proc. Workshop on Optimal Control in Hypersonic Flight, Greifswald, 7.-8. Oktober 1999*. München : Hieronymus Verlag, 2000, S. 71–82

Hei98 HEIM, A.: *Modellierung, Simulation und optimale Bahnplanung bei Industrierobotern*, Technische Universität München, Dissertation, 1998

Hel98 HELSGAUN, K.: An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic / Roskilde University. 1998 (81). – DATALOGISKE SKRIFTER (Writings on Computer Science)

Hen00 HENZINGER, T.A.: The theory of hybrid automata. In: INAN, M.K. (Hrsg.) ; KURSHAN, R.P. (Hrsg.): *Verification of Digital and Hybrid Systems*. Springer-Verlag, 2000 (NATO ASI Series F: Computer and Systems Sciences 170), S. 265–292

HR99 HEDLUND, S. ; RANTZER, A.: Optimal control of hybrid systems. In: *Proceedings of the 38th IEEE Conference on Decision and Control* Bd. 4, 1999, S. 3972 – 3977. –

optimal control problems for hybrid systems are studied using dynamic programming techniques

HR02 HEDLUND, S. ; RANTZER, A.: Convex Dynamic Programming for Hybrid Systems. In: *IEEE Transactions on Automatic Control* 47 (2002), September, Nr. 9, S. 1536–1540

HS00 HARDT, M. ; STRYK, O. von: Towards Optimal Hybrid Control Solutions for Gait Patterns of a Quadruped. CLAWAR: International Conference on Climbing and Walking Robots. In: *CLAWAR: International Conference on Climbing and Walking Robots*. Madrid, Spain, October 2000, S. 385–392

JLS03 *Kapitel* 6.43.28.1. In: JOHANSSON, K. H. ; LYGEROS, J. ; SASTRY, S.: *Modeling of hybrid systems*. UNESCO-EOLSS Joint Committee, 2003

KAGB04 KESAVAN, P. ; ALLGOR, R.J. ; GATZKE, E.P. ; BARTON, P.I.: Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. In: *Mathematical Programming* 100 (2004), July, Nr. 3, S. 517 – 535

Kal02 KALLRATH, Josef: *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis*. Braunschweig/Wiesbaden : Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 2002

KJ03 KOBIALKA, Hans-Ulrich ; JAEGER, Herbert: Experiences Using the Dynamical System Paradigm for Programming RoboCup Robots. In: RCKERT, U. (Hrsg.) ; SITTE, J. (Hrsg.): *AMiRE 2003 (2nd International Symposium on Autonomous Minirobots for Research and Edutainment)*, 2003, S. 193–202

KLS03 KELLER, J. ; LEIDEN, K. ; SMALL, R.: Cognitive task analysis of commercial jet aircraft pilots during instrument approaches for baseline and synthetic vision displays. In: FOYLE, D.C. (Hrsg.) ; GOODMAN, A. (Hrsg.) ; HOOEY, B.L. (Hrsg.): *Proceedings of the 2003 Conference on Human Performance Modeling of Approach and Landing with Augmented Displays,*, 2003, S. 15 – 69

Kö05 KÖNIG, A.: *SICOM a Solver-Independent Continuation Method*, Technische Universität Darmstadt, Diplomarbeit, 2005

Ley01 LEYFFER, S.: Integrating SQP and Branch-and-Bound for Mixed Integer Nonlinear Programming. In: *Computational Optimization and Applications* 18 (2001), March, Nr. 3, S. 295–309

LG01 LEE, S. ; GROSSMANN, I.: A Global Optimization Algorithm for Nonconvex Generalized Disjunctive Programming and Applications to Process Systems. In: *Computers and Chemical Engineering* 25 (2001), Nr. 11-12, S. 1675–1697

LRJ01 LADNYI, L. ; RALPHS, T.K. ; JR., L.E. T.: Branch, Cut, and Price: Sequential and Parallel. In: *Lecture Notes in Computer Science* 2241 / 2001 (2001), S. 223

Luc01 LUCCHESI, M.: *Coaching the 3-4-1-2 and 4-2-3-1*. Reedswain Publishing, 2001

Mar01 MARTIN, A.: General Mixed Integer Programming: Computational Issues for Branch-and-Cut Algorithms. In: *Lecture Notes in Computer Science* 2241 / 2001 (2001), S. 1

Mat94 MATARIĆ, M.: *Interaction and Intelligent Behavior*, MIT, Diss., 1994

MBS98 MARECZEK, J. ; BUSS, M. ; SCHMIDT, G.: Robust Global Stabilization of the Underactuated 2-DOF Manipulator R2Dl. In: *Proceedings of the IEEE International Conference on Robotics & Automation Leuven*. Leuven, Belgium, May 1998

MBS99 MARECZEK, J. ; BUSS, M. ; SCHMIDT, G.: Robuste Regelung eines nicht-holonomen, unteraktuierten SCARA Roboters. In: *at - Automatisierungstechnik* 47 (1999), Nr. 5, S. 199 – 208

MCDO05 MERINO, Luis ; CABALLERO, Fernando ; DIOS, J.R. M. ; OLLERO, Anbal: Cooperative Fire Detection using Unmanned Aerial Vehicles. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005

MOE98 MOSTERMAN, P. J. ; OTTER, M. ; ELMQVISTZ, H.: *Modelling petrinets as local constraint equations for hybrid systems using Modelica*. 1998

Mos97 MOSTERMAN, P. J.: *Hybrid Dynamic Systems: A hybrid bond graph modeling paradigm and its application in diagnosis*. Tennessee, Vanderbilt University, Diss., May 1997

MSS99 MENON, P.K. ; SWERIDUK, G.D. ; SRIDHAR, B.: Optimal Strategies for Free-Flight Air Traffic Conflict resolution. In: *Journal of Guidance, Control, and Dynamics* 22 (1999), Nr. 2, S. 202–211

OMHL03 OLDENBURG, J. ; MARQUARDT, W. ; HEINZ, D. ; LEINEWEBER, D.B.: Mixed-logic dynamic optimization applied to batch distillation process design. In: *AIChE Journal* 49 (2003), Nr. 11, S. 2900 – 2917

OREM00 OTTER, M. ; REMELHE, M. ; ENGELL, S. ; MOSTERMAN, P.: Hybrid Models of Physical stems and Discrete Controllers. In: *at – Automatisierungstechnik* 48 (2000), September, Nr. 9, S. 426–437

Pap96 PAPAGEORGIOU, M.: *Optimierung Statische, dynamische, stochastische Verfahren für die Anwendung*. 2. München Wien : Oldenburg, 1996

Par98 PARKER, L. E.: ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. In: *IEEE Transactions on Robotics and Automation* 14 (1998), Nr. 2, S. 220 – 240

Par02 PARKER, L. E.: Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets. In: *Autonomous Robots* 12 (2002), May, Nr. 3, S.

231–255. http://dx.doi.org/10.1023/A:1015256330750. – DOI 10.1023/A:1015256330750

PB94 PESCH, H.J. ; BULIRSCH, R.: The maximum principle, Bellman's equation, and Caratheodory's work. In: *Journal of Optimization Theory and Applications* 80 (1994), Februar, Nr. 2, S. 199–225

PBBY04 POLANI, Daniel (Hrsg.) ; BROWNING, Brett (Hrsg.) ; BONARINI, Andrea (Hrsg.) ; YOSHIDA, Kazuo (Hrsg.): *Lecture Notes in Computer Science*. Bd. 3020 / 2004: *RoboCup 2003: Robot Soccer World Cup VII*. Springer-Verlag GmbH, 2004

PFB02 PALLOTTINO, L. ; FERON, E. ; BICCHI, A.: Conflict Resolution Problems for Air Traffic Management Systems Solved with Mixed Integer Programming. In: *IEEE Transactions on Intelligent Transportation Systems* 3 (2002), March, Nr. 1, S. 3–11

Pic98 PICCOLI, B.: Hybrid systems and optimal control. In: *Proceedings of the 37th IEEE Conference on Decision and Control* Bd. 1, 1998, S. 13 – 18

Pic99 PICCOLI, B.: Necessary conditions for hybrid optimization. In: *Proceedings of the 38th IEEE Conference on Decision and Control* Bd. 1, 1999, S. 410 – 415

Ral04 RALPHS, T.K.: *SYMPHONY Version 4.0 User's Manual*. Bethlehem, PA: Department of Industrial and Systems Engineering, Lehigh University, 2004

Rei79 REIF, J.: Complexity of the mover's problem and generalizations. In: *Proceedings of the 20th IEEE Symposium on Fundations of Computer Science*, 1979, S. 421 – 427

Ret03 RETTIG, U.: *Optimale und robust-optimale Steuerungen: Grundlagen, numerische Berechnung und Anwendung bei der semi-aktiven Kfz-Schwingungsdämpfung*, Technische Universität Darmstadt, Dissertation, 2003

RGS+04 RAGHUNATHAN, A. U. ; GOPAL, V. ; SUBRAMANIAN, D. ; BIEGLER, L. T. ; SAMAD, T.: Dynamic Optimization for 3D Conflict Resolution for Multiple Aircraft. In: *AIAA Journal of Guidance, Control and Dynamics*, 27 (2004), Nr. 4, S. 586–594

RKIZ99 RIEDINGER, P. ; KRATZ, F. ; IUNG, C. ; ZANNE, C.: Linear quadratic optimization for hybrid systems. In: *Proceedings of the 38th IEEE Conference on Decision and Control*, 1999

RS94 REIF, J. H. ; SHARIR, M.: Motion Planning in the Presence of Moving Obstacles. In: *Journal of ACM* 41 (1994), Nr. 4, S. 764 – 790

Rub98 RUBIO, J.E.: The Global Optimization of Variational Problems with Discontinuous Solutions. In: *Journal of Global Optimization* 12 (1998), April, Nr. 3, S. 225 – 237

Rud92 RUDOLPH, H.: The SILP-relaxation method in optimal control: General boundary conditions. In: *Zeitschrift für Analysis und ihre Anwendungen* 11 (1992). – *I:* 1, 143-151; *II:* 3, 431-436.

RZK99 RIEDINGER, P. ; ZANNE, C. ; KRATZ, F.: Time optimal control of hybrid systems. In: *Proceedings of the American Control Conference* Bd. 4, 1999, S. 2466 – 2470

SB92 STRYK, O. von ; BULIRSCH, R.: Direct and indirect methods for trajectory optimization. In: *Annals of Operations Research* 37 (1992), S. 357–373

SB04 SINGER, A.B. ; BARTON, P.I.: Global Solution of Optimization Problems with Parameter-Embedded Linear Dynamic Systems. In: *Journal of Optimization Theory and Applications* 121 (2004), June, Nr. 3, S. 149 – 182. – Communicated by F. Zirilli

Sch86 SCHRIJVER, Alexander: *Theory of Linear and Integer Programming*. Chichester : Wileym, 1986

Sch03 SCHRIJVER, Alexander: *Algorithms and Combinatorics*. Bd. Vol. 24: *Cobinatorial Optimization Polyhedra and Efficiency*. Berlin Heidelberg New York : Springer-Verlag, 2003

Sch05 SCHRIJVER, A.: On the history of combinatorial optimization (till 1960). Version: 2005. `http://homepages.cwi.nl/~lex/files/histco.pdf`. In: AARDAL, K. (Hrsg.) ; NEMHAUSER, G.L. (Hrsg.) ; WEISMANTEL, R. (Hrsg.): *Handbooks in Operations Research and Management Science* Bd. 12. Elsevier

Sei87 SEIDMAN, T.I: Optimal control for switching systems. In: *Proceedings 21st Annual Conf. on Inf. Sci., Syst.* (1987), S. 485 – 489

SG00 STRYK, O. von ; GLOCKER, M.: Decomposition of mixed-integer optimal control problems using branch and bound and sparse direct collocation. In: ENGELL, S. (Hrsg.) ; KOWALEWSKI, S. (Hrsg.) ; ZAYTOON, J. (Hrsg.): *ADPM 2000 – The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems, Dortmund, September 18-19*. Aachen : Shaker, 2000, S. 99–104

SG01 STRYK, O. von ; GLOCKER, M.: Numerical mixed-integer optimal control and motorized traveling salesmen problems. In: *APII – JESA (Journal europen des systmes automatiss - European Journal of Control)* 35 (2001), Nr. 4, S. 519–533

Sha89 SHARIR, M.: Algorithmic motion planning in robotics. In: *Computer* 22 (1989), S. 9 – 19

SSD+02 SIMMONS, R. ; SMITH, T. ; DIAS, M. B. ; GOLDBERG, D. ; HERSHBERGER, D. ; STENTZ, A. ; ZLOT, R. M.: A Layered Architecture for Coordination of Mobile Robots. In: *Multi-Robot Systems: From Swarms to Intelligent Automata, Proceedings from the 2002 NRL Workshop on Multi-Robot Systems*, 2002

Str01 STRYK, O. von: User's Guide for DIRCOL Version 2.1: A direct collocation method for the numerical solution of optimal control problems. / Fachgebiet Simulation und

Systemoptimierung. Technische Universität Darmstadt, 2001. – Forschungsbericht. http://www.sim.informatik.tu-darmstadt.de/sw/dircol

Str03 STRYK, O. von: *Numerical Hybrid Optimal Control and Related Topics*, Technische Universität München, Habilitationsschrift, 2003

Sus99 SUSSMANN, H.J.: A maximum principle for hybrid optimal control problems. In: *Proceedings of the 38th IEEE Conference on Decision and Control* Bd. 1, 1999, S. 425 – 430

Sus00 SUSSMANN, H.J.: Set-valued differentials and the hybrid maximum principle. In: *Proceedings of the 39th IEEE Conference on Decision and Control* Bd. 1, 2000, S. 558 – 563

SV99 STONE, P. ; VELOSO, M.: Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. In: *Artificial Intelligence* (1999)

TB04 TORRISI, F.D. ; BEMPORAD, A.: HYSDEL - A tool for generating computational hybrid models. In: *IEEE Trans. Contr. Systems Technology* 12 (2004), March, Nr. 2, S. 235 – 249

TPK+98 TOMLIN, C.J. ; PAPPAS, G.J. ; KOSECKA, J. ; LYGEROS, J. ; SASTRY, S.S.: Advanced air traffic automation: a case study in distributed decentralized control. In: SICILIANO, B. (Hrsg.) ; VALAVANIS, K.P. (Hrsg.): *Control Problems in Robotics and Automation. Lecture Notes in Control and Information Sciences* Bd. 230, Springer-Verlag, 1998, S. 261–295

VG90 VISWANATHAN, J. ; GROSSMANN, I.E.: A combined penalty function and outer-approximation method for MINLP optimization. In: *Computational Chemical Engineering* 14 (1990), Nr. 7, S. 769–782

WBM87 WATSON, L.T. ; BILLUPS, S.C. ; MORGAN:, A.P.: ALGORITHM 652 - HOMPACK: A Suite of Codes for Globally Convergent Homotopy Algorithms. In: *ACM Transactions on Mathematical Software* 13 (1987), September, Nr. 3

Wit66 WITSENHAUSEN, H.: A class of hybrid-state continuous-time dynamic systems. In: *IEEE Transactions on Automatic Control* 11 (1966), April, Nr. 2, S. 161– 167

WL04 WELGE-LUESSEN, T.: Modelling and Mechanical Design of Underactuated Robot Manipulators with Impact and Friction. In: *Proc. Appl. Math. Mech. (PAMM)* Bd. 4, 2004, S. 175–176

WP95 WESTERLUND, T. ; PETTERSSON, F.: An extended cutting plane method for solving convex MINLP problems. In: *Computational Chemical Engineering Suppl.* 19 (1995), S. 131–136

XA00 XU, X. ; ANTSAKLIS, P.J.: Optimal control of switched systems: new results and open problems. In: *Proceedings of the 2000 American Control Conference* 4 (2000), June, S. 2683 – 2687

XA01 XU, X. ; ANTSAKLIS, P.J.: An Approach for Solving General Switched Linear Quadratic Optimal Control Problems. In: *Proceedings of the 40th IEEE Conference on Decision and Control*. Orlando, Florida, December 2001, S. 2478–2483

XA03 XU, X. ; ANTSAKLIS, P.J.: Results and Perspectives on Computational Methods for Optimal Control of Switched Systems. In: *Lecture Notes in Computer Science* Bd. 2623 / 2003, Springer-Verlag Heidelberg, July 2003, S. 540 – 555

XA04 XU, X. ; ANTSAKLIS, P. J.: Optimal Control of Switched Systems Based on Parameterization of the Switching Instants. In: *IEEE Transactions on Automatic Control* 49 (2004), January, Nr. 1, S. 2–16

Xu01 XU, X.: *Analysis and Design of Switched Systems*. Notre Dame, Indiana, USA, Department of Electrical Engineering, University of Notre Dame, Diss., August 2001

Index

W

Z

Wissenschaftlicher Werdegang

Persönliche Daten

Name	Markus Glocker
Geboren	17. 12. 1973 in München
Nationalität	deutsch
Familienstand	ledig

Schulbildung:

09/1980–07/1984	Grundschule an der Feldmochingerstraße in München
09/1984–07/1993	Wittelsbacher Gymnasium München **Allgemeine Hochschulreife**

Studium:

10/1994–05/2000	Technische Universität München **Diplom Mathematik** Studienrichtung: Maschinenbau

Wissenschaftliche Tätigkeit:

06/2000–10/2000	Technische Universität München **Wissenschaftlicher Angestellter**
11/2000–04/2006	Technische Universität Darmstadt **Wissenschaftlicher Mitarbeiter**

Darmstadt, 2. Mai 2006