Entwicklung und Implementierung eines Verfahrens zur Verwendung der indirekten Methode zur Lösung von Optimalsteuerungsproblemen

I n a u g u r a l d i s s e r t a t i o n

zur

Erlangung des akademischen Grades eines

doctor rerum naturalium (Dr. rer. nat.)

an der Mathematisch-Naturwissenschaftlichen Fakultät

der

Ernst-Moritz-Arndt-Universität Greifswald

vorgelegt von
Andreas May
geboren am 14.01.1982
in Greifswald

Greifswald, 26.08.2011

Dekan:	Prof. Dr. Klaus Fesser
1.Gutachter:	Prof. Dr. Bernd Kugelmann
2.Gutachter:	Prof. Dr. Hans Joachim Oberle
Tag der Promotion:	30.03.2012

Inhaltsverzeichnis

Einleitung **1**

1 Optimale Steuerung **3**

1.1 Einführung 3

1.2 Definition Optimalsteuerungsproblem 6

1.3 Das Maximumprinzip 11

2 Implementierung der Mehrzielmethode **21**

2.1 Das Newton-Verfahren 26

2.1.1 Bestimmung der Ableitungen 26

2.1.2 Dämpfung des Newton-Verfahrens 29

2.1.3 Die Broydenapproximation 30

2.1.4 Abbruchkriterien 32

2.2 Lösen der Anfangswertprobleme 33

2.2.1 Nichtsteife Differenzialgleichungen 34

2.2.2 Steife Differenzialgleichungen 37

2.3 Die Singulärwertzerlegung 42

2.3.1 Theorie 43

2.3.2 Numerische Bestimmung 45

3 Komponenten und Funktionen des Programms **51**

3.1 Erstellung der grafische Oberfläche mit gtkmm und Glade . . 52

3.2 Evaluierung von Strings mit muParser 53

3.3 Erstellen von Grafiken mit gnuplot 57

3.4 Export von Problemstellungen mit pdfLaTeX 58

3.5 Installation 59

3.5.1 Linux 60

3.5.2 Windows 61

3.6 Funktionen und Bedienung 62
3.6.1 Eingabe der Projektdaten 62
3.6.2 Laden und Abspeichern eines Projekts 69
3.6.3 Optionen der Mehrzielmethode 71
3.6.4 Lösung eines Problems, Erstellen von Grafiken und PDF-Export 73
3.6.5 Fehlerbehandlung 76

4 Beispielrechnungen 79
4.1 Das Brachistochrone-Problem 79
4.2 Das Min-Energy-Problem 85
4.2.1 Das unbeschränkte Min-Energy-Problem 86
4.2.2 Das Min-Energy-Problem mit Berührpunkt 89
4.2.3 Das Min-Energy-Problem mit Randstück 92
4.3 Das Rayleigh-Problem 96
4.4 Das Re-Entry-Problem 104

5 Zusammenfassung 111

Abbildungsverzeichnis 113

Tabellenverzeichnis 115

Literaturverzeichnis 117

A Lösungsausgaben des Programms bei den Beispielen 119
A.1 Das Brachistochrone-Problem 119
A.2 Das Min-Energy-Problem 121
A.3 Das Rayleigh-Problem 125
A.4 Das Re-Entry-Problem 132

Einleitung

Die Arbeit beschäftigt sich mit dem numerischen Lösen von Optimalsteuerungsproblemen. Bei diesen Problemen handelt es sich um verallgemeinerte Variationsprobleme, die seit den 50er Jahren des 20. Jahrhunderts untersucht werden. Die Anwendungsgebiete der optimalen Steuerung sind breit gefächert. Sie reichen von physikalischen, biologischen und chemischen Prozessen aus den Naturwissenschaften über die Ingenieurswissenschaften bis hin zu den Wirtschaftswissenschaften. Dabei wird das dynamische Verhalten der Zustände der Probleme meist durch Differenzialgleichungen beschrieben. Auf diese Zustände kann mittels einer oder mehrerer Steuerungen Einfluss genommen werden. In den Abschnitten 1.1 und 1.2 der vorliegenden Arbeit wird zunächst auf die Entwicklung der Optimalsteuerungsprobleme eingegangen und anschließend eine Definition dieser Aufgabenstellung angegeben.

Im Laufe der Jahre wurden verschiedene Methoden zur Lösung solcher Probleme entwickelt. Diese lassen sich in zwei grundlegende Klassen einteilen – direkte und indirekte Methoden. Dabei basiert die bekannteste der indirekten Methoden auf dem sogenannten Maximumprinzip. Dieses liefert notwendige Bedingungen für eine optimale Lösung eines Problems der optimalen Steuerung. Diese Vorgehensweise wird im Abschnitt 1.3 vorgestellt.

Da für die resultierenden Randwertprobleme oft keine geschlossene analytische Lösung angegeben werden kann, müssen diese mit numerischen Verfahren gelöst werden. Dazu wird häufig die sogenannte Mehrzielmethode verwendet, welche auch in diversen Programmen, mit deren Hilfe Optimalsteuerungsprobleme gelöst werden können, implementiert wurde. Diese weisen allerdings einige Nachteile auf, die vor allem die Bedienung erschweren. So entstand bei der Arbeit mit diesen Programmen die Idee, die Mehrzielmethode erneut umzusetzen und dabei den Fokus auf eine intuitive Bedienung zu legen. Dabei soll das Programm den folgenden Anforderungen genügen:

- Die Bedienung soll durch die Verwendung einer grafischen Oberfläche intuitiver und komfortabler gestaltet werden.
- Die Eingabedaten eines Optimalsteuerungsproblems sollen nicht im Quellcode eingetragen werden müssen, sodass der Quellcode unangetastet bleiben kann.
- Für die Benutzung des Programms sollen keine Programmierkenntnisse notwendig sein.

- Der Funktionsumfang bestehender Programme soll erweitert werden, um die Möglichkeiten der Mehrzielmethode besser ausnutzen sowie die Methoden an das jeweilige zu lösende Problem anpassen zu können. Bereits existierende Programme besitzen einen beschränkten Funktionsumfang oder die Nutzung der Funktionen ist aufgrund notwendiger Quellcodeanpassungen unkomfortabel.

- Alle verwendeten Tools sollen frei verfügbar sein, damit die Nutzung des Programms uneingeschränkt erfolgen kann.

Die Mehrzielmethode sowie die verwendeten Methoden werden in Kapitel 2 erläutert. Dabei wird zunächst auf die Mehrzielmethode selbst und anschließend auf das Newtonverfahren, die Anfangswertproblemlöser und die Singulärwertzerlegung eingegangen.

Das 3. Kapitel beschäftigt sich mit der Implementierung der verschiedenen zur Anwendung kommenden Tools sowie der Installation des Programms auf Linux- und Windowssystemen. Den Abschluss dieses Abschnitts bildet eine Beschreibung der Funktionen und der Bedienung des Programms.

Zum Abschluss der Arbeit werden in Kapitel 4 mehrere Optimalsteuerungsprobleme beispielhaft gelöst, um die Funktionsweise des Programms sowie die Nutzung der verschiedenen Optionen zu demonstrieren.

Kapitel 1

Optimale Steuerung

1.1 Einführung

Die Theorie der optimalen Steuerung entwickelte sich aus der seit dem Ende des 17.Jahrhunderts entwickelten Variationsrechnung. Als Geburtsstunde der Variationsrechnung wird häufig das sogenannte Brachistochrone-Problem angesehen, welches Johann Bernoulli im Jahr 1696 anderen berühmten Mathematikern als Herausforderung stellte.[1]

Die Brachistochrone ist diejenige Verbindung zweier Punkte durch eine Kurve, auf der sich ein Massenpunkt unter dem Einfluss der Gravitationskraft am schnellsten von einem Punkt zum anderen bewegt. Diese Kurve ist

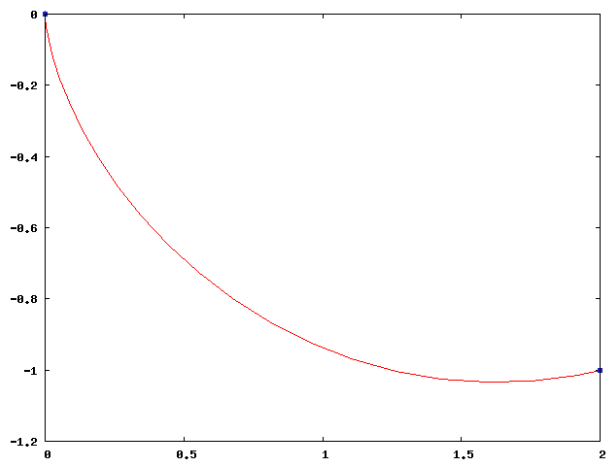


Abbildung 1.1: Brachistochrone $(0,0) \to (-2,-1)$

weder die direkte Verbindung der beiden Punkte durch eine Gerade, noch ein Kreisbogen, sondern das Spiegelbild einer Zykloide. Eine Zykloide oder auch Rollkurve ist die Bahn, die ein Kreispunkt beim Abrollen eines Kreises auf einer Kurve, beispielsweise einer Geraden, beschreibt. Dabei muss die Brachistochrone keine monoton fallende Kurve sein, wie man in Abbildung

[1] U.a. Newton, Leibniz, Jacob Bernoulli, L'Hôpital und von Tschirnhaus.

1.1 gut erkennen kann. Später wird man in Beispiel 1.1.1 sehen, wie man das Problem der Brachistochrone als Optimalsteuerungsproblem formulieren kann.

Die Optimalsteuerungsprobleme entwickelten sich ab 1950 aus den Variationsproblemen. Sie sind aufgrund der Unterscheidung zwischen Steuer- und Zustandsvariablen verallgemeinerte Variationsprobleme. Entscheidenden Anteil am Durchbruch der Theorie der optimalen Steuerung trug der russische Mathematiker Lev S. Pontryagin, dem es gelang, das Maximumprinzip zu beweisen.[2] Dieses liefert notwendige Optimalitätsbedingungen für ein Optimalsteuerungsproblem. Zur Herleitung dieser Bedingungen vergleiche 1.3 und zur Lösung des resultierenden Randwertproblems mittels der Mehrzielmethode Abschnitt 2.

Ein Optimalsteuerungsproblem enthält im Wesentlichen die folgenden Bestandteile:

- Die **Zustandsvariablen** $x(t)$ beschreiben den dynamischen Zustand eines Systems zum Zeitpunkt t.
- Die **Steuervariablen** $u(t)$ ermöglichen die Beeinflussung des dynamischen Verhaltens des Zustands $x(t)$ zum Zeitpunkt t.
- Die **Differenzialgleichungen** $\dot{x}(t) = f(t, x(t), u(t))$ legen die Änderung des Zustands zum Zeitpunkt t in Abhängigkeit des aktuellen Zeitpunkts t, des aktuellen Zustands $x(t)$ und der aktuellen Steuerung $u(t)$ fest.
- Das **Zielfunktional** wird o.B.d.A. minimiert.
- Die **Beschränkungen** für den Zustand $x(t)$ und die Steuerung $u(t)$ müssen eingehalten werden.

Ein Anwendungsgebiet sind beispielsweise die Naturwissenschaften. Dort werden mittels Optimalsteuerungsproblemen physikalische, biologische und chemische Prozesse modelliert. Weitere Anwendungen finden sich sowohl in den Ingenieurs- (Steuerung von Robotern, Flugbahnoptimierungen und Simulation von Testfahrten) als auch den Wirtschaftswissenschaften (Optimierung von Kapitaleinsätzen, Anlagestrategien etc.). Dabei kann das dynamische Verhalten der Zustände der Probleme häufig durch Differenzialgleichungen beschrieben werden. Alle Zustände und Steuerungen, die die Beschränkungen nicht verletzen, werden als zulässige Lösung bezeichnet. Man ist allerdings nicht an irgendeiner zulässigen Lösung interessiert, sondern an einer solchen, die optimal ist, also den Wert einer Zielfunktion minimiert oder maximiert.

Im Folgenden werden Beispiele aus den verschiedenen genannten Gebieten vorgestellt sowie das bereits angesprochene Brachistochrone-Problem als Optimalsteuerungsproblem formuliert (Beispiel 1.1.1).

[2] Etwa zur gleichen Zeit gelang auch Magnus R. Hestenes ein Beweis dieses Satzes.

- biologische Prozesse:
  In biologischen Prozessen kann beispielsweise die dynamische Veränderung von Populationen in einem Zeitintervall durch Differenzialgleichungen beschrieben werden. Dabei kann die Population etwa durch das Fangen von Spezies oder durch gezielte Ausbreitung von natürlichen Feinden beeinflusst werden. Neben natürlichen Beschränkungen, wie der Tatsache, dass die Populationsgröße stets größer gleich Null sein muss, sind weitere Bedingungen denkbar. Dies sind z.B. eine Beschränkung der Fangmenge einer Spezies nach oben oder auch die Forderung, dass die Populationsgröße einen bestimmten Wert nicht unterschreiten darf. Als Optimierungsziel ist die Maximierung der Fangmenge einer Population, ohne dass diese gänzlich ausstirbt, möglich.

- chemische Prozesse:
  In der Chemie können die Vorgänge innerhalb einer chemischen Reaktion mittels Differenzialgleichungen modelliert werden. Eine Steuerung der Reaktion kann durch die Zugabe von bestimmten Stoffen wie Katalysatoren oder die Manipulation der Temperatur geschehen. Häufig ist hierbei eine Limitierung der Temperatur aus Sicherheitsaspekten vonnöten. Das Ziel der Steuerung einer chemischen Reaktion ist in der Regel die Maximierung der Ausbeute eines Reaktionsproduktes.

- Ingenieurswissenschaften:
  Optimalsteuerprobleme tauchen in den Ingenieurswissenschaften vor allem bei der Steuerung von Fahrzeugen, Robotern etc. auf. Dabei werden deren Änderungen der Position und der Geschwindigkeit mit Hilfe von Differenzialgleichungen dargestellt. Die Steuerung kann in einer positiven oder negativen Änderung der Beschleunigung sowie einer Richtungsänderung der Bewegung bestehen. Einzuhaltende Bedingungen sind beispielsweise Beschränkungen der maximalen Beschleunigung, der Position eines Fahrzeugs aufgrund von Hindernissen sowie der Größe der Richtungsänderung. Diese ist z.B. durch einen maximalen Lenkradeinschlag oder das Auftreten von Fliehkräften begründet. Sinnvolle Optimierungsziele bestehen u.a. in der Minimierung des Energieverbrauchs und/oder der Fahrzeit.

- Wirtschaftswissenschaften:
  Hier ist eine Modellierung der zeitlichen Entwicklung des Kapitals eines Unternehmens oder der Produktion möglich. Diese Vorgänge können etwa mittels Einflussnahme auf die Produktionsmenge von Erzeugnissen und der Durchführung von Kapitalanlagen gesteuert werden. Bei Letzterem bedeutet Steuerung nicht nur, in welcher Höhe eine Investition getätigt, sondern ebenso ob diese überhaupt durchgeführt wird. Häufig ist eine Begrenzung des Investitionsrahmens sinnvoll bzw. zwingend, da ein Unternehmen in der Regel nur über eine begrenzte Zahl von Mitteln verfügt. Des Weiteren sind aufgrund der beschränkten Produktionskapazität der vorhandenen Maschinen obere Schranken für die maximal erzeugte Menge eines Produktes denkbar. In ökonomischen Prozessen

ist das Optimierungsziel in der Regel eine Maximierung des erzielten Gewinns.

Beispiel 1.1.1 (Brachistochrone-Problem)[3] Nun soll das Brachistochrone-Problem als Optimalsteuerungsproblem formuliert werden. Dabei nutzt man aus, dass beim Abrollen eines Massenpunktes auf einer Schiene ein Anteil der Schwerebeschleunigung $\vec{g}$ senkrecht zur Schiene als Zwangskraft auf die Schiene wirkt und nicht zur Beschleunigung des Massenpunktes beiträgt. Die Schwerebeschleunigung wird also zerlegt in $\vec{g} = \vec{n} + \vec{a}$, wobei $\vec{n}$ senkrecht auf dem Geschwindigkeitsvektor $\vec{v}$ steht und $\vec{a}$ die auf den Massenpunkt wirkende Beschleunigung beschreibt.
Der Normalanteil der Beschleunigung werde beschrieben durch

$$\vec{n} = u\,\vec{v}^{\perp} = u \begin{pmatrix} v_2 \\ -v_1 \end{pmatrix},$$

wobei $u = u(t)$ ein unbestimmter Skalar (die Steuerung) sei. Die Beschleunigung des Massenpunktes ist damit gegeben durch

$$\vec{a} = \vec{g} - \vec{n} = \begin{pmatrix} -u\,v_2 \\ -g + u\,v_1 \end{pmatrix}$$

und man kann das folgende Optimalsteuerungsproblem formulieren:

Man bestimme die Fallzeit $t_b > 0$ und eine Steuerung $u \in C^1([t_a, t_b], \mathbb{R})$, sodass das Funktional

$$I(x, v, u, t_b) := t_b = \int_0^{t_b} 1\,dt$$

minimiert wird unter den Nebenbedingungen

$$\begin{aligned}
x_1' &= v_1, & x_1(0) &= 0, & x_1(t_b) &= 2,\\
x_2' &= v_2, & x_2(0) &= 0, & x_2(t_b) &= -1,\\
v_1' &= -u\,v_2, & v_1(0) &= 0, & &\\
v_2' &= -g + u\,v_1, & v_2(0) &= 0. & &
\end{aligned}$$

Die Endzeit t_b und die Endgeschwindigkeit $(v_1(t_b), v_2(t_b))^T$ sind frei. □

1.2 Definition Optimalsteuerungsproblem[4]

Definition 1.2.1 (Optimalsteuerungsproblem) *Es sei $[t_0, t_f] \subset \mathbb{R}$ ein nichtleeres, kompaktes Intervall mit Intervallgrenzen $t_0 < t_f$ und $\mathcal{U} \subseteq \mathbb{R}^{n_u}$*

[3] Siehe [12].
[4] Die Ausführungen in diesem Abschnitt sind größtenteils aus [6] entnommen.

eine abgeschlossene Menge. Die Zeitpunkte t_0 und t_f seien o.B.d.A. fest vorgegeben.[5] *Weiterhin seien die folgenden(hinreichend glatten) Abbildungen gegeben:*

$$\begin{aligned}
\varphi &\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}, \\
f_0 &\colon [t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}, \\
f &\colon [t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}, \\
\psi &\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_\psi}, \\
c &\colon [t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}, \\
s &\colon [t_0, t_f] \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_s}.
\end{aligned}$$

Finde einen ***Zustand*** $x \in W^{1,\infty}([t_0, t_f], \mathbb{R}^{n_x})$ *und eine* ***Steuerung*** $u \in L^\infty([t_0, t_f], \mathbb{R}^{n_u})$*, sodass die* ***Zielfunktion***

$$\varphi(x(t_0), x(t_f)) + \int_{t_0}^{t_f} f_0(t, x(t), u(t))\, dt \tag{1.1}$$

minimal wird unter Beachtung der ***Differenzialgleichung***

$$\dot{x}(t) = f(t, x(t), u(t)) \qquad \text{f.ü. in } [t_0, t_f], \tag{1.2}$$

den ***Randbedingungen***

$$\psi(x(t_0), x(t_f)) = 0_{n_\psi}, \tag{1.3}$$

den ***gemischten Steuer- und Zustandsbeschränkungen***

$$c(t, x(t), u(t)) \leq 0_{n_c} \qquad \text{f.ü. in } [t_0, t_f], \tag{1.4}$$

den ***reinen Zustandsbeschränkungen***

$$s(t, x(t)) \leq 0_{n_s} \qquad \text{in } [t_0, t_f], \tag{1.5}$$

und den ***Mengenbeschränkungen***

$$u(t) \in \mathcal{U} \qquad \text{f.ü. in } [t_0, t_f]. \tag{1.6}$$

Bemerkung 1.2.2 Im Folgenden werden einige Begriffe und Definitionen eingeführt:

- Je nach Gestalt der Zielfunktion (1.1) wird das Optimalsteuerungsproblem (1.2.1) auch
  - Bolza-Problem ($\varphi \not\equiv 0$ und $f_0 \not\equiv 0$) oder
  - Mayer-Problem ($\varphi \not\equiv 0$ und $f_0 \equiv 0$) oder
  - Lagrange-Problem ($\varphi \equiv 0$ und $f_0 \not\equiv 0$)

[5] Probleme mit freier Anfangs- und/oder Endzeit können stets in Probleme mit festen Zeitpunkten transformiert werden.

genannt.

- Die Unterscheidung zwischen reinen Zustandsbeschränkungen (1.5) sowie gemischten Steuer- und Zustandsbeschränkungen (1.4) ist aus theoretischer Sicht notwendig, da die beiden Beschränkungstypen unterschiedlich behandelt werden müssen.

- Das Problem (1.2.1) heißt autonom, wenn die Funktionen f_0, f, c, s nicht explizit von der Zeit t abhängen.

- $(x, u) \in W^{1,\infty}([t_0, t_f], \mathbb{R}^{n_x}) \times L^\infty([t_0, t_f], \mathbb{R}^{n_u})$ heißt zulässig für das Optimalsteuerungsproblem (1.2.1), wenn die Bedingungen (1.2) - (1.6) erfüllt sind.

□

Alternative Klassen von Optimalsteuerungsproblemen lassen sich auf die Form in (1.2.1) transformieren. Zunächst soll ein Optimalsteuerproblem mit freier Anfangszeit t_0 und/oder Endzeit t_f umgeformt werden. Solche Problemstellungen können durch die Zeittransformation

$$t(\tau) := t_0 + \tau(t_f - t_0), \qquad \tau \in [0, 1],$$

auf ein Problem mit festem Zeitintervall $\tau \in [0, 1]$ transformiert werden. Dazu definiere man einen neuen Zustand $\bar{x}$

$$\bar{x}(\tau) := x(t(\tau)) = x(t_0 + \tau(t_f - t_0))$$

sowie eine neue Steuerung $\bar{u}$

$$\bar{u}(\tau) := u(t(\tau)) = u(t_0 + \tau(t_f - t_0)).$$

Mittels Differenziation des neuen Zustands bezüglich der neuen Zeitvariablen τ erhält man das neue Differenzialgleichungssystem

$$\begin{aligned}
\bar{x}' &= \dot{x}(t_0 + \tau(t_f - t_0)) \cdot t'(\tau) \\
&= (t_f - t_0) \cdot f(t_0 + \tau(t_f - t_0), x(t_0 + \tau(t_f - t_0)), u(t_0 + \tau(t_f - t_0))) \\
&= (t_f - t_0) \cdot f(t_0 + \tau(t_f - t_0), \bar{x}(\tau), \bar{u}(\tau))).
\end{aligned}$$

Die Größen t_0 und/oder t_f werden als zusätzliche, konstante Zustände interpretiert, die den folgenden Differenzialgleichungen genügen:

$$\begin{aligned}
\dot{t}_0(\tau) &= 0, \qquad t_0(0) \text{ frei}, \\
\dot{t}_f(\tau) &= 0, \qquad t_f(0) \text{ frei}.
\end{aligned}$$

Insgesamt lautet somit das transformierte Problem:

$$\text{Minimiere } \varphi(\bar{x}(0), \bar{x}(1)) + \int_0^1 (t_f - t_0)\, f_0(t_0 + \tau(t_f - t_0), \bar{x}(\tau), \bar{u}(\tau)))\, d\tau$$

unter

$$\begin{aligned}
\bar{x}'(\tau) &= (t_f - t_0)\, f(t_0 + \tau(t_f - t_0), \bar{x}(\tau), \bar{u}(\tau)) \\
&\quad \text{f.ü. in } [0,1], \\
\psi(\bar{x}(0), \bar{x}(1)) &= 0_{n_\psi}, \\
c(t_0 + \tau(t_f - t_0), \bar{x}(\tau), \bar{u}(\tau)) &\le 0_{n_c} \qquad \text{f.ü. in } [0,1], \\
s(t_0 + \tau(t_f - t_0), \bar{x}(\tau)) &\le 0_{n_s} \qquad \text{in } [0,1], \\
\bar{u}(\tau) &\in \mathcal{U} \qquad \text{f.ü. in } [0,1].
\end{aligned}$$

Ist ein nichtautonomes Optimalsteuerungsproblem gegeben, so kann dieses durch Einführung eines zusätzlichen Zustands

$$\dot{T} = 1, \qquad T(t_0) = t_0$$

in ein äquivalentes autonomes Problem transformiert werden. Dabei ist die explizit auftretende Zeit t in den Funktionen f_0, f, c, s durch den neu eingeführten Zustand T zu ersetzen und man erhält folgendes autonomes Optimalsteuerungsproblem:

$$\text{Minimiere } \varphi(x(t_0), x(t_f)) + \int_{t_0}^{t_f} f_0(T(t), x(t), u(t)))\, dt$$

unter

$$\begin{aligned}
\dot{x}(t) &= f_0(T(t), x(t), u(t)) \qquad \text{f.ü. in } [t_0, t_f], \\
\dot{T}(t) &= 1, \\
\psi(x(t_0), x(t_f)) &= 0_{n_\psi}, \\
T(t_0) &= t_0, \\
c(T(t), x(t), u(t)) &\le 0_{n_c} \qquad \text{f.ü. in } [t_0, t_f], \\
s(T(t), x(t)) &\le 0_{n_s} \qquad \text{in } [t_0, t_f], \\
u(t) &\in \mathcal{U} \qquad \text{f.ü. in } [t_0, t_f].
\end{aligned}$$

Ebenso ist es möglich ein Bolza-Problem durch Transformation in ein Mayer-Problem umzuwandeln. Betrachte dazu die Zielfunktion (1.1):

$$\varphi(x(t_0), x(t_f)) + \int_{t_0}^{t_f} f_0(t, x(t), u(t))\, dt.$$

Um den Integralterm der Zielfunktion zu eliminieren, wird ein neuer Zustand z mit

$$\dot{z}(t) = f_0(t, x(t), u(t)), \qquad z(t_0) = 0$$

eingeführt. Dann gilt:

$$z(t_f) = z(t_0) + \int_{t_0}^{t_f} f_0(t, x(t), u(t))\, dt = \int_{t_0}^{t_f} f_0(t, x(t), u(t))\, dt.$$

Die Bolza-Zielfunktion (1.1) kann also durch die äquivalente Mayer-Zielfunktion

$$\phi(x(t_0), x(t_f)) + z(t_f)$$

ersetzt werden.

Zuletzt soll ein Tschebyscheff-Problem in ein Optimalsteuerproblem der Form (1.2.1) transformiert werden. Unter einem Tschebyscheff-Problem versteht man die Aufgabe, die Funktion

$$\max_{t \in [t_0, t_f]} h(t, x(t), u(t))$$

unter den Nebenbedingungen (1.2)-(1.6) zu minimieren. Man führt eine Größe $\alpha \in \mathbb{R}$ ein, für die gilt:

$$h(t, x(t), u(t)) \leq \alpha \qquad \text{f.ü. in } [t_0, t_f]. \tag{1.7}$$

Dies ist eine zusätzliche gemischte Steuer- und Zustandsbeschränkung. Das Tschebyscheff-Problem ist äquivalent mit

$$\alpha \to \min$$

unter den Nebenbedingungen (1.2)-(1.7). Den Parameter α kann man entweder als reelle Optimierungsvariable mitführen oder als zusätzlichen, konstanten Zustand interpretieren, der der Differenzialgleichung

$$\dot{\alpha} = 0, \qquad \alpha(t_0) \text{ frei}$$

genügt. Da $\alpha(t)$ konstant ist, gilt $\alpha(t) = \alpha(t_f)$ für alle t und es entsteht ein Mayer-Problem. Insgesamt ist damit das Tschebyscheff-Problem durch die Einführung eines zusätzlichen Zustandes sowie einer gemischten Steuer- und Zustandsbeschränkung auf die Form (1.2.1) transformiert worden.

Bemerkung 1.2.3 Die beschriebenen Transformationen vergrößern durch die Einführung zusätzlicher Differenzialgleichungen die Dimension des Differenzialgleichungssystems, wodurch der Rechenaufwand erhöht wird. Allerdings besitzen diese oft eine rechte Seite identisch Null - die Lösung der entsprechenden Differenzialgleichung ist also eine Konstante. Um die Erhöhung des Rechenaufwandes zu vermeiden bzw. zu verringern, kann die Lösung einer solchen Differenzialgleichung im vorliegenden Programm als unbekannte konstante Größe behandelt werden. □

Es gibt zwei grundlegende Klassen von Lösungsmethoden, um Probleme der optimalen Steuerung zu lösen. Direkte Lösungsmethoden diskretisieren das Optimalsteuerungsproblem, um so ein endlich-dimensionales Problem zu erhalten, welches dann durch iterative numerische Verfahren gelöst wird. Die zweite hier verfolgte Variante sind die indirekten Methoden, die auf der Auswertung von notwendigen Bedingungen basieren, um das Optimalsteuerungsproblem durch ein endlich-dimensionales Problem zu approximieren. Die resultierende Aufgabe wird dann analytisch oder numerisch gelöst. Das im folgenden Abschnitt behandelte Maximumprinzip gehört dieser Klasse von Methoden an. Dabei wird das ursprüngliche Optimalsteuerungsproblem durch

Auswertung von notwendigen Bedingungen in ein Mehrpunktrandwertproblem überführt, welches dann analytisch oder numerisch gelöst werden kann. Allerdings ist die Lösung lediglich ein Kandidat für ein Optimum, sodass diese durch Auswertung von hinreichenden Bedingungen auf Optimalität zu überprüfen ist. In dem dieser Arbeit zugrunde liegenden Programm wird dieses Mehrpunktrandwertproblem mit der Mehrzielmethode gelöst.[6]

1.3 Das Maximumprinzip

In diesem Kapitel werden die aus dem Maximumprinzip folgenden notwendigen Bedingungen dargestellt.[7] Betrachtet man das Optimalsteuerungsproblem (1.2.1) zunächst ohne Beschränkungen an die Steuerung und/oder die Zustände, führt die Anwendung des Maximumprinzips im Allgemeinen auf ein gewöhnliches Randwertproblem, welches dann beispielsweise mit der in Abschnitt 2 vorgestellten Mehrzielmethode gelöst werden kann. Enthält das zu lösende Optimalsteuerungsproblem hingegen komplexere Bedingungen, so resultiert die Anwendung des Maximumprinzips häufig in sogenannten Mehrpunktrandwertproblemen. Diese weisen neben den Differenzialgleichungen und Randbedingungen auch Zeitpunkte auf, an denen die Steuerung geändert, also umgeschaltet wird. Des Weiteren können die Adjungierten an diesen Schaltpunkten Unstetigkeiten aufweisen, sogenannte Sprünge, die unbekannte konstante Größen sind. Damit sind solche Mehrpunktrandwertprobleme deutlich komplexer als gewöhnliche Randwertprobleme, was sich auch in der für solche Probleme angepassten Mehrzielmethode niederschlägt.

Man betrachte also zunächst das folgende vereinfachte Optimalsteuerungsproblem:

Definition 1.3.1 ***(Optimalsteuerungsproblem 2)*** *Dieses Optimalsteuerungsproblem enthalte im Gegensatz zum Optimalsteuerungsproblem (1.2.1) keine gemischten Steuer- und Zustandsbeschränkungen (1.4) als auch keine reinen Zustandsbeschränkungen (1.5).*

Ferner werden die Definitionen der Hamiltonfunktion sowie des Systems der adjungierten Differenzialgleichungen benötigt.

Definition 1.3.2 (Hamiltonfunktion) *Die Funktion*

$$H(t, x, u, \lambda_0, \lambda) := \lambda_0 \, f_0(t, x, u) + \lambda' \, f(t, x, u) \tag{1.8}$$

mit $\lambda_0 \in \mathbb{R}$ und $\lambda \in \mathbb{R}^n$ heißt Hamiltonfunktion für das Problem (1.3.1).

[6] Die Mehrzielmethode sowie die Lösung von Randwertproblemen mit diesem Verfahren wird in Abschnitt 2 behandelt.

[7] Dieser Abschnitt folgt den Ausführungen in [14] S.147-205. Zu ausführlichen Darstellungen sowie einer großen Zahl an Beispielen sei hierauf sowie [17] verwiesen.

Definition 1.3.3 ***(Adjungiertes Differenzialgleichungssystem)*** *Bezüglich der Hamiltonfunktion (1.8) und Funktionen $(x(\cdot), u(\cdot))$, die den Differenzialgleichungen (1.2) genügen, wird das folgende System von linearen Differenzialgleichungen*

$$\dot{\lambda}(t) = -\frac{\partial H(t, x, u(t), \lambda_0(t), \lambda(t))}{\partial x}\bigg|_{x=x(t)}' \tag{1.9}$$

$$\dot{\lambda}_0(t) = 0 \tag{1.10}$$

als adjungiertes Differenzialgleichungssystem für das Optimalsteuerungsproblem (1.3.1) bezeichnet.

Die Gleichung (1.2) kann ersetzt werden durch

$$\dot{x}(t) = \left(\frac{\partial H(t, x(t), u(t), \lambda_0, \lambda}{\partial \lambda}\right)'. \tag{1.11}$$

Die Gleichungen (1.11) und (1.9) bilden das sogenannte Hamilton-System. Das folgende Resultat gilt für Optimalsteuerungsprobleme der Form (1.3.1).

Satz 1.3.4 *Sei $u^o(\cdot) \in \mathcal{U}$ eine auf dem Intervall $[t_0^o, t_f^o]$, $t_0^o < t_f^o$ definierte Steuerung, welche die Zustände des Systems (1.2) von einem zulässigen Punkt $x^o(t_0^o) \in \mathbb{R}^n$ in einen zulässigen Punkt $x^o(t_f^o) \in \mathbb{R}^n$ überführt. Eine notwendige Bedingung für die Optimalität des Tupels $(t_0^o, t_f^o, x^o(\cdot), u^o(\cdot))$ ist die Existenz sowohl der $\vartheta_i \in \mathbb{R}$, $i = 1, \ldots, n$ als auch einer Lösung $\lambda^o(\cdot)$, λ_0^o des adjungierten Differenzialgleichungssystems (1.9) bezüglich $(x^o(\cdot), u^o(\cdot))$, sodass die folgenden Bedingungen erfüllt sind:*

(i)
$$H(t, x^o(t), u^o(t), \lambda_0^o, \lambda^o(t)) \leq H(t, x^o(t), u, \lambda_0^o, \lambda^o(t)) \quad t \in [t_0^o, t_f^o],\ f.\ ü.;\ \forall u \in \mathcal{U}$$

(ii)
$$\lambda_0^o \geq 0$$

(iii)
$$\begin{pmatrix} \lambda_0^o \\ \lambda^o(\cdot) \end{pmatrix} \neq 0$$

Transversalitätsbedingungen (Diese gelten nur, wenn die Endzeit und/oder Startzeit frei sind.):

(iv)
$$H(t_f^o, x^o(t_f^o), u^o(t_f^o), \lambda_0^o, \lambda^o(t_f^o)) + \lambda_0^o \left.\frac{\partial \varphi(t, x^o(t_f^o))}{\partial t}\right|_{t=t_f^o} = 0$$

(v)
$$H(t_0^o, x^o(t_0^o), u^o(t_0^o), \lambda_0^o, \lambda^o(t_0^o)) + \lambda_0^o \left.\frac{\partial \varphi(t, x^o(t_0^o))}{\partial t}\right|_{t=t_0^o} = 0$$

Orthogonalitätsbedingungen:

$$\text{(vi)} \qquad \lambda^o(t_f^o) - \lambda_0^o \left.\frac{\partial \varphi(t_f^o, x)}{\partial x}\right|'_{x=x^o(t_f^o)} = \sum_{i=1}^{n} \vartheta_i \left.\frac{d\psi_i(x)}{dx}\right|'_{x=x^o(t_f^o)},$$

$$\text{(vii)} \qquad \lambda^o(t_0^o) - \lambda_0^o \left.\frac{\partial \varphi(t_0^o, x)}{\partial x}\right|'_{x=x^o(t_0^o)} = \sum_{i=1}^{n} \vartheta_i \left.\frac{d\psi_i(x)}{dx}\right|'_{x=x^o(t_0^o)}.$$

Die Bedingung (i) verlangt, dass die optimale Steuerung $u^o(t)$ die Hamiltonfunktion fast überall minimiert. Somit muss ein Kandidat für eine optimale Steuerung eine Nullstelle der ersten Ableitung der Hamiltonfunktion sein. Eine Steuerung, die sich auf einer Menge vom Maß Null von der optimalen Steuerung unterscheidet, ist ebenfalls optimal. Die sogenannten Transversalitätsbedingungen (iv) und (v) gelten nur, wenn die Endzeit und/oder die Startzeit nicht gegeben sind. Ist die Endzeit bzw. die Startzeit bekannt, muss eine optimale Lösung die entsprechende Transversalitätsbedingung (iv) bzw. (v) nicht erfüllen. Die Orthogonalitätsbedingungen (vi) und (vii) werden häufig gleichermaßen als Transversalitätsbedingungen bezeichnet. Ist ein Zustand am End- und/oder Startzeitpunkt vollständig vorgegeben, so ist die entsprechende Orthogonalitätsbedingung dennoch erfüllt.

Bemerkung 1.3.5 Es ist zu beachten, dass Satz (1.3.4) lediglich notwendige Bedingungen liefert. Daher stellt ein Tupel $(x(\cdot), u(\cdot), t_0, t_f))$, welches die Bedingungen des Problems erfüllt und es erlaubt, eine Lösung (λ, λ_0) des zugehörigen adjungierten Systems, die den Bedingungen des Satzes (1.3.4) genügt, zu bestimmen, lediglich einen Kandidaten für eine optimale Lösung dar. □

Bemerkung 1.3.6 (Entartete Probleme) Betrachte im Folgenden ein Paar (λ^o, λ_0^o), welches den Bedingungen von Satz (1.3.4) genügt. Dann erfüllt das Paar $(\hat{\lambda}^o, \hat{\lambda}_0^o)$ mit $\hat{\lambda}^o = \alpha\lambda^o$, $\hat{\lambda}_0^o = \alpha\lambda_0^o$, $\alpha > 0$ ebenfalls die fraglichen Bedingungen. Man kann also stets $\lambda_0^o = 1$ setzen, sofern $\lambda_0^o \neq 0$ gilt. Den Fall $\lambda_0^o = 0$ bezeichnet man als entarteten Fall. □

Bemerkung 1.3.7 (Freie End- und/oder Startzustände) In vielen Optimalsteuerungsproblemen müssen der End- oder Startzustand keiner speziellen Beschränkung genügen, d.h. ein Teil der End- und Startzustände ist frei. Dann folgt aus den Orthogonalitätsbedingungen (vi) bzw. (vii), dass die zum freien Zustand x_i zugehörige Adjungierte λ_i im Endpunkt bzw. im Startpunkt den Wert Null annimmt. □

Bemerkung 1.3.8 (Autonome Probleme) Ist ein autonomes Optimalsteuerungsproblem gegeben, so kann man zeigen, dass die Hamiltonfunktion entlang der Lösung konstant ist:

$$H(x^o(t), u^o(t), \lambda_0^o, \lambda^o(t))) = const.$$

Dies liefert, dass das Tupel $x^o, u^o, \lambda_0^o, \lambda^o$ die Bedingung (i) des Satzes 1.3.4 erfüllt und (λ_0^o, λ_0) eine Lösung des zu (x^o, u^o) korrespondierenden adjungierten Differenzialgleichungssystems ist. Daher kann man die Transversalitätsbedingungen in einem beliebigen Zeitpunkt $t \in [t_0, t_f]$ überprüfen bzw. als notwendige Optimalitätsbedingung vorgeben. □

Bemerkung 1.3.9 (Beschränkungen an den End- und/oder Startzeitpunkt) Sei ein Optimalsteuerungsproblem mit veränderten Randbedingungen der Form $\psi(t_0, t_f, x(t_0), x(t_f)) = 0$ gegeben. Wie bereits in Abschnitt 1.2 gesehen, lassen sich nichtautonome Probleme durch Einführung eines zusätzlichen Zustands T mit

$$\dot{T} = 1, \qquad T(t_0) = t_0$$

in ein äquivalentes autonomes Optimalsteuerungsproblem umwandeln. Es sei angenommen, dass für den Endzeitpunkt eine oder mehrere solcher Randbedingungen vorliegen. Man erhält die veränderte Hamiltonfunktion

$$\begin{aligned} H^*(T, x, u, \lambda_0, \lambda, \mu) &:= H(t, x, u, \lambda_0, \lambda) + \mu \\ &= \lambda_0 f_0(T, x, u) + \lambda' f(T, x, u) + \mu \end{aligned}$$

und die Differenzialgleichung

$$\dot{\mu}(t) = -\frac{\partial H^*(T, x(t), u(t), \lambda_0, \lambda(t), \mu(t))}{\partial T}\bigg|_{T=T(t)}$$

muss an das adjungierte System angekoppelt werden. Die Orthogonalitätsbedingung (vi) ändert sich zu

$$\lambda^o(t_f^o) - \lambda_0^o \left.\frac{\partial \varphi(t_f^o, x)}{\partial x}\right|'_{x=x^o(t_f^o)} = \sum_{i=1}^n \vartheta_i \left.\frac{\partial \psi_i(t_f^o, x)}{\partial x}\right|'_{x=x^o(t_f^o)},$$

$$\mu^o(t_f^o) - \lambda_0^o \left.\frac{\partial \varphi(T, x)}{\partial T}\right|'_{T=t_f^o} = \sum_{i=1}^n \vartheta_i \left.\frac{\partial \psi_i(T, x^o(t_f^o))}{\partial T}\right|'_{T=t_f^o},$$

wobei ϑ_i ein geeigneter Skalar sei. Diese Gleichungen zeigen, dass eine Problemtransformation trotz der Beschränkungen an den Endzeitpunkt nicht notwendig ist. Stattdessen bietet es sich an, lediglich die Transversalitätsbedingung (iv) zu modifizieren:

$$\begin{aligned} H(t_f^o, x^o(t_f^o), u^o(t_f^o), \lambda_0^o, \lambda^o(t_f^o)) + \lambda_0^o \left.\frac{\partial \varphi(t, x^o(t_f^o))}{\partial t}\right|_{t=t_f^o} \\ + \sum_{i=1}^n \vartheta_i \left.\frac{\partial \psi_i(t, x^o(t_f^o))}{\partial t}\right|'_{t=t_f^o} = 0 \end{aligned}$$

Analog geht man vor, wenn solche Bedingungen für den Startzeitpunkt vorliegen. □

Nun sollen solche Probleme betrachtet werden, die unter Anderem auch Bedingungen der Form (1.4) und (1.5) enthalten können. Als Resultat werden für eine weite Klasse von Problemen notwendige Optimalitätsbedingungen dargestellt. Dabei werden die Bedingungen meist mittels Multiplikatoren an die Hamiltonfunktion angekoppelt und anschließend die notwendigen Bedingungen mittels der veränderten Hamiltonfnktion hergeleitet. Es ist jedoch stets zu bedenken, dass nicht alle Aussagen des Maximumprinzips hinsichtlich des Ankoppelns von Bedingungen an die Hamiltonfunktion bewiesen sind.[8]

Es wird jeweils die spezielle Behandlung einer Klasse von Bedingungen erörtert, sodass die Ergebnisse auch auf Probleme, die mehrere Formen dieser Bedingungen enthalten, angewendet werden können. Die folgenden Arten von Beschränkungen werden betrachtet:

(a) Zunächst werden solche Optimalsteuerungsprobleme, die Integralbedingungen für Zustands- und/oder Steuerungsvariablen beinhalten, untersucht:

$$\int_{t_0}^{t_f} w_e(t, x(t), u(t))dt = \bar{w}_e,$$
$$\int_{t_0}^{t_f} w_d(t, x(t), u(t))dt \leq \bar{w}_d.$$

(b) Die Zustands- und/oder Steuerungsvariablen müssen auf dem gesamten Intervall gewissen Gleichungsbedingungen genügen:

$$w(t, x(t), u(t)) = 0, \quad t_0 \leq t \leq t_f.$$

(c) Einige der Zustandsvariablen müssen an bestimmten Punkten t_i Gleichungsbedingungen erfüllen:

$$w^{(i)}(t_i, x_i(t_i)) = 0, \quad t_0 < t_1 < \cdots < t_i < \cdots < t_s < t_f.$$

Diese Punkte t_i können sowohl gegeben als auch unbekannt sein.

(d) Die Zustands- und/oder Steuerungsvariablen dürfen auf dem gesamten Intervall gewisse Ungleichungsbedingungen nicht verletzen:[9]

$$w(t, x(t), u(t)) \leq 0, \quad t_0 \leq t \leq t_f.$$

[8] Man vergleiche dazu [9].

[9] Diese Bedingungen entsprechen den Bedingungen (1.4) und (1.5) aus der Definition 1.2.1.

Dabei seien sowohl w_e, w_d, w und $w^{(i)}$ als auch deren partielle Ableitungen nach t und x stetige Funktionen. In den Fällen (c) und (e) werden sogar Ableitungen höherer Ordnung von w benötigt, wenn w nicht explizit von u abhängig ist. Weiterhin seien $\bar{w}_e$ und $\bar{w}_d$ gegebene Vektoren und die Notation $\alpha \leq \beta$ (α, β Vektoren) ist komponentenweise zu verstehen.

Integralbedingungen:

Es sei ein Optimalsteuerungsproblem gegeben, welches Integralbedingungen an die Zustands- und/oder Steuerungsvariablen beinhaltet. Zunächst betrachte man Integralbedingungen in Gleichungsform:

$$\int_{t_0}^{t_f} w_e(t, x(t), u(t))dt = \bar{w}_e. \tag{1.12}$$

w_e ist dabei ein r_e-dimensionaler Vektor von Funktionen, deren erste Ableitungen nach t und x stetig seien. Ferner sei $\bar{w}_e$ ein gegebener Vektor. Diese Bedingungen kann man durch Einführung eines Vektors zusätzlicher Zustandsvariablen $z \in r_e$ handhaben. Diese Variablen sind definiert durch

$$\begin{aligned} \dot{z}(t) &= w_e(t, x(t), u(t)), \\ z(t_0) &= 0, \end{aligned}$$

sodass (1.12) erfüllt ist, wenn $z(t_f) = \bar{w}_e$ gilt. Die für das resultierende Problem zu erfüllenden notwendigen Bedingungen wurden bereits vorgestellt. Analog können Optimalsteuerungsprobleme mit Integralbedingungen in Ungleichungsform

$$\int_{t_0}^{t_f} w_d(t, x(t), u(t))dt \leq \bar{w}_d. \tag{1.13}$$

transformiert werden. Gilt $z(t_f) \leq \bar{w}_d$, dann ist die Ungleichung (1.13) erfüllt.

Globale Gleichungsbedingungen:

Die folgenden Problemstellungen sind charakterisiert durch globale Gleichungsbeschränkungen an die Zustands- und/oder Steuerungsvariablen:

$$w(t, x(t), u(t)) = 0, \qquad t_0 \leq t \leq t_f,$$

wobei sowohl die vektorwertige Funktion w als auch eine geeignete Anzahl von Ableitungen nach t und x stetig seien.

Zunächst sei angenommen, dass jede Komponente von w explizit von der Steuerung u abhängig ist. Dann kann man diese Bedingungen mittels geeigneter Multiplikatoren an die Hamiltonfunktion ankoppeln

$$H^*(t, x, u, \lambda_0, \lambda, \mu) := H(t, x, u, \lambda_0, \lambda) + \mu' w(t, x, u)$$

und die vorigen Resultate auf die angepasste Hamiltonfunktion H^* anwenden.

Tritt derjenige Fall ein, dass eine oder mehrere Komponenten von w nicht explizit von der Steuerung u abhängen, kann man das Problem auf die obige

Situation zurückführen. Die Bedingung $w_i(t, x(t)) = 0,\ t_0 \leq t \leq t_f$, impliziert, dass jede totale Ableitung nach t identisch 0 sein muss. Setzt man außerdem voraus, dass jede Zustandsvariable entweder direkt oder indirekt von der Steuerung beeinflusst wird, existiert im Allgemeinen ein $q \geq 1$, für das gilt:

$$\frac{d^q w_i(t, x(t))}{dt^q} := w_i^*(t, x(t), u(t)).$$

Sei ν_i der kleinste Wert von q, sodass die q-te Ableitung explizit von u abhängt. Dann ist die Beschränkung $w_i = 0$ äquivalent zu den $\nu_i + 1$ Bedingungen

$$\begin{aligned} w_i^*(t, x(t), u(t)) &= 0, \qquad t_0 \leq t \leq t_f, \\ \left.\frac{d^j w_i(t, x(t), u(t))}{dt^j}\right|_{t=\tau} &= 0, \qquad j = 0, \ldots, \nu_i - 1, \qquad \tau \in [t_0, t_f]. \end{aligned}$$

Isolierte Gleichungsbedingungen:

Man betrachte ein Optimalsteuerungsproblem mit Gleichungsbedingungen für die Zustandsvariablen, die an gewissen Zeitpunkten erfüllt sein müssen:

$$w^{(i)}(t_i, x(t_i)) = 0, \qquad t_0 < t_1 < \cdots < t_i < \cdots < t_s < t_f,$$

wobei die Zeitpunkte $t_i,\ i = 1, \ldots, s$ sowohl vorgegeben als auch unbekannt sein können und die vektorwertige Funktion $w^{(i)}$ als auch deren Ableitungen nach t und x stetig seien.

Für den Fall, dass solche Bedingungen auftreten, können die Hamiltonfunktion und die adjungierten Funktionen λ in den Zeitpunkten t_i unstetig sein, d.h. sie weisen in diesen Zeitpunkten Sprungstellen auf. Seien x^o, u^o optimal und λ_0^o, λ^o die zugehörigen Lösungen des adjungierten Systems, sodass die notwendigen Bedingungen erfüllt sind. Dann gilt für $1 \leq i \leq s$:

$$\lim_{t \to t_i^-} \lambda^o(t) = \lim_{t \to t_i^+} \lambda^o(t) + \left.\frac{\partial w^{(i)}(t_i, x)}{\partial x}\right|_{x = x^o(t_i)}' \mu^{(i)}, \tag{1.14}$$

mit einem geeignetem Vektor $\mu^{(i)}$. Außerdem muss, sofern der Zeitpunkt t_i unbekannt ist, gelten:

$$\begin{aligned} \lim_{t \to t_i^-} H(t, x^o(t), u^o(t), \lambda_0^o, \lambda^o(t)) = \lim_{t \to t_i^+} & H(t, x^o(t), u^o(t), \lambda_0^o, \lambda^o(t)) \\ & - \left.\frac{\partial w^{(i)}(t, x^o(t_i))}{\partial t}\right|_{t=t_i}' \mu^{(i)}. \end{aligned} \tag{1.15}$$

Es hängt offensichtlich von der Funktion $w^{(i)}$ ab, ob die Hamiltonfunktion H und die adjungierten Funktionen λ unstetig sind und Sprünge aufweisen.

Globale Ungleichungsbedingungen:

Im Folgenden werden Problemstellungen mit globalen Ungleichungsbeschränkungen an die Zustands- und/oder Steuerungsvariablen untersucht. Diese beinhalten also Bedingungen der Form

$$w(t, x(t), u(t)) \leq 0, \qquad t_0 \leq t \leq t_f,$$

wobei wiederum sowohl die vektorwertige Funktion w als auch eine geeignete Anzahl von Ableitungen dieser nach t und x stetig seien.

Man gehe zunächst davon aus, dass jede der r Komponenten von w explizit von der Steuerung u abhängt. Dann ist es möglich, diese Beschränkungen durch geeignete Multiplikatoren μ an die Hamiltonfunktion anzukoppeln:

$$H^*(t, x, u, \lambda_0, \lambda, \mu) := H(t, x, u, \lambda_0, \lambda, \mu) + \mu' w(t, x, u).$$

Dabei müssen die Multiplikatoren den folgenden Bedingungen genügen:

$$\mu_i \begin{cases} \geq 0, & \text{falls } w_i(t, x, u) = 0, \\ = 0, & \text{falls } w_i(t, x, u) < 0. \end{cases}$$

Anschließend können wiederum alle früheren Resultate auf die reformulierte Hamiltonfunktion H^* angewendet werden.

Nun seien eine oder mehrere Komponenten von w nicht explizit von der Steuerung abhängig. Zunächst löst man das Optimalsteuerungsproblem ohne die Beschränkung. Verletzt die Lösung diese, so lässt sich dieser Fall wiederum auf den obigen Fall zurückführen. Da dann die Bedingung $w_i(t, x(t)) = 0$ auf einem gewissen Zeitintervall erfüllt sein muss, sind alle totalen Zeitableitungen dieser Funktion auf diesem Intervall identisch 0. Analog zur Vorgehensweise bei globalen Gleichungsbedingungen erhält man, dass die Bedingung $w_i(t, x(t)) = 0,\ t_0 \leq t_{i,j} \leq t \leq t_{i,j+1} \leq t_f$ äquivalent zu den $\nu_i + 1$ Bedingungen

$$\begin{aligned} w_i^*(t, x(t), u(t)) &\leq 0, && t_{i,j} \leq t \leq t_{i,j+1}, \\ \left.\frac{d^k w_i(t, x(t), u(t))}{dt^k}\right|_{t=t_{i,j}} &= 0, && k = 0, \ldots, \nu_i - 1, \end{aligned}$$

ist. Die erste Bedingung erlaubt den Fall, dass die Ungleichung $w_i \leq 0$ nach dem Zeitpunkt $t_{i,j+1}$ nicht mehr mit Gleichheit erfüllt ist, während die restlichen ν_i Bedingungen Unstetigkeiten der Hamiltonfunktion H und/oder der adjungierten Variablen λ nach sich ziehen können.

Singuläre Teilstücke:

Oftmals ist in Optimalsteuerungsproblemen die Hamiltonfunktion linear in der Steuerung u, sodass diese die folgende Form hat:

$$H(t, x, u, \lambda_0, \lambda) = \alpha'(t, x, \lambda_0, \lambda) u + \beta(t, x, \lambda_0, \lambda). \tag{1.16}$$

Dann kann derjenige Fall eintreten, dass ein $\lambda_0^* \geq 0$ sowie ein Paar von Funktionen (x^*, λ^*) korrespondierend zu einer gegebenen Steuerung u^* existieren, sodass eine oder mehrere Komponenten von α in einem Zeitintervall $t \in [t_1, t_2],\ t_1 < t_2$, gleich 0 sind. Dabei ist das Funktionenpaar (x^*, λ^*) eine Lösung der Gleichungen

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), u^*(t)), \\ \dot{\lambda}(t) &= -\left.\frac{\partial H(t, x, u^*(t), \lambda_0^*, \lambda(t))}{\partial x}\right|_{x=x^*(t)}'. \end{aligned}$$

In einem solchen Intervall wird die Zustandstrajektorie als singuläres Teilstück bezeichnet. Sofern auf einem Intervall $[t_1, t_2]$ ein singuläres Teilstück existiert, wird das Paar (x^*, u^*) als singuläre Lösung bezeichnet.

Eine singuläre Lösung muss die Bedingungen des folgenden Satzes (1.3.10) erfüllen, damit diese singuläre Lösung ein Teil einer Lösung eines Optimalsteuerungsproblems sein kann. Dieser Satz stellt gewisse Anforderungen an die Differenzierbarkeit der Funktionen α und β. Ferner bezeichne $\alpha^{(i)}(t, x(t), \lambda_0, \lambda(t), u(t)),\ i = 0, 1, 2, \ldots,$ die i-te totale Zeitableitung von α:

$$\begin{aligned}\alpha^{(1)}(t, x(t), \lambda_0, \lambda(t), u) = {} & \left.\frac{\partial \alpha(t, x, \lambda_0, \lambda)}{\partial t}\right|_{x=x(t), \lambda=\lambda(t)} \\ & + \left.\frac{\partial \alpha(t, x, \lambda_0, \lambda(t))}{\partial x}\right|_{x=x(t)} f(t, x(t), u) \\ & - \left.\frac{\partial \alpha(t, x(t), \lambda_0, \lambda)}{\partial \lambda}\right|_{\lambda=\lambda(t)} \left.\frac{\partial H(t, x, u, \lambda_0, \lambda(t))}{\partial x}\right|'_{x=x(t)}.\end{aligned}$$

Satz 1.3.10 *Seien die zu einem gegebenen Problem gehörige Hamiltonfunktion von der Form (1.16) und $(u^o(\cdot), x^o(\cdot))$ ein optimales Paar. Außerdem existiere eine Lösung $\lambda_0^o, \lambda^o(\cdot)$ des adjungierten Differenzialgleichungssystems. Eine notwendige Bedingung dafür, dass das Paar $(u^o(\cdot), x^o(\cdot))$ auf dem Intervall $[t_1, t_2]$ eine singuläre Lösung enthält, ist:*

$$\text{(i)} \qquad \frac{d^k \alpha(t, x^o(t), \lambda_0^o, \lambda^o(t))}{dt^k} = 0, \quad k = 0, 1, 2, \ldots,\ t \in [t_1, t_2],$$

$$\text{(ii)} \qquad \frac{\partial \alpha^{(l)}(t, x^o(t), \lambda_0^o, \lambda^o(t), u)}{\partial u} = 0, \quad l = 1, 3, 5, \ldots,\ t \in [t_1, t_2],$$

$$\text{(iii)} (-1)^m \frac{\partial \alpha^{(2m)}(t, x^o(t), \lambda_0^o, \lambda^o(t), u)}{\partial u} \geq 0, \quad m = 1, 2, 3, \ldots,\ t \in [t_1, t_2].$$

Sei $\bar{m}$ die kleinste Zahl m, für die die Bedingung (iii) mit echter Ungleichheit erfüllt ist. Dann kann der Test der obigen Bedingungen bei $\bar{k} = 2\bar{m}$, $\bar{l} = 2\bar{m} - 1$, $\bar{m}$ abgebrochen werden.

Kapitel 2

Implementierung der Mehrzielmethode[1]

Eine einfache Methode zur Lösung von Randwertproblemen ist die sogenannte Mehrzielmethode[2]. Im folgenden Abschnitt wird zunächst das Einfachschießverfahren sowie anschließend das Mehrfachschießverfahren vorgestellt. Da die Anwendung des Maximumprinzips auch auf sogenannte Mehrpunktrandwertprobleme führen kann, wird die Mehrzielmethode dahingehend erweitert, dass auch Randwertprobleme mit Schaltpunkten und -bedingungen, unbekannten Konstanten sowie Sprüngen der adjungierten Variablen gelöst werden können.

Man betrachte ein allgemeines Randwertproblem für n gesuchte Funktionen $y_i(x),\ i = 1, \ldots, n$

$$y' = f(x, y), \quad r(y(a), y(b)) = 0, \quad y = (y_1, \ldots, y_n)^T, \tag{2.1}$$

wobei $f(x, y)$ und $r(u, v)$ Vektoren von n Funktionen sind. Nun versucht man, einen Startvektor $s \in \mathbb{R}^n$ für das Anfangswertproblem

$$y' = f(x, y), \quad y(a) = s, \tag{2.2}$$

zu finden, sodass die Lösung $y(x) = y(x, s)$ die Randbedingungen

$$r(y(a, s), y(b, s)) \equiv r(s, y(b, s)) = 0$$

erfüllt. Man sucht also eine Nullstelle $s = (\sigma_1, \ldots, \sigma_n)^T$ der Gleichung

$$F(s) = r(s, y(b, s)).$$

Für jedes s lässt sich $F(s)$ durch Lösen des Anfangswertproblems (2.2) bestimmen. Im Allgemeinen verwendet man für die Berechnungen einer neuen Näherungslösung $s^{(i+1)}$ das allgemeine Newton-Verfahren

$$s^{(i+1)} = s^{(i)} - DF(s^{(i)})^{-1} F(s^{(i)}).$$

[1] Vergleiche zum einfachen Schießverfahren als auch zur Mehrzielmethode bspw. [18], [16] und [11].

[2] Auch als Mehrfachschießverfahren oder Multiple Shooting Method bezeichnet.

In jedem Iterationsschritt sind also $F(s^{(i)})$, die Funktionalmatrix

$$DF(s^{(i)}) = \left.\frac{\partial F_j}{\partial \sigma_k}\right|_{s=s^{(i)}}$$

sowie die Lösung $d^{(i)} := s^{(i)} - s^{(i+1)}$ des linearen Gleichungssystems

$$DF(s^{(i)})d^{(i)} = F(s^{(i)})$$

zu berechnen.[3]

Beim einfachen Schießverfahren können allerdings einige Schwierigkeiten auftreten:

(a) Die Lösung von $y' = f(x, y),\ y(a) = s$ kann sehr empfindlich von s abhängen. Für $b \gg a$ kann eine Ungenauigkeit im Anfangswert erhebliche Abweichungen in $y(b, s)$ bedeuten.

(b) Meist besitzt f zwar partielle Ableitungen nach y, aber $\|D_y f(x, y)\|$ ist auf $[a, b] \times \mathbb{R}^n$ unbeschränkt. Dies bedeutet, dass die Lösung des Anfangswertproblems $y' = f(x, y),\ y(x_0) = s$ nur noch in einer gewissen Umgebung von x_0, deren Größe von s abhängt, definiert ist. Damit existieren unter Umständen $y(a, x_0, s)$ und $y(b, x_0, s)$ nur für die Werte s aus einer kleinen Menge M. Wählt man nun ein $s^{(0)}$, von dem man in der Regel nicht weiß, ob dieses in der Menge M liegt, so bricht das einfache Schießverfahren zusammen.

Um diese Probleme zu umgehen, verwendet man stattdessen die Mehrzielmethode, deren Idee darin besteht, das Intervall $[a, b]$ in $(m - 1)$ Teilintervalle zu zerlegen und daraufhin das einfache Schießverfahren auf alle Teilintervalle anzuwenden. Es sei also eine Zerlegung $a = x_1 < \cdots < x_m = b$ sowie Startwerte $s_1, \ldots, s_m,\ s_i \in \mathbb{R}^n,\ i = 1, \ldots, m$ gegeben. Anschließend löst man die Anfangswertprobleme

$$y' = f(x, y), \quad y(x_k) = s_k, \quad k = 1, \ldots, m - 1, \tag{2.3}$$

und bestimmt die $s_k,\ k = 1, \ldots, m - 1$ so, dass die aus den Lösungen $y(x, x_k, s_k),\ k = 1, \ldots, m - 1$ der Anfangswertprobleme zusammengesetzte Funktion

$$y(x, s) = \begin{cases} y(x, x_i, s_i), & \text{für } x \in [x_i, x_{i+1}] \\ s_m, & \text{für } x = b \end{cases}$$

stetig ist. Dies führt auf die folgenden $n \times m$ Bedingungen

$$\begin{aligned} y(x_{i+1}, x_i, s_i) &= s_{i+1}, \quad i = 1, \ldots, m - 1, \\ r(s_1, s_m) &= 0. \end{aligned}$$

[3] Auf die Berechnung der Funktionalmatrix wird bei der Vorstellung der Mehrzielmethode und auf die Lösung des linearen Gleichungssystems in 2.3 genauer eingegangen.

Man erhält also ein nichtlineares Gleichungssystem für $s = \left(s_1^T, \ldots, s_m^T\right)^T$ der Form

$$F(s) := \begin{pmatrix} F_1(s_1, s_2) \\ F_2(s_2, s_3) \\ \vdots \\ F_{m-1}(s_{m-1}, s_m) \\ F_m(s_1, s_m) \end{pmatrix} := \begin{pmatrix} y(x_2, x_1, s_1) - s_2 \\ y(x_3, x_2, s_2) - s_3 \\ \vdots \\ y(x_m, x_{m-1}, s_{m-1}) - s_m \\ r(s_1, s_m) \end{pmatrix} = 0. \tag{2.4}$$

Dieses Gleichungssystem kann analog zum einfachen Schießverfahren wieder mit dem Newtonverfahren

$$s^{(i+1)} = s^{(i)} - DF(s^{(i)})^{-1} F(s^{(i)}) \tag{2.5}$$

iterativ gelöst werden. Dazu müssen wiederum in jedem Iterationsschritt $F(s^{(i)})$, $DF(s^{(i)})$ sowie die Lösung $d^{(i)} := s^{(i)} - s^{(i+1)}$ des Gleichungssystems $DF(s^{(i)})d^{(i)} = F(s^{(i)})$ bestimmt werden. Den Vektor $F(s)$ erhält man durch Bestimmung der Werte $y(x_{k+1}, x_k, s_k)$, $k = 1, \ldots, m-1$, welche die Lösungen der $(m-1)$ Anfangswertprobleme (2.3) sind. Die Funktionalmatrix $DF(s) = D_{s_k} F_i(s)|_{i,k=1,\ldots,m}$ hat wegen der Struktur der F_i (2.4) die Gestalt

$$DF(s) = \begin{pmatrix} G_1 & -I & 0 & \cdots & 0 \\ 0 & G_2 & -I & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & G_{m-1} & -I \\ A & 0 & \cdots & 0 & B \end{pmatrix}$$

mit den $n \times n$ Funktionalmatrizen $A, B, G_k,\ k = 1, \ldots, m-1$

$$\begin{aligned} G_k &:= D_{s_k} F_k(s_k, s_{k+1}) = D_{s_k} y(x_{k+1}, x_k, s_k), \quad k = 1, \ldots, m-1, \\ A &:= D_{s_1} F_m(s_1, s_m) = D_{s_1} r(s_1, s_m), \\ B &:= D_{s_m} F_m(s_1, s_m) = D_{s_m} r(s_1, s_m). \end{aligned}$$

Dies reicht allerdings nicht aus, um aus dem Maximumprinzip folgende Randwertprobleme zu lösen, da diese weitere Elemente enthalten können. Dies sind im Einzelnen:

- Schaltpunkte $sp_k,\ k = 1, \ldots, \alpha$
- Schaltbedingungen $sb_k,\ k = 1, \ldots, \beta$
- Sprünge $ju_k,\ k = 1, \ldots, \gamma$
- Konstanten $c_k,\ k = 1, \ldots, \delta$.

Sobald ein Optimalsteuerungsproblem Schaltpunkte enthält, können Schaltbedingungen, die analog zu den Anfangs- und Endbedingungen mit Gleichheit erfüllt sein müssen, auftreten. Die adjungierten Variablen können in

den Schaltpunkten Unstetigkeiten, die sogenannten Sprünge, aufweisen. Diese Sprünge implizieren keine zusätzlichen Bedingungen, sondern müssen lediglich am entsprechenden Schaltpunkt ausgeführt werden. Allerdings ist im Allgemeinen die Größe des Sprunges unbekannt, sodass diese als konstante Größe in das zu lösende Randwertproblem integriert und die Größe des Sprungs bestimmt werden muss.

Nun sollen die daraus resultierenden Änderungen für die zu erfüllenden Bedingungen untersucht werden. Bisher enthielt der Vektor $F(s)$ $n \times m$ Bedingungen und zwar im Einzelnen $n \times (m-1)$ Stetigkeitsbedingungen an den Intervallgrenzen sowie m Randbedingungen. Nun enthält das Problem aber zusätzlich β Schaltbedingungen, welche den Randbedingungen hinzugefügt werden. Analog zum Mehrfachschießverfahren sei $y(x,s)$ die aus den Lösungen $y(x, x_k, s_k, sp, c),\ k = 1, \ldots, m-1$ der Anfangswertprobleme

$$y' = f(x, y, c), \quad y(x_k) = s_k, \quad k = 1, \ldots, m-1$$

zusammengesetzte Funktion. Man beachte, dass bei der Lösung der Anfangswertprobleme nun Konstanten $c_i,\ i = 1, \ldots, \delta$ und Schaltpunkte $sp_i,\ i = 1, \ldots, \alpha$ zu berücksichtigen sind. Enthält das zu lösende Anfangswertproblem einen Schaltpunkt, müssen die Steuerung an diesem umgeschaltet und eventuelle Sprungbedingungen ausgeführt werden. Für den Vektor der Rand- und Schaltbedingungen ergibt sich:

$$\begin{aligned} r(y(a), y(b), &y(sp_1), \ldots, y(sp_\alpha), sp, c) \\ &= r(y(a,s), y(b,s), y(sp_1, s), \ldots, y(sp_\alpha, s), sp, c) \\ &= r(s, sp, c) \\ &=: F_m(s, sp, c). \end{aligned}$$

Da dem System α Schaltpunkte und δ Konstanten hinzugefügt wurden, benötigt man in der Regel $n + \alpha + \delta$ Bedingungen[4], um das System eindeutig lösen zu können. Man hat folglich ein nichtlineares Gleichungssystem für $s = (s_1, \ldots, s_m)^T$, $sp \in \mathbb{R}^\alpha$ und $c \in \mathbb{R}^\delta$ der Form

$$\begin{aligned} F(s, sp, c) &:= \begin{pmatrix} F_1(s_1, s_2, sp, c) \\ F_2(s_2, s_3, sp, c) \\ \vdots \\ F_{m-1}(s_{m-1}, s_m, sp, c) \\ F_m(s, sp, c) \end{pmatrix} \\ &:= \begin{pmatrix} y(x_2, x_1, s_1, sp, c) - s_2 \\ y(x_3, x_2, s_2, sp, c) - s_3 \\ \vdots \\ y(x_m, x_{m-1}, s_{m-1}, sp, c) - s_m \\ r(s, sp, c) \end{pmatrix} = 0 \end{aligned} \tag{2.6}$$

mit $F(s, sp, c) \in \mathbb{R}^{m \cdot n + \alpha + \delta}$ zu lösen.

[4]Statt lediglich n Randbedingungen.

Bei der Anpassung der Mehrzielmethode, sodass auch Mehrpunktrandwertprobleme gelöst werden können, verändern sich ebenfalls die Gestalt sowie die Dimension der Funktionalmatrix DF:

$$DF(s, sp, c) = \begin{pmatrix} G_1 & -I & 0 & \cdots & 0 & H_1 & N_1 \\ 0 & G_2 & -I & \ddots & \vdots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & \vdots \\ 0 & \cdots & 0 & G_{m-1} & -I & H_{m-1} & N_{m-1} \\ L_1 & \cdots & \cdots & \cdots & L_m & M & P \end{pmatrix}.$$

Dabei ist $DF(s, sp, c)$ eine $(n \cdot m + \alpha + \delta)$-dimensionale quadratische Funktionalmatrix mit den Funktionalmatrizen

$$\begin{aligned}
G_k &:= D_{s_k} F_k(s_k, s_{k+1}, sp, c) = D_{s_k} y(x_{k+1}, x_k, s_k, sp, c), \quad k = 1, \ldots, m-1, \\
L_k &:= D_{s_k} F_m(s, sp, c) = D_{s_k} r(s, sp, c), \quad k = 1, \ldots, m, \\
H_k &:= D_{sp} F_k(s_k, s_{k+1}, sp, c) = D_{sp} y(x_{k+1}, x_k, s_k, sp, c), \quad k = 1, \ldots, m-1, \\
M &:= D_{sp} F_m(s, sp, c) = D_{sp} r(s, sp, c), \\
N_k &:= D_c F_k(s_k, s_{k+1}, sp, c) = D_c y(x_{k+1}, x_k, s_k, sp, c), \quad k = 1, \ldots, m-1, \\
P &:= D_c F_m(s, sp, c) = D_c r(s, sp, c),
\end{aligned}$$

die die folgenden Dimensionen besitzen

$$\begin{aligned}
G_k &\in Mat(n, n), \\
L_k &\in Mat(n + \alpha + \delta, n), \\
H_k &\in Mat(n, \alpha), \\
M &\in Mat(n + \alpha + \delta, \alpha), \\
N_k &\in Mat(n, \delta), \\
P &\in Mat(n + \alpha + \delta, \delta).
\end{aligned}$$

Es sind demnach in jeder Iteration der Vektor $F(s^{(i)}, sp^{(i)}, c^{(i)})$, die Funktionalmatrix $DF(s^{(i)}, sp^{(i)}, c^{(i)})$ sowie die Lösung

$$d^{(i)} = \begin{pmatrix} s^{(i)} - s^{(i+1)} \\ sp^{(i)} - sp^{(i+1)} \\ c^{(i)} - c^{(i+1)} \end{pmatrix}$$

des Gleichungssystems

$$DF(s^{(i)}, sp^{(i)}, c^{(i)}) \cdot d^{(i)} = F(s^{(i)}, sp^{(i)}, c^{(i)})$$

zu bestimmen. Dabei müssen für $F(s^{(i)}, sp^{(i)}, c^{(i)})$ die $m-1$ Anfangswertprobleme

$$y' = f(x, y, sp, c), \quad y(x_k) = s_k, \quad k = 1, \ldots, m-1,$$

mit den Lösungen $y(x_{k+1}, x_k, s_k, sp, c)$, $k = 1, \ldots, m-1$, gelöst, sowie die Rand- und Schaltbedingungen ausgewertet werden. Auf die Bestimmung der Ableitungen und damit die Berechnung der Matrix $DF(s^{(i)}, sp^{(i)}, c^{(i)})$ wird im Abschnitt 2.1.1 genauer eingegangen. Der Abschnitt 2.2 behandelt die verschiedenen zur Auswahl stehenden Anfangswertproblemlöser, während die Lösung des in jeder Iteration zu lösenden Gleichungssystems, wie bereits erwähnt, in Abschnitt 2.3 betrachtet wird.

2.1 Das Newton-Verfahren

2.1.1 Bestimmung der Ableitungen

Um die Ableitungen für die Funktionalmatrix $DF(s, sp, c)$ zu bestimmen, werden die einzelnen Funktionalmatrizen durch Differenzenquotienten ersetzt. Am Beispiel der Funktionalmatrizen $G_k(s_k, s_{k+1}, sp, c)$ führt die Bildung des Differenzenquotienten auf

$$\begin{aligned} G_k(s_k, s_{k+1}, sp, c) &:= D_{s_k} F_k(s_k, s_{k+1}, sp, c) \approx \Delta F_k(s_k, s_{k+1}, sp, c) \\ &= (\Delta_1 F_k(s_k, s_{k+1}, sp, c), \ldots, \Delta_n F_k(s_k, s_{k+1}, sp, c)), \\ &\quad k = 1, \ldots, m-1, \end{aligned}$$

mit

$$\Delta_j F_k(s_k, s_{k+1}, sp, c) = \frac{F_k(s_k + \Delta\sigma_{kj} \cdot e_j, s_{k+1}, sp, c) - F_k(s_k, s_{k+1}, sp, c)}{\Delta\sigma_{kj}},$$

wobei σ_{kj} die j-te Komponente von s_k und $e_j \in \mathbb{R}^n$ der j-te Einheitsvektor seien. Für jede Matrix $G_k(s_k, s_{k+1}, sp, c)$ müssen dazu weitere n Anfangswertprobleme

$$y' = f(x, y, sp, c), \quad y(x_k) = (s_k + \Delta\sigma_{kj} \cdot e_j), \quad j = 1, \ldots, n,$$

gelöst werden, um die Werte $F_k(s_k + \Delta\sigma_{kj} \cdot e_j, s_{k+1}, sp, c)$ zu erhalten, während die Werte $F_k(s_k, s_{k+1}, sp, c)$ bereits aus der Berechnung von $F(s, sp, c)$ bekannt sind. Man hat also in jedem Iterationsschritt zu den n bereits gelösten weitere $n \cdot (m-1)$ Anfangswertprobleme zu lösen.

Für die Berechnung der weiteren Funktionalmatrizen werden die folgenden Differenzenquotienten verwendet:

$\underline{L_k,\ k = 1, \ldots, m}$

Für die Berechnung der L_k ist es nicht nötig, weitere Anfangswertprobleme zu lösen. Die benötigten Lösungen der Anfangswertprobleme mit gestörten Startwerten fallen bei der Bestimmung der Matrizen G_k ab. Diese Lösungen werden bei der Berechnung der Matrizen $G_k,\ k = 1, \ldots, m-1$ gespeichert und können nun in die Rand- und Schaltbedingungen eingesetzt werden. Die Differenzenquotienten werden durch Auswertung der Rand- und Schaltbedingungen bestimmt.

$$\begin{aligned} L_k &= D_{s_k} F_m(s, sp, c) \approx \Delta F_m(s, sp, c) \\ &= (\Delta_1 F_m(s, sp, c), \ldots, \Delta_n F_m(s, sp, c)), \\ &\quad k = 1, \ldots, m, \end{aligned}$$

mit

$$\Delta_j F_m(s, sp, c) = \\ \frac{F_m(s_1, \ldots, s_{k-1}, s_k + \Delta\sigma_{kj} \cdot e_j, s_{k+1}, \ldots, s_m, sp, c) - F_m(s, sp, c)}{\Delta\sigma_{kj}},$$

wobei σ_{kj} die j-te Komponente von s_k und $e_j \in \mathbb{R}^n$ der j-te Einheitsvektor seien.

$H_k,\ k = 1, \ldots, m-1$, und M

Die Matrix $H_k,\ k \in \{1, \ldots, m\}$ beschreibt den Einfluss einer Änderung der Schaltpunkte auf den Endwert $y(x_{k+1})$ des k-ten Zeitintervalls $[x_k,\ x_{k+1}]$. Sofern der j-te Schaltpunkt $sp_j,\ j \in \{1, \ldots, \alpha\}$ nicht im k-ten Teilintervall liegt, ist die j-te Spalte der Matrix H_k identisch Null. Demzufolge ist die Matrix H_k genau dann die Nullmatrix, wenn das k-te Teilintervall überhaupt keinen Schaltpunkt enthält. Man hat somit genau die α Anfangswertprobleme zu lösen, bei denen im k-ten Teilintervall für den oder die Schaltpunkte sp_j gilt: $sp_j \in [x_k, x_{k+1}]$. Für die Bestimmung der Funktionalmatrix M kann man analog zu den vorhergehenden Berechnungen erneut die bereits ermittelten Lösungen aus der Berechnung der Matrizen H_k verwenden. Die Matrix M beschreibt den Einfluss einer Änderung der Schaltpunkte auf die Rand- und Schaltbedingungen. Im Folgenden sei $e_j \in \mathbb{R}^\alpha$ der j-te Einheitsvektor.

$$\begin{aligned} H_k &= D_{sp}F_k(s_k, s_{k+1}, sp, c) \approx \Delta F_k(s_k, s_{k+1}, sp, c) \\ &= (\Delta_1 F_k(s_k, s_{k+1}, sp, c), \ldots, \Delta_\alpha F_k(s_k, s_{k+1}, sp, c)), \\ &\quad k = 1, \ldots, m-1, \end{aligned}$$

mit

$$\Delta_j F_k(s_k, s_{k+1}, sp, c) = \frac{F_k(s_k, s_{k+1}, sp + \Delta sp_j \cdot e_j, c) - F_k(s_k, s_{k+1}, sp, c)}{\Delta sp_j},$$

sowie

$$\begin{aligned} M &= D_{sp}F_m(s, sp, c) \approx \Delta F_m(s, sp, c) \\ &= (\Delta_1 F_m(s, sp, c), \ldots, \Delta_\alpha F_m(s, sp, c)) \end{aligned}$$

mit

$$\Delta_j F_m(s, sp, c) = \frac{F_m(s, sp + \Delta sp_j \cdot e_j, c) - F_m(s, sp, c)}{\Delta sp_j}.$$

$N_k,\ k = 1, \ldots, m-1$, und P

Bei der Bestimmung der Differenzenquotienten für die Funktionalmatrizen $N_k,\ k = 1, \ldots, m-1$ und P ist zwischen zwei Arten von Konstanten zu unterscheiden:

1. Sprungkonstante $c_j,\ j \in \{1, \ldots, \delta\}$
   In diesem Fall taucht die Konstante lediglich in einer Sprungbedingung, die am Schaltpunkt $sp_i,\ i \in \{1, \ldots, \alpha\}$ ausgeführt werden muss, auf. Dies bedeutet, dass für diese Konstante in demjenigen Teilintervall k ein Anfangswertproblem gelöst werden muss, das den entsprechenden Schaltpunkt beinhaltet. Die Einträge der j-ten Spalten der Matrizen $N_l,\ l = 1, \ldots, m-1,\ l \neq k$ sind dementsprechend ebenso identisch Null wie die j-te Spalte der Matrix P.

2. "Konstante Differenzialgleichungen"
   Diese Konstanten werden eingeführt, wenn die rechte Seite einer oder mehrerer Differenzialgleichungen identisch Null und damit die Lösung dieser eine Konstante ist. Dies hat, wie bereits erwähnt, den Vorteil, dass die Dimension der zu lösenden Differenzialgleichungssysteme und damit auch die Rechenzeit verringert wird. Diese Konstante $c_j,\ j \in \{1, \ldots, \delta\}$ tritt nun, sofern sie nicht unabhängig vom übrigen System ist, bei mindestens einer der anderen Differenzialgleichungen in der rechten Seite oder aber in der Steuerung auf. Hier muss in jedem Teilintervall $k,\ k = 1, \ldots, m-1$ ein Anfangswertproblem gelöst werden. Die bestimmten Lösungen können auch hier zur Berechnung der j-ten Spalte der Matrix P verwendet werden.

Diese Unterscheidung wird vom Programm vorgenommen, damit gegebenenfalls die Anzahl der zu lösenden Anfangswertprobleme reduzieren werden kann, um somit Rechenzeit einzusparen. Im günstigsten Fall[5] sind folglich lediglich δ zusätzliche Anfangswertprobleme zu lösen, währenddessen es im schlechtesten Fall[6] $\delta \cdot (m-1)$ sind. Im Folgenden sei $e_j \in \mathbb{R}^\delta$ der j-te Einheitsvektor.

$$\begin{aligned} N_k &= D_c F_k(s_k, s_{k+1}, sp, c) \approx \Delta F_k(s_k, s_{k+1}, sp, c) \\ &= (\Delta_1 F_k(s_k, s_{k+1}, sp, c), \ldots, \Delta_\delta F_k(s_k, s_{k+1}, sp, c)), \\ &\quad k = 1, \ldots, m-1, \end{aligned}$$

mit

$$\Delta_j F_k(s_k, s_{k+1}, sp, c) = \frac{F_k(s_k, s_{k+1}, sp, c + \Delta c_j \cdot e_j) - F_k(s_k, s_{k+1}, sp, c)}{\Delta c_j},$$

sowie

$$\begin{aligned} P &= D_c F_m(s, sp, c) \approx \Delta F_m(s, sp, c) \\ &= (\Delta_1 F_m(s, sp, c), \ldots, \Delta_\delta F_m(s, sp, c)) \end{aligned}$$

mit

$$\Delta_j F_m(s, sp, c) = \frac{F_m(s, sp, c + \Delta c_j \cdot e_j) - F_m(s, sp, c)}{\Delta c_j}.$$

Folglich sind insgesamt in jedem Iterationsschritt des Newtonverfahrens für die Berechnung von $F(s, sp, c)$ und $DF(s, sp, c)$ mindestens $n \cdot m + \alpha + \delta$ und maximal $n \cdot m + \alpha + \delta \cdot (m-1)$ Anfangswertprobleme zu lösen.

Bei der Bestimmung der Ableitung $DF(\eta^{(i)})$ mit $\eta := (s, sp, c)^T$ durch einen Differenzenquotienten $\Delta F(\eta^{(i)})$

$$\Delta F(\eta^{(i)}) := \frac{F(\eta^{(i)} + \Delta\eta^{(i)}) - F(\eta^{(i)})}{\Delta\eta^{(i)}}$$

[5] Alle Konstanten gehören zu Fall 1.
[6] Alle Konstanten gehören zu Fall 2.

ist $\Delta\eta^{(i)}$ "genügend klein" zu wählen.[7] Wird $\Delta\eta^{(i)}$ zu groß gewählt, ist $\Delta F(\eta^{(i)})$ nur eine schlechte Approximation für $DF(\eta^{(i)})$ und das Verfahren konvergiert dann wesentlich schlechter oder überhaupt nicht. Wählt man dagegen $\Delta\eta^{(i)}$ zu klein, tritt bei der Subtraktion $F(\eta^{(i)} + \Delta\eta^{(i)}) - F(\eta^{(i)})$ Auslöschung auf, da $F(\eta^{(i)} + \Delta\eta^{(i)}) \approx F(\eta^{(i)})$ gilt. Dies hat zur Folge, dass selbst kleine Fehler bei der Berechnung von $F(\eta^{(i)})$ und $F(\eta^{(i)} + \Delta\eta^{(i)})$ große Auswirkungen auf den Differenzenquotienten $\Delta F(\eta^{(i)})$ haben und diesen somit stark verfälschen. Daher ist die Lösung der Anfangswertprobleme möglichst genau vorzunehmen.[8] Wird $\Delta\eta^{(i)}$ so gewählt, dass $F(\eta^{(i)})$ und $F(\eta^{(i)} + \Delta\eta^{(i)})$ bei t-stelliger Rechnung die ersten $t/2$ Stellen gemeinsam haben, sind die Auswirkungen der Auslöschung noch erträglich. Dies ist im Allgemeinen bei der Wahl

$$\Delta\eta^{(i)} = \sqrt{eps} \cdot \eta^{(i)}$$

der Fall, wobei eps die Maschinengenauigkeit ist. Bei der Programmierung in C++ mit doppelter Genauigkeit gilt dabei $eps \approx 2.220446049 \cdot 10^{-16}$, also $\sqrt{eps} \approx 1.490116119 \cdot 10^{-8}$. Damit $\Delta\eta^{(i)}$ nicht zu klein gewählt wird, führt man eine untere Schranke $\Delta\eta^{(i)} \geq \sqrt{eps}$ ein. Insgesamt wird $\Delta\eta^{(i)}$ also nach folgender Vorschrift bestimmt

$$\Delta\eta^{(i)} = \begin{cases} \sqrt{eps} \cdot \eta^{(i)}, & \text{falls } \left|\eta^{(i)}\right| \geq 1 \\ \sqrt{eps}, & \text{sonst} \end{cases} .$$

2.1.2 Dämpfung des Newton-Verfahrens

Da die in einer Iteration des Newtonverfahrens durch das Lösen des linearen Gleichungssystems bestimmte Änderung der Lösung $d^{(i)}$[9] zu groß ausfallen können, wird ein sogenannter Relaxationsfaktor $\lambda \in (0, 1]$ bestimmt und anschließend auf der Grundlage von verschiedenen Testfunktionen entschieden, ob dieser Faktor akzeptiert wird oder gegebenenfalls verkleinert werden muss. Diese Vorgehensweise wurde im vorliegenden Programm größtenteils aus dem Fortran-Programm BNDSCO[10] übernommen.

Der Relaxationsfaktor $\lambda^{(i)}$ berechnet sich in der i-ten Iteration folgendermaßen

$$\mu^{(i)} := \frac{\|d^{(i)}\|_2}{\|d^{(i)} - \hat{d}^{(i)}\|_2} \lambda^{(i-1)} \text{ mit } \hat{d}^{(i)} := (DF^{(i-1)})^{-1} F^{(i)},$$

$$\lambda^{(i)} := \begin{cases} 1, & \text{falls } \mu^{(i)} \geq 0.7 \\ \mu^{(i)}, & \text{sonst} \end{cases} .$$

[7] Die Ausführungen zur Wahl von $\Delta\eta^{(i)}$ stammen aus [18] S.192 und wurden durch eine untere Schranke erweitert.
[8] Vergleiche dazu Abschnitt 2.2.
[9] Im folgenden als Newtonschritt bezeichnet.
[10] Siehe [11] S.30f.

Die Formel für den gedämpften Newtonschritt lautet wie folgt:

$$\eta^{(i+1)} - \eta^{(i)} = \lambda^{(i)} \cdot d^{(i)} = \lambda^{(i)} \cdot (DF^{(i)})^{-1} F^{(i)},$$

wobei η_i die Lösung, $DF^{(i)}$ die Jacobimatrix und $F^{(i)}$ die rechte Seite des zu lösenden nichtlinearen Gleichungssystems jeweils in der i-ten Iteration des Newtonverfahrens sind.

Nach der Bestimmung des Relaxationsfaktors werden die Größen

$$\begin{aligned} \eta^{(i+1)} &= \eta^{(i)} + \lambda_i d^{(i)}, \\ \hat{d}^{(i+1)} &= (DF^{(i)})^{-1} F^{(i+1)} \end{aligned}$$

berechnet und die "Güte"von λ_i anhand der drei folgenden Testfunktionen beurteilt:

$$\begin{aligned} T_1(F(\eta), d) &:= \|F(\eta)\|_2^2, \\ T_2(F(\eta), d) &:= \|d_1\|_2^2 + \sum_{j=1}^{m-1} \|F_j(\eta)\|_2^2, \\ T_3(F(\eta), d) &:= \|d\|_2^2. \end{aligned}$$

Dabei sind d_1 die ersten n Komponenten des Vektors d. Der Dämpfungsfaktor wird nur dann akzeptiert, wenn für mindestens eine der Testfunktionen die Ungleichung

$$T_i(F(\eta^{(i+1)}), \hat{d}^{(i+1)}) \leq T_i(F(\eta^{(i)}), d^{(i)}), \quad i \in \{1, 2, 3\},$$

erfüllt ist. Anderenfalls wird so lange λ_i halbiert, bis mindestens eine der Ungleichungen erfüllt ist. Dabei darf λ_i aber nicht kleiner als 0.01 werden, um sicherzustellen, dass der Newtonschritt nicht beliebig klein werden kann.

Bemerkung 2.1.1 Unterschreitet der Relaxationsfaktor λ_i die vorgegebene untere Schranke 0.01, ohne dass für mindestens eine der drei Testfunktionen die Ungleichung erfüllt ist, so wird eine Warnmeldung ausgegeben, die Mehrzielmethode jedoch mit dem nicht akzeptierten Schritt fortgesetzt. □

2.1.3 Die Broydenapproximation[11]

Wie bereits erwähnt wurde, ist die Berechnung der Funktionalmatrix DF aufgrund der zu lösenden Anfangswertprobleme sehr aufwendig. Um den benötigten Rechenaufwand zu senken, gibt es verschiedene Methoden, eine Approximation der Funktionalmatrix DF zu bestimmen. Eine Möglichkeit besteht darin, die Matrix über die sogenannten Rang-1-Updates[12] von Broyden zu approximieren.

[11] Die Ausführungen zur Broydenapproximation orientieren sich an [2] S.58ff sowie S.81ff und [13] Kapitel 9. Für eine ausführlichere Darstellung vergleiche man vor allem [2].

[12] Diese Update-Formeln werden Rang-1-Updates genannt, da für eine Matrix A und die mittels einer Updateformel approximierte Matrix A' folgendes gilt: $rang(A' - A) \leq 1$.

Man betrachtet das Newtonverfahren

$$\begin{aligned} DF(x^{(k)})\delta x^{(k)} &= -F(x^{(k)}), \\ x^{(k+1)} &= x^{(k)} + \lambda \delta x^{(k)}. \end{aligned}$$

Im Folgenden bezeichne $DF^{(k)} = DF(x^{(k)})$ die Jacobimatrix bzw. eine Näherung dieser und $F^{(k)} = F(x^{(k)})$ die rechte Seite des zu lösenden linearen Gleichungssystems in der k-ten Iteration des Newtonverfahrens. Die Jacobimatrix $DF^{(k+1)}$ soll nun approximativ bestimmt werden. Aufgrund der Taylorentwicklung von F im Entwicklungspunkt $x^{(k+1)}$ fordert man, dass die sogenannte Quasi-Newton-Bedingung (2.7) erfüllt sei:

$$DF^{(k+1)}\delta x^{(k)} = F^{(k+1)} - F^{(k)}. \tag{2.7}$$

Bis auf $DF^{(k+1)}$ sind alle Daten bekannt. Weiterhin verlangt man, dass sich $DF^{(k+1)}$ und $DF^{(k)}$ in Richtungen senkrecht zu $\delta x^{(k)}$ nicht unterscheiden:

$$(DF^{(k+1)} - DF^{(k)})v = 0, \quad \forall v \perp \delta x^{(k)}.$$

Somit muss $DF^{(k+1)}$ die folgende Darstellung besitzen:

$$DF^{(k+1)} = DF^{(k)} + \frac{u(\delta x^{(k)})^T}{\|\delta x^{(k)}\|_2^2}.$$

Aus der Quasi-Newton-Bedingung (2.7)

$$DF^{(k+1)}\delta x^{(k)} = DF^{(k)}\delta x^{(k)} + u = F^{(k+1)} - F^{(k)}$$

folgt dann:

$$u = F^{(k+1)} - F^{(k)} - DF^{(k)}\delta x^{(k)}.$$

Damit erhält man die Updateformel für das „gute" Broyden-Update:

$$DF^{(k+1)} = DF^{(k)} + \frac{(F^{(k+1)} - F^{(k)} - DF^{(k)}\delta x^{(k)})(\delta x^{(k)})^T}{\|\delta x^{(k)}\|_2^2}$$

oder vereinfacht

$$DF^{(k+1)} = DF^{(k)} \left(I + \frac{\overline{\delta x^{(k+1)}}(\delta x^{(k)})^T}{\|\delta x^{(k)}\|_2^2} \right)$$

mit

$$\overline{\delta x^{(k+1)}} := -(DF^{(k)})^{-1} F^{(k+1)} = \hat{d}^{(k+1)}.$$

Mittels der Sherman-Morrison-Formel erhält man die inverse Matrix:

$$(DF^{(k+1)})^{-1} = \left(I + \frac{\overline{\delta x^{(k+1)}}(\delta x^{(k)})^T}{\|\delta x^{(k)}\|_2^2 - (\delta x^{(k)})^T \overline{\delta x^{(k+1)}}} \right) (DF^{(k)})^{-1}.$$

Berechnet man ein Update der inversen Matrix $(DF^{(k)})^{-1}$ statt der Matrix $DF^{(k)}$, hat dies den Vorteil, dass zusätzlich zur schnelleren Berechnung der Funktionalmatrix auch die Lösung des linearen Gleichungssystems entfällt. Es muss lediglich eine Matrix-Vektor-Multiplikation durchgeführt werden, um $\delta x^{(k+1)} = -(DF^{(k+1)})^{-1}F^{(k+1)}$ zu bestimmen.

Es gibt noch weitere Rang-1-Updateformeln. Eine davon ist die sogenannte "schlechte"Broydenapproximation.[13] Diese liefert dabei in der Theorie eine genauso gute Näherung wie die "gute" Broydenapproximation. Dabei wird erneut ein Update der inversen Jacobimatrix $(DF^{(k)})^{-1}$ bestimmt:

$$(DF^{(k+1)})^{-1} = (DF^{(k)})^{-1}\left(I - \frac{F^{(k+1)}\delta F^{(k+1)}}{\|\delta F^{(k+1)}\|_2^2}\right)$$

mit

$$\delta F^{(k+1)} = F^{(k+1)} - F^{(k)}.$$

Ersetzt man im Newton-Verfahren die Jacobimatrix durch eine Broydenapproximation, so konvergiert das Verfahren unter geeigneten Voraussetzungen an F und die Schrittweitenwahl immer noch superlinear gegen eine Nullstelle x^* von F, d.h.

$$\lim_{x\to\infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0.$$

Ein Nutzer des vorliegenden Programms kann entscheiden, ob und falls ja, die „gute"oder die "schlechte"Broydenapproximation angewendet wird. Allerdings werden die Updateformeln aufgrund der verschlechterten Konvergenz des Newtonverfahrens gegenüber der Verwendung der exakten Funktionalmatrix nur dann benutzt, wenn in der vorhergehenden Iteration für alle Testfunktionen $T_i,\ i = 1, 2, 3$ die Ungleichung aus dem vorigen Abschnitt über die Dämpfung des Newtonverfahrens erfüllt ist.

2.1.4 Abbruchkriterien

Im Programm werden verschiedene Abbruchkriterien verwendet, um zu entscheiden, ob das Newtonverfahren eine Lösung gefunden hat. Dies sind in der i-ten Iteration des Newtonverfahrens die folgenden Kriterien:

1. Die skalierte Maximumnorm der Lösung des linearen Gleichungssystems $d^{(i)}$:

$$\max_{1\le m\le n}\left\{\lambda^{(i)}\frac{|d_m^{(i)}|}{\eta_{scal,m}^{(i)}}\right\} \le \varepsilon,$$

   mit $\eta_{scal}^{(i)} = \max\{1, |\eta^{(i)}|\}$ komponentenweise.

[13] Diese wird so bezeichnet, da sie sich in der Praxis als weniger gut herausgestellt hat.

2. Die skalierte Maximumnorm des Vektors $F^{(i+1)}$:

$$\max_{1\leq m\leq n}\left\{\frac{|F_m^{(i+1)}|}{\eta_{scal,m}^{(i)}}\right\}\leq\varepsilon,$$

mit $\eta_{scal}^{(i)}=\max\{1,|\eta^{(i)}|\}$ komponentenweise.

3. Die Maximumnorm der Differenz der Vektoren $F^{(i+1)}$ und $F^{(i)}$:

$$\max_{1\leq m\leq n}\left\{|F_m^{(i+1)}-F_m^{(i)}|\right\}\leq\sqrt{eps}.$$

Dabei sind $\eta^{(i)}$ die Lösung und $\lambda^{(i)}$ der Relaxationsfaktor des Newtonverfahrens in der i-ten Iteration sowie ε eine vom Benutzer wählbare Größe (Standardeinstellung: 10^{-4}). Die Skalierung hat den Effekt, dass an diejenigen Komponenten von $d^{(i)}$ bzw. $F^{(i+1)}$, die zu Komponenten des Lösungsvektors y gehören, welche betragsmäßig >1 sind, eine niedrigere Genauigkeitsforderung gestellt wird. Sind die beiden ersten Kriterien erfüllt, so wird das Newtonverfahren mit der Meldung, dass eine Lösung gefunden wurde, abgebrochen und diese ausgegeben. Konvergiert das Newtonverfahren hingegen nach einer ebenfalls vom Benutzer festzulegenden maximalen Zahl von Iterationen nicht, wird das Verfahren, ohne dass eine Lösung gefunden wurde, mit einer entsprechenden Meldung abgebrochen. Ein Problem des Newtonverfahrens besteht darin, dass sich oftmals die Lösung im Vergleich von einer Iteration zur nächsten kaum noch ändert, obwohl noch keine Nullstelle[14] gefunden wurde. Aus diesem Grund wurde das dritte Abbruchkriterium implementiert. Ändert sich der Vektor F von einer Iteration zur nächsten nur minimal, dann wird das Verfahren ebenfalls mit der Meldung, dass keine Lösung gefunden werden konnte, beendet.

2.2 Lösen der Anfangswertprobleme

An die Lösung der Anfangswertprobleme innerhalb der Mehrzielmethode werden verschiedene Anforderungen gestellt. Wie bereits in Abschnitt 2.1.1 erwähnt wurde, müssen die Anfangswertprobleme sehr genau gelöst werden, damit der relative Fehler bei der Berechnung von $F(s,sp,c)$ sowie $DF(s,sp,c)$ möglichst klein ist. Verfahren mit höherer Genauigkeit erfordern allerdings einen großen Rechenaufwand. Andererseits sollten Verfahren angewendet werden, die – aufgrund der großen Menge zu lösender Anfangswertprobleme[15] – einen geringen Rechenaufwand benötigen. Aus diesem Grund werden für

[14] Hier: eine Nullstelle des vektorwertigen Funktion F.

[15] Vergleiche wiederum Abschnitt 2.1.1: In jeder Iteration des Newtonverfahrens sind mindestens $m\cdot n+\alpha+\delta$ Anfangswertprobleme zu lösen, wobei das Randwertproblem $m-1$ Teilintervalle, ein n-dimensionales Differenzialgleichungssystem, α Schaltpunkte sowie δ Konstanten enthält.

nicht-steife Differenzialgleichungssysteme die eingebetteten Runge-Kutta-erfahren *Dormand-Prince 5(4)* sowie *Dormand-Prince 8(7)* zur Verfügung gestellt. Für die Lösung von steifen Differenzialgleichungssystemen ist ein *Rosenbrock-Typ-Verfahren* implementiert worden, welches zur Klasse der linear-impliziten Runge-Kutta-Verfahren gehört. Diese stellen einen Kompromiss aus expliziten und impliziten Runge-Kutta-Verfahren dar. In den beiden folgenden Abschnitten wird auf die angesprochenen Verfahren sowie deren Implementierung eingegangen.

2.2.1 Nichtsteife Differenzialgleichungen

Zur Lösung von Systemen nichtsteifer Differenzialgleichungen wurden die Verfahren *Dormand-Prince 5(4)*[16] sowie *Dormand-Prince 8(7)*[17] implementiert. Dabei stellt ersteres ein Verfahren dar, welches simple Anfangswertprobleme schnell und einfach lösen kann, dafür aber bei komplizierten Anfangswertproblemen nicht genau genug arbeitet, während das zweite Verfahren in der Regel einen höheren Rechenaufwand benötigt, allerdings auch eine höhere Genauigkeit erzielt. Beide Verfahren gehören zu den expliziten eingebetteten Runge-Kutta-Verfahren und sind somit Einschrittverfahren. Ihre Koeffizientenmatrizen sind demzufolge strikte untere Dreiecksmatrizen. Die Verfahren beinhalten jeweils 2 eingebettete Verfahren unterschiedlicher Ordnung, wodurch mit Hilfe einer Fehlerabschätzung eine automatische Schrittweitensteuerung ermöglicht wird.[18]

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
\gamma_i & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
\hat{\gamma}_i & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{200} & \frac{1}{40}
\end{array}
$$

Tabelle 2.1: Butcher-Tableau Dormand-Prince 5(4)

[16] Siehe [4].
[17] Siehe [5].
[18] Vergleiche dazu die zugehörigen Butcher-Tableaus in Tabelle 2.1 bzw. 2.2.

0												
$\frac{1}{18}$	$\frac{1}{18}$											
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$										
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$									
$\frac{5}{16}$	$\frac{5}{16}$	0	$-\frac{75}{64}$	$\frac{75}{64}$								
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$							
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$-\frac{28693883}{1125000000}$	$\frac{23124283}{1800000000}$						
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$-\frac{180193667}{1043307555}$					
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$-\frac{433636366}{683701615}$	$-\frac{421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$				
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$-\frac{37695042795}{15268766246}$	$-\frac{309121744}{1061227803}$	$-\frac{12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{393006217}{1396673457}$	$\frac{124872331}{1001029789}$			
$\frac{1201146811}{1299019798}$	$-\frac{1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{1432422823}$	$-\frac{10304129995}{1701304382}$	$-\frac{48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$-\frac{45442868181}{3398467696}$	$\frac{3065993473}{597172653}$		
1	$\frac{185892177}{718116043}$	0	0	$-\frac{3185094517}{667107341}$	$-\frac{477755414}{1098053517}$	$-\frac{703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$-\frac{4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$	
1	$\frac{403863854}{491063109}$	0	0	$-\frac{5068492393}{434740067}$	$-\frac{411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$-\frac{13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$-\frac{160528059}{685178525}$	$\frac{248638103}{1413531060}$	0
γ_i	$\frac{14005451}{335480064}$	0	0	0	0	$-\frac{59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{561292985}{797845732}$	$-\frac{1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$-\frac{528747749}{2220607170}$	$\frac{1}{4}$
$\hat{\gamma}_i$	$\frac{13451932}{455176623}$	0	0	0	0	$-\frac{808719846}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$-\frac{3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$	0

Tabelle 2.2: Butcher-Tableau Dormand-Prince 8(7)

Die für die automatische Schrittweitensteuerung benötigte Fehlerabschätzung wird über die folgenden Formeln realisiert

$$\begin{aligned} err &= y_{i+1} - \hat{y}_{i+1}, \quad err \in \mathbb{R}^n, \\ errmax &= \max_{1 \le m \le n} \left\{ \frac{err_m}{y_{scal,m}} \right\}, \end{aligned} \tag{2.8}$$

mit $y_{scal} = \max\{1, |y|\}$ (komponentenweise). Dabei stellt y das Ergebnis des Verfahrens $(p+1)$-ter Ordnung und $\hat{y}$ das Ergebnis des Verfahrens p-ter Ordnung dar, mit $p = 4$ bei *Dormand-Prince 5(4)* bzw. $p = 7$ bei *Dormand-Prince 8(7)*. Als Fehlerabschätzung wird das skalierte Maximum $errmax$ der Komponenten des Fehlervektors err aus (2.8) gewählt. Unterschreitet diese Fehlerschätzung eine gewisse Grenze tol[19], wird der ausgeführte Schritt akzeptiert, anderenfalls nicht. Die neue Schrittweite h_{neu} wird folgendermaßen bestimmt[20]

$$h_{neu} = \begin{cases} 0.9 \cdot h \cdot \sqrt[p]{\frac{tol}{errmax}}, & \text{falls } errmax > tol \\ 0.9 \cdot h \cdot \sqrt[p+1]{\frac{tol}{errmax}}, & \text{falls } errmax \le tol \end{cases}, \tag{2.9}$$

wobei zusätzlich gelte:

$$0.5 \cdot h \le h_{neu} \le 1.5 \cdot h.$$

Außerdem darf die Schrittweite eine vom Benutzer wählbare untere Schranke h_{min} nicht unterschreiten.

Bemerkung 2.2.1 Um zu verhindern, dass die Verfahren bei der Lösung eines Anfangswertproblems nicht zum Ziel führen, wurden zwei weitere Größen implementiert, die vom Benutzer gewählt werden können. Tritt einer der beiden Fälle ein, so wird die Mehrzielmethode nach der aktuellen Iteration mit einer Fehlermeldung abgebrochen. Zum Einen ist die Anzahl der Verkleinerungen der Schrittweite in jedem Zeitschritt beschränkt. Wird diese Grenze erreicht, ohne dass die Fehlerabschätzung die Schranke tol unterschreitet, wird der Schritt dennoch akzeptiert. In diesem Fall verletzt der akzeptierte Zeitschritt zwar die Genauigkeitsforderung, gleichzeitig wird aber verhindert, dass das Verfahren unendlich oft versucht, die Schrittweite zu verkleinern. Die zweite Größe ist die Beschränkung der Zahl der Zeitschritte. Diese obere Grenze kann beispielsweise dann erreicht werden, wenn man mit einem der beiden zur Verfügung stehenden Dormand-Prince-Verfahren versucht, ein steifes Differenzialgleichungssystem zu lösen oder das Integrationsintervall zu groß gewählt wurde. □

[19] Diese kann im Programm je nach gewünschter Genauigkeit frei gewählt werden.
[20] [18] S.135ff., [15] S.719ff. und S.743ff.

Algorithmus 2.2.2 *Dormand-Prince-Verfahren*
Es sei das Anfangswertproblem $y' = f(y)$ mit dem Startwert $y(x_s)$ gegeben und dessen Lösung $y(x_e)$ gesucht.

1. *Initialisierung: $\xi := x_s,\ \eta := y(x_s)$, sowie eine Startschrittweite h seien gegeben.*[21]
2. *Ist $\xi = x_e \Rightarrow$ STOPP.*
3. *Berechnen von $\hat{\eta}$ sowie der Fehlerabschätzung nach (2.8).*
4. *Berechnung der neuen Schrittweite h nach (2.9)*
   - *Schritt akzeptiert: $\xi := \xi + h,\ \eta := \hat{\eta}\ \Rightarrow$ Gehe zu 2.*
   - *Schritt nicht akzeptiert $\Rightarrow$ Gehe zu 3.*

2.2.2 Steife Differenzialgleichungen

In diesem Abschnitt wird ein Rosenbrock-Typ-Verfahren[22] sowie dessen Implementierung in C++ vorgestellt. Diese stellen eine Klasse von Einschrittverfahren dar, die aus den Runge-Kutta-Verfahren abgeleitet werden können. Die expliziten Einschrittverfahren sind für steife Differenzialgleichungssysteme ungeeignet, da diese in einem solchen Fall nur für sehr kleine Schrittweiten stabil sind.[23] Die Rosenbrock-Typ-Verfahren sind sogenannte linear-implizite Runge-Kutta-Verfahren, welche einen Kompromiss aus expliziten und impliziten Runge-Kutta-Verfahren darstellen, weil sie einerseits den Vorteil der impliziten Verfahren – die Stabilität – behalten und andererseits den moderaten Arbeitsaufwand der expliziten Verfahren aufweisen.

Zunächst betrachtet man ein n-dimensionales autonomes Anfangswertproblem

$$y' = f(y), \quad y(0) = y_0,$$

und ein s-stufiges Runge-Kutta-Verfahren. Implizite Runge-Kutta-Verfahren weisen – im Gegensatz zu den expliziten – eine Koeffizientenmatrix A auf, die keine strikte untere Dreiecksgestalt hat. Daher ist in jedem Zeitschritt ein nichtlineares Gleichungssystem der Dimension $s \cdot n \times s \cdot n$ zu lösen, woraus ein hoher Rechenaufwand resultiert. Ist diese Koeffizientenmatrix A nun eine untere Dreiecksmatrix mit nicht verschwindenden Diagonalelementen, erhält man s entkoppelte Gleichungen für die Steigungen, sodass lediglich s n-dimensionale nichtlineare Gleichungssysteme zu lösen sind:

$$k_j = hf\left(y_i + \sum_{\nu=1}^{j-1} \alpha_{j\nu} k_\nu + \alpha_{jj} k_j\right), \quad j = 1, \ldots, s.$$

[21] Die Startschrittweite ist im Programm relativ zum Intervall $[x_s, x_e]$ frei wählbar.
[22] Vergleiche dazu [8] S.601-607 sowie [7] S.110-127.
[23] Zu steifen Differenzialgleichungen sowie zur Stabilität Vergleiche auch [7] und [8].

Die Anwendung des Newton-Verfahrens zur Lösung der j-ten Gleichung in der l-ten Iteration führt auf

$$(I - \alpha_{jj} hJ)(k_j^{(l)} - k_j^{(l-1)}) = hf\left(y_i + \sum_{\nu=1}^{j-1} \alpha_{j\nu} k_\nu + \alpha_{jj} k_j^{(l-1)}\right) - k_j^{(l-1)}$$

beziehungsweise

$$(I - \alpha_{jj} hJ) k_j^{(l)} = hf\left(y_i + \sum_{\nu=1}^{j-1} \alpha_{j\nu} k_\nu + \alpha_{jj} k_j^{(l-1)}\right) - \alpha_{jj} hJ k_j^{(l-1)}.$$

Dabei wird nicht die Ableitung an der aktuellen y-Koordinate, sondern stattdessen die Näherungsableitung $J = \frac{\partial f}{\partial y}(y_i)$ verwendet. Dies hat den Effekt, dass diese Ableitung in jedem Zeitschritt nur einmal bestimmt werden muss, sodass der benötigte Rechenaufwand gesenkt wird. Häufig genügt es dabei, lediglich einen Newtonschritt durchzuführen, sofern die Schrittweite vernünftig gewählt wird und man über eine hinreichend gute Startnäherung $k_j^{(0)}$ verfügt. Wählt man

$$k_j^{(0)} = \sum_{\nu=1}^{j-1} \gamma_{j\nu}/\alpha_{jj} * k_\nu$$

mit zusätzlichen Koeffizienten $\gamma_{i\nu}$, so erhält man ein sogenanntes linear-implizites Runge-Kutta-Verfahren:

$$\begin{aligned}(I - \alpha_{jj} hJ) k_j &= hf\left(y_i + \sum_{\nu=1}^{j-1} (\alpha_{j\nu} + \gamma_{j\nu}) k_\nu\right) - hJ \sum_{\nu=1}^{j-1} \gamma_{j\nu} k_\nu, \quad j = 1, \ldots, s,\\ y_{i+1} &= y_i + \sum_{j=1}^{s} b_j k_j.\end{aligned}$$

Auf jeder Stufe j, $j = 1, \ldots, s$ muss ein lineares Gleichungssystem gelöst werden. Verwendet man nun auf jeder Stufe den gleichen Koeffizienten $\alpha_{jj} = \alpha$, so besitzen alle linearen Gleichungssysteme die gleiche Koeffizientenmatrix $I - \alpha hJ$. Daher reicht es aus, in jedem Schritt lediglich eine LR-Zerlegung durchzuführen.[24] Diese Vereinfachung und das Lösen von der bisherigen Bedeutung der Koeffizienten $\alpha_{j\nu}$ führt dann auf die Rosenbrock-Typ-Verfahren für autonome Systeme:

$$\begin{aligned}(I - \alpha hJ) k_j &= hf\left(y_i + \sum_{\nu=1}^{j-1} \alpha_{j\nu} k_\nu\right) - hJ \sum_{\nu=1}^{j-1} \gamma_{j\nu} k_\nu, \quad j = 1, \ldots, s,\\ y_{i+1} &= y_i + \sum_{j=1}^{s} b_j k_j\end{aligned}$$

[24] Für $0 < h < 1/(2\alpha\|J\|_2)$ ist die Matrix $I - \alpha hJ$ regulär und damit die LR-Zerlegung der Koeffizientenmatrix möglich. Im Programm wird für die LR-Zerlegung der Algorithmus von Crout verwendet. Dieser beinhaltet eine Spaltenpivotsuche sowie eine implizite Skalierung der Zeilen der Matrix. Mehr dazu siehe [15] S.47ff.

mit geeignet zu wählenden Parametern α, $\alpha_{j\nu}$, $\gamma_{j\nu}$ und b_j, die im Hinblick auf Ordnung und Stabilität des Verfahrens optimiert werden können.

Betrachte nun ein Differenzialgleichungssystem der Form

$$y' = f(x, y), \quad y(0) = y_0.$$

Jedes nichtautonome Differenzialgleichungssystem lässt sich durch Einführung einer zusätzlichen Komponente in ein autonomes transformieren.[25] Diese Umformung führt auf ein allgemeines Rosenbrock-Typ-Verfahren:

$$\begin{aligned}
(I - \alpha h J)k_j &= hf\left(x_i + \alpha_j h, y_i + \sum_{\nu=1}^{j-1} \alpha_{j\nu} k_\nu\right) + \gamma_j h^2 \frac{\partial f}{\partial x}(x_i, y_i) \\
&\quad - hJ \sum_{\nu=1}^{j-1} \gamma_{j\nu} k_\nu, \quad j = 1, \ldots, s, \\
y_{i+1} &= y_i + \sum_{j=1}^{s} b_j k_j,
\end{aligned}$$

mit

$$J = \frac{\partial f}{\partial y}(x_i, y_i), \qquad \alpha_j = \sum_{\nu=1}^{j-1} \alpha_{j\nu}, \qquad \gamma_j = \sum_{\nu=1}^{j-1} \gamma_{j\nu}.$$

In jeder Stufe dieses Verfahrens muss man die Matrix-Vektor-Multiplikation $J \cdot \sum_{\nu=1}^{j-1} \gamma_{j\nu} k_\nu$ durchführen. Um den dafür nötigen Rechenaufwand zu reduzieren, führt man die folgende Ersetzung ein:

$$g_j := \sum_{\nu=1}^{j-1} \gamma_{j\nu} k_\nu + \alpha k_j.$$

So erhält man ein effizienteres Rosenbrock-Typ-Verfahren

$$\begin{aligned}
\left(\frac{1}{\alpha h} I - J\right) g_j &= f\left(x_i + a_j h, y_i + \sum_{\nu=1}^{j-1} a_{j\nu} g_\nu\right) \\
&\quad + c_j h \frac{\partial f}{\partial x}(x_i, y_i) \\
&\quad - \frac{1}{h} \sum_{\nu=1}^{j-1} c_{j\nu} g_\nu, \quad j = 1, \ldots, s, \\
y_{i+1} &= y_i + \sum_{j=1}^{s} b_j g_j
\end{aligned}$$

mit den Koeffizienten α, $a_{j\nu}$, $c_{j\nu}$, a_j, c_j, b_j.

[25] Vergleiche S.9.

Analog zu den expliziten eingebetteten Runge-Kutta-Verfahren[26] wird auch hier durch Einbettung zweier Verfahren eine automatische Schrittweitensteuerung implementiert. Diese Verfahren der Ordnungen q und $q-1$ besitzen die gleichen Koeffizienten, damit der Rechenaufwand nicht unnötig erhöht wird.

Bemerkung 2.2.3 (Koeffizientensatz Shampine(1982))
Hier wird der Koeffizientensatz von Shampine[27] verwendet, welcher zwei Verfahren dritter und vierter Ordnung ineinander einbettet.

- $\alpha = 0.5$
- $a_{21} = 2,\ a_{31} = \frac{48}{25},\ a_{32} = \frac{6}{25}$
- $c_{21} = -8,\ c_{31} = \frac{372}{25},\ c_{32} = \frac{12}{5},\ c_{41} = -\frac{112}{125},\ c_{42} = -\frac{54}{125},\ c_{43} = -0.4$
- $a_2 = 1,\ a_3 = 0.6$
- $c_1 = 0.5,\ c_2 = -1.5,\ c_3 = \frac{121}{50},\ c_4 = \frac{29}{250}$
- $b_1 = \frac{19}{9},\ b_2 = 0.5,\ b_3 = \frac{25}{108},\ b_4 = \frac{125}{108}$
- $e_1 = \frac{17}{54},\ e_2 = \frac{7}{36},\ e_3 = 0,\ e_4 = \frac{125}{108}$

□

Die für die automatische Schrittweitensteuerung benötigte Fehlerabschätzung wird über die folgenden Formeln realisiert

$$\begin{aligned} err &= y_{i+1} - \hat{y}_{i+1} = \sum_{j=1}^{s} e_j g_j, \quad err \in \mathbb{R}^n, \\ errmax &= \max_{1 \le m \le n} \left\{ \frac{err_m}{y_{scal,m}} \right\}, \end{aligned} \tag{2.10}$$

mit $y_{scal} = \max\{1, |y_i|\}$ (komponentenweise), wobei y das Ergebnis des Verfahrens 4. Ordnung und $\hat{y}$ das Ergebnis des Verfahren 3. Ordnung darstellt. Um weiteren Rechenaufwand einzusparen, wird das Ergebnis des niedrigeren Verfahrens nicht explizit berechnet, sondern stattdessen über die Koeffizienten $e_j,\ j = 1, \ldots, s$ direkt die Differenz zwischen den Ergebnissen beider Verfahren. Als Fehlerabschätzung wird dann das skalierte Maximum $errmax$ der Komponenten des Fehlervektors err aus (2.10) gewählt. Unterschreitet diese Fehlerschätzung eine gewisse Grenze tol[28], wird der ausgeführte Schritt

[26] Vergleiche dazu z.B. [8] S.608ff.
[27] Siehe [15] S.744 – ein weiterer Koeffizientensatz von Kaps-Rentrop findet sich auf S. 745. Für weitere Koeffizientensätze sowie deren Stabilität vergleiche [7] S.119.
[28] Diese kann im Programm je nach gewünschter Genauigkeit frei gewählt werden.

akzeptiert, andernfalls nicht. Die neue Schrittweite h_{neu} wird folgendermaßen bestimmt:[29]

$$h_{neu} = \begin{cases} 0.9 \cdot h \cdot \sqrt[3]{\frac{tol}{errmax}}\,, & \text{falls } errmax > tol \\ 0.9 \cdot h \cdot \sqrt[4]{\frac{tol}{errmax}}\,, & \text{falls } errmax \leq tol \end{cases}, \tag{2.11}$$

wobei zusätzlich gelte:

$$0.5 \cdot h \leq h_{neu} \leq 1.5 \cdot h.$$

Außerdem darf die Schrittweite eine vom Benutzer wählbare untere Schranke nicht unterschreiten.

Die Ableitungen in jedem Zeitschritt werden über zentrale Differenzenquotienten

$$\begin{aligned} J = \frac{\partial f}{\partial y}(x_i, y_i) &= \frac{f(x_i, y_i + h) - f(x_i, y_i - h)}{2h} + O(h^2), \\ \frac{\partial f}{\partial x}(x_i, y_i) &= \frac{f(x_i + h, y_i) - f(x_i - h, y_i)}{2h} + O(h^2) \end{aligned} \tag{2.12}$$

mit der Störung $h = |y_i|\sqrt[3]{eps}$, $h \geq \sqrt[3]{eps}$, wobei eps die Maschinengenauigkeit ist, ermittelt.[30] Dies hat zum Einen den Nachteil eines höheren Rechenaufwands aufgrund der Mehrzahl an Funktionsauswertungen gegenüber dem Vorwärts- oder Rückwärtsdifferenzenquotienten, auf der anderen Seite erreicht man allerdings eine höhere Genauigkeit bei der Berechnung der Ableitungen. Jede Funktionsauswertung mit muParser[31] benötigt deutlich mehr Zeit als dieselbe Auswertung von im Quellcode stehenden Funktionen. Daher wird durch Nullsetzen von Ableitungen, sofern x und/oder Komponenten von y nicht in der rechten Seite des Differenzialgleichungssystems auftauchen[32], die Anzahl der nötigen Funktionsauswertungen reduziert um Rechenzeit einzusparen.

Analog zu den implementierten Verfahren von Dormand-Prince wurden auch hier zwei vom Benutzer wählbare Größen integriert, um sicherzustellen, dass das Verfahren nicht unendlich lang versucht, eine Lösung des gegebenen Anfangswertproblems zu bestimmen.[33]

Algorithmus 2.2.4 *Rosenbrock-Typ-Verfahren*
Es sei das Anfangswertproblem $y' = f(y)$ mit dem Startwert $y(x_s)$ gegeben und dessen Lösung $y(x_e)$ gesucht.

1. *Initialisierung: $\xi := x_s$, $\eta := y(x_s)$, sowie eine Startschrittweite h seien gegeben.*[34]

[29] [18] S.135ff., [15] S.719ff. und S.743ff.
[30] $eps \approx 2.220446049 \cdot 10^{-16}$, also $\sqrt[3]{eps} \approx 1.304608114 \cdot 10^{-6}$.
[31] Siehe Abschnitt 3.2.
[32] Ist das Differenzialgleichungssystem unabhängig von x, handelt es sich um ein autonomes Problem.
[33] Vergleiche dazu Bemerkung 2.2.1 auf Seite 36.
[34] Die Startschrittweite ist im Programm relativ zum Intervall $[x_s, x_e]$ frei wählbar.

2. *Ist* $\xi = x_e \Rightarrow$ *STOPP.*

3. *Bestimmung der Jacobimatrix* $J = \frac{\partial f}{\partial y}(x_i, y_i)$ *sowie* $\frac{\partial f}{\partial x}(x_i)$ *nach (2.12).*

4. *Matrix* $(\frac{1}{\alpha h} I - J)$ *bestimmen und LR-Zerlegung nach Crouts Algorithmus durchführen.*

5. *Berechnung von* $g_j,\ j = 1, 2, 3, 4$ *(Bestimmen der jeweiligen rechten Seite des LGS und Lösen desselbigen durch Rückwärts- und Vorwärtssubstitution).*

6. *Ermittlung von* $\hat{\eta}$ *sowie der Fehlerabschätzung nach (2.10).*

7. *Berechnung der neuen Schrittweite* h *nach (2.11):*

   - *Schritt akzeptiert:* $\xi := \xi + h,\ \eta := \hat{\eta} \ \Rightarrow$ *Gehe zu 2.*
   - *Schritt nicht akzeptiert* $\Rightarrow$ *Gehe zu 4.*

2.3 Die Singulärwertzerlegung

Wie bereits in Abschnitt 2 erwähnt wurde, ist bei der Mehrzielmethode in jeder Iteration ein lineares Gleichungssystem zu lösen. Dazu wurde die Singulärwertzerlegung implementiert, da mittels dieser - im Gegensatz zu vielen anderen Verfahren - auch singuläre bzw. fastsinguläre Gleichungssysteme stabil gelöst werden können. Das Householder-Verfahren beispielsweise versagt bei singulären Matrizen und die Mehrzielmethode müsste abgebrochen werden oder alternativ ein anderes zusätzliches Verfahren, welches mit singulären Matrizen umgehen kann, angewendet werden. Dies ist bei der Singulärwertzerlegung nicht der Fall. Ist das Gleichungssystem eindeutig lösbar, liefert diese die eindeutig bestimmte Lösung und im Falle einer singulären Matrix diejenige Lösung, deren euklidische Norm minimal ist.[35] Da die Lösung des Gleichungssystems einen relativ geringen Anteil am gesamten Rechenaufwand einer Iteration der Mehrzielmethode hat, kann der gegenüber anderen Verfahren höhere Rechenaufwand in Kauf genommen werden. Außerdem liefert die Singulärwertzerlegung ohne großen Mehraufwand nicht nur die gesuchte Lösung des Gleichungssystems, sondern auch die Pseudo-Inverse[36], welche für das Broyden-Update verwendet werden kann, da dort idealerweise nicht die Jacobimatrix sondern deren Inverse approximiert wird.[37] Im folgenden Abschnitt wird auf die Existenz sowie einige wichtige Eigenschaften eingegangen und anschließend die Implementierung der numerischen Bestimmung der Singulärwertzerlegung einer Matrix genauer erläutert.

[35] Dazu mehr in Abschnitt 2.3.1.

[36] Diese wird auch als Moore-Penrose-Inverse bezeichnet und stimmt, sofern die Ausgangsmatrix regulär ist, mit der inversen Matrix überein.

[37] So kann bei der Anwendung dieser Approximation nicht nur die sehr aufwendige Berechnung der Jacobimatrix, sondern zusätzlich auch die Lösung des linearen Gleichungssystems eingespart werden. Vergleiche dazu Abschnitt 2.1.3.

2.3.1 Theorie

Die folgenden Bemerkungen werden beim Beweis des nachfolgenden Satzes (2.3.2) benötigt.

Bemerkung 2.3.1 Sei eine Matrix $A \in \mathbb{R}^{n\times n}$ gegeben. Dann ist $A^T A$ symmetrisch und außerdem positiv semidefinit, da

$$x^T(A^T A)x = \|Ax\|_2^2 \geq 0, \quad \forall x \in \mathbb{R}^n,$$

gilt. Die Matrix $A^T A$ besitzt demzufolge nur reelle und nichtnegative Eigenwerte, die der Größe nach geordnet seien:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0.$$

Es werden nun diese Eigenwerte in der Form

$$\lambda_i = \sigma_i^2, \qquad \sigma_i \geq 0,$$

geschrieben. Die Zahlen $\sigma_i \geq 0,\ i = 1, \ldots, n$ heißen Singulärwerte der Matrix A. □

Satz 2.3.2 (Singulärwertzerlegung)[38]
Es sei $A \in \mathbb{R}^{n\times n}$ eine beliebige quadratische Matrix.[39] *Dann existieren zwei orthogonale Matrizen $U \in \mathbb{R}^{n\times n}$ und $V \in \mathbb{R}^{n\times n}$, sodass*

$$U^T AV = \Sigma$$

eine $(n \times n)$-Diagonalmatrix der folgenden Form ist:

$$\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}, \qquad D \equiv diag(\sigma_1, \ldots, \sigma_r), \qquad \sigma_1 \geq \sigma_2 \cdots \geq \sigma_r > 0.$$

Die $\sigma_1, \ldots, \sigma_r$ stellen hierbei die nichtverschwindenden Singulärwerte der Matrix A dar und es gilt: $rang(A) = r$.

Beweis: Der Beweis basiert auf dem folgenden aus der linearen Algebra bekannten Resultat:

- Eigenwertzerlegung:
  Zu jeder reellen und symmetrischen Matrix $A \in \mathbb{R}^{n\times n}$ existiert eine orthogonale Matrix $V \in \mathbb{R}^{n\times n}$ mit

  $$V^T AV = \Lambda = diag(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n\times n},$$

  wobei $\lambda_1, \ldots, \lambda_n$ die Eigenwerte von A sind.

[38] Siehe [10].

[39] Der Satz (2.3.2) gilt für beliebige Matrizen. Da im Rahmen der Mehrzielmethode jedoch lediglich quadratische Matrizen auftauchen, wird hier von einer beliebigen quadratischen Matrix ausgegangen.

Wie bereits in Bemerkung 2.3.1 festgestellt, ist die Matrix $A^T A$ symmetrisch und positiv semidefinit. Damit ist die Voraussetzung der obigen Aussage erfüllt, d.h. es existiert eine orthogonale Matrix $V \in \mathbb{R}^{n\times n}$, sodass

$$V^T(A^T A)V = \Lambda = diag(\lambda_1, \dots, \lambda_n), \quad \lambda_i \geq 0, \quad \forall i = 1, \dots, n,$$

gilt. O.B.d.A. kann man davon ausgehen, dass folgendes gilt:

$$\lambda_1 \geq \cdots \geq \lambda_r > 0, \quad \lambda_{r+1} = \cdots = \lambda_n = 0, \quad r \in \{0, \dots, n\}.$$

Aus der Eigenwertgleichung $A^T A v_j = \lambda_j v_j$, wobei v_j die j-te Spalte von V sei, folgt

$$\lambda_j = v_j^T A^T A v_j = \|Av_j\|_2^2$$

und damit

$$Av_j \begin{cases} \neq 0, & j = 1, \dots, r \\ = 0, & j = r+1, \dots, n \end{cases}.$$

Mit

$$\sigma_j \equiv \sqrt{\lambda_j} \quad \text{und} \quad u_j \equiv Av_j/\sigma_j, \quad j = 1, \dots, r$$

folgt daraus für $i, j = 1, \dots, r$

$$u_i^T u_j = \frac{1}{\sigma_i \sigma_j} v_i^T A^T A v_j = \frac{\lambda_j}{\sigma_i \sigma_j} v_i^T v_j = \delta_{ij}.$$

Die Vektoren $\{u_1, \dots, u_r\}$ bilden daher ein Orthonormalsystem und können durch Hinzunahme weiterer Vektoren $u_{r+1}, \dots, u_n$ zu einer Orthonormalbasis des R^n ergänzt werden. Damit ist die resultierende Matrix $U = (u_1, \dots, u_n) \in \mathbb{R}^{n\times n}$ orthogonal.

Mit $\sigma_{r+1} = \cdots = \sigma_n = 0$ folgt wegen

$$(U^T AV)_{ij} = u_i^T A v_j = \begin{cases} \sigma_j, & j = 1, \dots, r \\ 0, & j = r+1, \dots, n \end{cases}$$

die gesuchte Beziehung $U^T AV = \Sigma$. □

Definition 2.3.3 (Singulärwertzerlegung) *Die Zerlegung der Matrix A*

$$A = U\Sigma V^T$$

wird Singulärwertzerlegung genannt.

Satz 2.3.4 *Sei $U^T AV = \Sigma$ die Singulärwertzerlegung von $A \in \mathbb{R}^{n\times n}$ mit $rang(A) = r$ und*

$$\Sigma = diag(\sigma_1, \dots, \sigma_r, 0, \dots, 0).$$

Dann ist die Pseudoinverse $A^+ \in \mathbb{R}^{n\times n}$ definiert durch:

$$A^+ = V\Sigma^+ U^T \quad mit \quad \Sigma^+ = diag(\sigma_1^{-1}, \dots, \sigma_r^{-1}, 0, \dots, 0).$$

Ist $rang(A) = n$, so gilt: $A^+ = A^{-1}$.

Beweis: Siehe [3] S.147. □

2.3.2 Numerische Bestimmung[40]

Wie im vorigen Abschnitt bereits erwähnt wurde, sind die Singulärwerte σ_i einer Matrix $A \in \mathbb{R}^{n\times n}$ die Wurzeln der Eigenwerte von $A^T A$

$$\sigma_i(A) = \sqrt{\lambda_i(A^T A)}.$$

Da das Eigenwertproblem der Matrix $A^T A$ gut konditioniert ist, ist ebenso das Singulärwertproblem von A gut konditioniert. Dies gilt allerdings nur insoweit, wie sich die explizite Berechnung von $A^T A$ vermeiden lässt, da sich die Konditionszahl in diesem Fall quadrieren würde. Besitzt die Matrix A kleine von Null verschiedene Singulärwerte, werden diese ungenau berechnet. Man muss also versuchen, eine Methode zur Bestimmung der Singulärwerte zu entwickeln, die die explizite Berechnung von $A^T A$ vermeidet. Es zeigt sich, dass viele Ansätze zur Lösung eines Eigenwertproblems ebenfalls auf das Singulärwertproblem angewendet werden können. Das Eigenwertproblem vereinfacht sich stark, wenn die Matrix Tridiagonalgestalt besitzt, da dann beispielsweise der QR-Algorithmus effizient angewendet werden kann. Besitzt die Matrix A Bidiagonalgestalt, so ist die Matrix $A^T A$ tridiagonal.

Definition 2.3.5 (Bidiagonalgestalt) *Eine Matrix $A \in \mathbb{R}^{n\times n}$ heißt bidiagonal, wenn gilt: $b_{ij} = 0,\ \forall i > j$ und $i < j - 1$, d.h. die Matrix hat folgende Gestalt*

$$A = \begin{pmatrix} * & * & & & \\ & * & * & & \\ & & * & \ddots & \\ & & & \ddots & * \\ & & & & * \end{pmatrix}.$$

Die beiden folgenden Lemmata besagen, dass man jede beliebige Matrix mittels orthogonaler Transformationen auf Bidiagonalgestalt bringen kann und sich die Singulärwerte dabei nicht ändern.

Lemma 2.3.6 *Für jede Matrix $A \in \mathbb{R}^{n\times n}$ existieren orthogonale Matrizen $U \in \mathbb{R}^{n\times n}$ und $V \in \mathbb{R}^{n\times n}$, sodass*

$$UAV = \begin{pmatrix} * & * & & & \\ & * & * & & \\ & & * & \ddots & \\ & & & \ddots & * \\ & & & & * \end{pmatrix} = B,$$

wobei $B \in \mathbb{R}^{n\times n}$ Bidiagonalgestalt hat.

Lemma 2.3.7 *Sei $A \in \mathbb{R}^{n\times n}$ eine beliebige Matrix und seien $U \in \mathbb{R}^{n\times n}$ und $V \in \mathbb{R}^{n\times n}$ orthogonale Matrizen. Dann besitzen A und $B := UAV$ die gleichen Singulärwerte.*

[40] Die Ausführungen zur numerischen Bestimmung sind aus [3] S.148ff entnommen.

Diese Transformationen lassen sich beispielsweise durch abwechselnde Multiplikationen der Matrix A von links und rechts mit Householdermatrizen realisieren. Dabei wird die Matrix A im ersten Schritt von links mit U_1 multipliziert, um in den Einträgen $a_{i1},\ i = 2, \ldots, n$ Nullen zu erzeugen, und anschließend von rechts mit V_1, um in den Einträgen $a_{1j},\ j = 3, \ldots, n$ ebenfalls Nullen zu erhalten. Analog geht man in den weiteren Schritten vor, um die Matrix auf Bidiagonalgestalt zu bringen.

$$\begin{pmatrix} * & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \end{pmatrix} \xrightarrow{U_1\cdot} \begin{pmatrix} * & * & \cdots & * \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix} \xrightarrow{\cdot V_1} \begin{pmatrix} * & * & 0 & \cdots & 0 \\ 0 & * & \cdots & \cdots & * \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & * & \cdots & \cdots & * \end{pmatrix}$$

$$\xrightarrow{U_2\cdot} \begin{pmatrix} * & * & 0 & \cdots & 0 \\ 0 & * & * & \cdots & 0 \\ \vdots & 0 & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{pmatrix} \xrightarrow{\cdot V_2} \ldots \xrightarrow{U_{n-1}\cdot} \begin{pmatrix} * & * & & \\ & \ddots & \ddots & \\ & & \ddots & * \\ & & & * \end{pmatrix}$$

Man erhält also nach $n-1$ Schritten die Bidiagonalmatrix B

$$B = \underbrace{U_{n-1} \ldots U_1}_{=:U} A \underbrace{V_1 \ldots V_{n-2}}_{=:V}.$$

Bemerkung 2.3.8 Nach der Bidiagonalisierung durch Householder-Transformationen sind viele Einträge außerhalb der beiden Diagonalen aufgrund von Rundungsfehlern nicht identisch Null, sondern besitzen Werte in der Größenordnung der Maschinengenauigkeit. Um zu vermeiden, dass dies Auswirkungen auf das anschließende Chasing-Verfahren hat, werden diese auf Null gesetzt. □

Die Berechnung der Singulärwertzerlegung der Matrix A wurde nun also bereits auf die Bestimmung der Singulärwertzerlegung der Bidiagonalmatrix B reduziert. Dieses Problem ist äquivalent zur Berechnung der Eigenwerte und -vektoren der Tridiagonalmatrix $B^T B$. Für diese Aufgabe stehen verschiedene numerische Verfahren zur Verfügung, beispielsweise das QR-Verfahren. Da die explizite Berechnung des Produkts $B^T B$ numerische Probleme mit sich bringen kann[41], stellt sich nun die Frage, ob man den bekannten QR-Algorithmus so umformulieren kann, dass man die explizite Berechnung des Matrixprodukts vermeiden kann. Dies bedeutet, dass der QR-Algorithmus nur auf der Matrix B operiert und diese mittels orthogonaler Transformationen auf Diagonalgestalt bringt. Dann besitzt auch $B^T B$ Diagonalgestalt und die Singulärwerte entsprechen gerade den Diagonalelementen. Führt man den ersten Givens-Eliminationsschritt des QR-Algorithmus auf

[41] Analog zu Berechnung von $A^T A$.

$B^T B$ aus,

$$B^T B \to \Omega_{12} B^T B \Omega_{12}^T = \underbrace{(B\Omega_{12}^T)^T}_{\tilde{B}^T} \underbrace{B\Omega_{12}}_{\tilde{B}},$$

so erhält man für $\tilde{B}$ die Matrix

$$\tilde{B} = B\Omega_{12} = \begin{pmatrix} * & * & & & \\ \oplus & * & * & & \\ & & \ddots & \ddots & \\ & & & \ddots & * \\ & & & & * \end{pmatrix}.$$

Dabei wurde an der Stelle b_{21} ein neues Element $\oplus$ erzeugt. Betrachtet man die weiteren Schritte des QR-Algorithmus, zeigt sich, dass dieser auf der Matrix B dem folgenden Eliminationsprozess entspricht

$$\begin{pmatrix} * & * & z_3 & & & & \\ z_2 & * & * & z_5 & & & \\ & z_4 & * & * & z_7 & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & z_{2n-6} & * & * & z_{2n-3} \\ & & & & z_{2n-4} & * & * \\ & & & & & z_{2n-2} & * \end{pmatrix}$$

eliminiere z_2	(Givens von links)	$\to$ erzeugt z_3
eliminiere z_3	(Givens von rechts)	$\to$ erzeugt z_4
$\vdots$	$\vdots$	$\vdots$
eliminiere z_{2n-3}	(Givens von rechts)	$\to$ erzeugt z_{2n-2}
eliminiere z_{2n-2}	(Givens von links).	

Man erzeugt also durch eine Givens-Rotation von rechts (ggfs. mit Shift) einen Eintrag $z_2 \neq 0$. Anschließend "verfolgt"man diesen Eintrag durch die Matrix, indem man abwechselnd durch Givens-Rotationen von links bzw. rechts den entstandenen Eintrag z_i eliminiert und dadurch einen neuen Eintrag z_{i+1} erzeugt.[42] Nach z_{n-1} Givens-Rotationen hat die Matrix wieder Bidiagonalgestalt und ein Schritt des QR-Verfahrens wurde nur auf B statt auf $B^T B$ durchgeführt. Der folgende Satz wird für die Konvergenz des Verfahrens benötigt.

Satz 2.3.9 *Sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische Matrix mit den Eigenwerten $\lambda_1, \ldots, \lambda_n$, sodass*

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0.$$

Weiterhin seien die $n\times n$-Matrizen $A_k = (a_{ij}^{(k)}), Q_k$ und R_k wie folgt definiert:

1. $A_1 = A$,
2. $A_k = Q_k R_k$, *QR-Zerlegung*,
3. $A_{k+1} = R_k Q_k$.

[42] Dieses Verfahren wird als *Chasing* bezeichnet.

Dann gilt:

$$
\begin{aligned}
&1. \quad \lim_{k\to\infty} Q_k = I, \\
&2. \quad \lim_{k\to\infty} R_k = \Lambda, \\
&3. \quad a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right), \quad \textit{für } i > j.
\end{aligned}
$$

Beweis: Siehe [3] S.142f. □

Nach Satz 2.3.9 gilt

$$B_k^T B_k \to \Lambda = diag(\sigma_1^2, \ldots, \sigma_n^2) = \Sigma^2, \quad \text{für } k \to \infty.$$

Daraus folgt, dass die Folge der B_k gegen die Diagonalmatrix der Singulärwerte von B konvergiert

$$B_k \to \Sigma, \quad \text{für } k \to \infty.$$

Die Konvergenzgeschwindigkeit des Chasing-Verfahrens lässt sich durch die Benutzung einer Shift-Strategie verbessern. Dieser Shift wird in jeder Iteration bei Durchführung der ersten Givens-Rotation angewendet und anschließend das Chasing-Verfahren in der oben beschriebenen Weise fortgesetzt. Im Speziellen wird hier der Wilkinson-Shift verwendet - dieser garantiert Konvergenz und die Konvergenzgeschwindigkeit ist hoch.

Definition 2.3.10 (Wilkinson-Shift)
Sei die Matrix $B \in \mathbb{R}^{n\times n}$ die aktuelle Iterierte und C die untere rechte 2×2-Untermatrix von $B^T B$

$$C = \begin{pmatrix} b_{n-2,n-1}^2 + b_{n-1,n-1}^2 & b_{n-1,n} b_{n-1,n-1} \\ b_{n-1,n} b_{n-1,n-1} & b_{n-1,n}^2 + b_{n,n}^2 \end{pmatrix}. \tag{2.13}$$

Dann ist der Wilkinson-Shift ρ derjenige Eigenwert von (2.13), der näher an $b_{n-1,n}^2 + b_{n,n}^2$ liegt.

Bemerkung 2.3.11 Um den für das Chasing-Verfahrens benötigten Rechenaufwand zu vermindern, wird die sogenannte Deflation angewendet. Sind die ersten bzw. letzten Nebendiagonalelemente von B kleiner als $\sqrt{eps}$, wobei *eps* die Maschinengenauigkeit sei, werden die ersten bzw. letzten Zeilen von B nicht mehr verändert und das Chasing-Verfahren nur noch auf die verbleibende Matrix $\tilde{B}$ angewendet. Nach $n-1$ Deflationen ist das Verfahren beendet. □

Algorithmus 2.3.12 *Pseudoinversenbestimmung mittels Singulärwertzerlegung*

1. *Bringe die Matrix $A \in \mathbb{R}^{n \times n}$ mittels Householder-Transformationen auf Bidiagonalgestalt.*

2. *Wende das Chasing-Verfahren inklusive Shift-Strategie und Deflation auf die Bidiagonalmatrix an.*

3. *Berechne die Pseudoinverse A^+ nach Satz 2.3.4 aus*

$$A^+ = V\Sigma^+U^T,$$

wobei U bzw. V^T das Produkt aller im Zuge der Householder-Reflexionen und Givens-Rotationen von links bzw. rechts an die Matrix A multiplizierter Orthogonalmatrizen ist.

Kapitel 3

Komponenten und Funktionen des Programms

In diesem Kapitel wird zunächst auf die weiteren Komponenten des Programms eingegangen und anschließend die Funktionen und die Bedienung des Programms beschrieben.

Das vorliegende Programm sollte in erster Linie möglichst anwenderfreundlich gestaltet werden. Daher wurde beispielsweise statt einer Konsolenanwendung eine grafische Oberfläche[1] erstellt, um unter anderem zu vermeiden, dass ein Nutzer die Daten des zu lösenden Problems direkt im Quellcode ändern muss. Dies würde Kenntnisse in der verwendeten Programmiersprache C++ sowie eine Rekompilierung des Programms bei jeder Änderung der Problemstellung nach sich ziehen. So ist nun lediglich bei der Installation des Programms eine Kompilierung nötig.[2] Die Erstellung der grafischen Oberfläche mit den Werkzeugen gtkmm und Glade wird im nächsten Abschnitt dargestellt.

Anschließend wird in 3.2 auf die Evaluierung von Strings mit Hilfe eines Parsers eingegangen. Dies ist notwendig, weil die Problemdaten nicht im Quellcode auftauchen, sondern zur Laufzeit aus der grafischen Oberfläche eingelesen werden. Diese Daten liegen dann als Strings vor und müssen folglich in Formeln umgewandelt und ausgewertet werden.

Um die Anwenderfreundlichkeit weiter zu steigern, sollte das Programm über eine breite Palette von Funktionen und Einstellmöglichkeiten verfügen. Daher wurde beispielsweise die grafische Darstellung der Lösung eines Problems direkt im Programm ermöglicht. Außerdem ist es möglich, sofern die entsprechenden LaTeX-Pakete installiert sind, die Daten der Problemstellung mitsamt der Lösung und den Grafiken in ein PDF-Dokument zu exportieren. Diese Funktionen werden in 3.3 und 3.4 näher beschrieben.

[1] engl. graphical user interface, kurz: GUI.

[2] Sofern die Windowsversion verwendet wird, kann man sogar komplett auf eine Kompilierung verzichten. Auf Linuxsystemen muss das Programm unter Umständen einmal kompiliert werden.

Des Weiteren wird in Abschnitt 3.5 die Installation des Programmpakets unter verschiedenen Linux- und Windowssystemen beschrieben. Den Abschluss des Kapitels bildet die Erläuterung der im Programm implementierten Funktionen sowie dessen Bedienung.

3.1 Erstellung der grafische Oberfläche mit gtkmm und Glade

Für die Erstellung der grafischen Oberfläche wurde die freie C++-Programmbibliothek *gtkmm*[3] verwendet. Diese ist eine Anbindung des GIMP-Toolkits an C++. Die gtkmm-Bibliothek steht unter einer freien Lizenz und ist sowohl für Linux- als auch für Windowssysteme erhältlich. Grafische Oberflächen können mit *gtkmm* explizit programmiert oder mit Hilfe des Programms Glade erstellt werden. Da die explizite Programmierung gerade bei GUIs, die viele Elemente enthalten, sehr aufwendig wird, wurde hier das Baukastensystem Glade verwendet. Dieses ermöglicht es, eine GUI ohne Programmierkenntnisse zu erstellen und den zugrunde liegenden *gtkmm*-Quellcode dann in einer *glade*-Datei abzuspeichern. Diese kann dann innerhalb des C++-Codes aufgerufen und die einzelnen in der Datei gespeicherten Elemente angesprochen werden. Zu den wichtigsten hier verwendeten Elementen, sogenannten Widgets, gehören:

- **Gtk::Window** Ein Fenster der grafischen Oberfläche, das mit verschiedenen Widgets gefüllt wird.
- **Gtk::Entry** In diese Textfelder werden alle Eingabedaten des Problems eingetragen und ausgelesen.
- **Gtk::TextView** Mehrzeilige Textfelder, die in diesem Programm beispielsweise zur Ausgabe der Problemstellung und der Lösung verwendet werden.
- **Gtk::Button** Für einen Button wird ein sogenanntes Signal definiert, welches eine Funktion innerhalb des Quellcodes aufruft. In dieser Funktion wird dann die auszuführende Aktion programmiert, beispielsweise das Öffnen oder Schließen eines Fensters oder die Durchführung von Berechnungen.
- **Gtk::RadioButton** Dies ist ein Auswahldialog, wobei von mehreren zu einer Gruppe zusammengefassten RadioButtons stets nur genau einer aktiv sein kann.
- **Gtk::CheckButton** Wird in diesem Widget ein Häkchen gesetzt, ist diese Option ausgewählt, anderenfalls nicht. Im Gegensatz zum RadioButton-Widget ist der CheckButton unabhängig von anderen Widgets

[3] gtkmm = gimp tool kit minus minus.

- **Gtk::Image** Dieses Widget stellt einen Bereich zur Verfügung, in dem Bilder angezeigt werden können.

Am Beispiel des Widgets Gtk::Entry soll nun gezeigt werden, wie auf die Widgets der GUI zugegriffen werden kann.[4]

- Zunächst wird in einer C++-Header-Datei ein Zeiger definiert.

  Gtk::Entry textfeld*

- Anschließend wird diesem Zeiger ein Textfeld aus der *glade*-Datei zugewiesen. Dabei ist *refXml* ein Zeiger auf die *glade*-Datei.

  refXml→get_widget("Name des Textfeldes in der *glade*-Datei", textfeld)

- Nun kann mittels

  textfeld→set_text(data)

  der Wert der Variablen *data* in das Textfeld geschrieben werden oder mit dem Befehl

  data=textfeld→get_text()

  der Inhalt des Textfeldes eingelesen werden. Es können beliebige Variablen in das Textfeld geschrieben werden, also z.B. Strings, Double- oder Integervariablen. Beim Einlesen hingegen liegt der Eintrag des Textfeldes als String vor. Demzufolge müssen diese z.B. in eine Doublevariable umgewandelt werden, wenn die eingelesene Größe eine Zahl ist.[5] Soll die eingelesene Größe hingegen als Formel vorliegen, muss der vorliegende String mit Hilfe eines Parsers ausgewertet werden.

3.2 Evaluierung von Strings mit muParser

Wie bereits erwähnt, werden die auszuwertenden Formeln zugunsten der Anwenderfreundlichkeit direkt aus der grafischen Oberfläche eingelesen. Dies hat den Vorteil, dass das Programm nicht nach jeder Änderung neu kompiliert werden muss, birgt allerdings auch den Nachteil, dass die als Strings vorliegenden Gleichungen des zu lösenden Problems nicht ohne weiteres ausgewertet werden können. Um diese Strings als Formeln behandeln zu können, benötigt man einen sogenannten Parser. Dieser ist in der Lage, vorgegebene Zeichenketten innerhalb eines Strings durch Zahlenwerte zu ersetzen und die somit entstehende Formel auszuwerten. Hier wird der frei verfügbare Parser *muParser*[6] für diese Aufgabe genutzt.

Der Parser wird für die folgenden Eingabedaten verwendet:

[4] Für weitere Details, ausführliche Dokumentationen und Beispiele sei hier auf die Webseiten glade.gnome.org und www.gtkmm.org verwiesen.
[5] Mit Stringstreams ist diese Umwandlung problemlos möglich
[6] Homepage: muparser.sourceforge.net.

- Differenzialgleichung(en)
- Steuerung(en)
- Rand- und Schaltbedingung(en)
- Sprünge
- bekannte Konstanten
- Startschätzung

Während die ersten 4 Punkte unabdingbar sind, da diese Daten Gleichungen darstellen und Variablen beinhalten, ist die Verwendung sowohl bei der Auswertung der gegebenen Konstanten als auch der Startschätzungen nicht zwingend notwendig. Der Parser wird allerdings auch hier benutzt, da dies den Vorteil mit sich bringt, dass dann nicht nur Zahlen sondern auch Terme angegeben werden können, beispielsweise Brüche oder trigonometrische Funktionen.

Die Tabelle 3.1 zeigt die in *muParser* implementierten mathematischen Funktionen. Dabei akzeptieren die letzten 4 Funktionen eine beliebige Zahl von Argumenten, während die anderen Funktionen jeweils genau ein Argument erfordern. Das bzw. die Argumente sind in Klammern hinter die jeweilige Funktion zu setzen. Bei mehreren Argumenten werden diese durch Kommata voneinander getrennt. Außerdem werden die gängigsten binären Operatoren unterstützt – diese kann man der Tabelle 3.2 entnehmen.

Der Parser kann sowohl durch Einbindung einer Bibliothek als auch durch direkte Einbindung diverser C++-Dateien in das Projekt verwendet werden. Um den Parser betriebssystemunabhängig verwenden zu können, wurde die zweite Variante gewählt. Diese Variante hat den Vorteil, dass man auf verschiedenen Systemen lediglich das Programm kompilieren muss und nicht zusätzlich auch die *muParser*-Bibliothek.

Der im Listing 3.1 dargestellte Quellcode zeigt ein kleines Beispiel zur Verwendung des Parsers.

Listing 3.1: Beispiel *muParser*

```
1     #include "muParser.h"
2     using namespace mu;
3
4     int main()
5     {
6         //gesucht: Funktionswert von f an der Stelle (1,2)
7         double xwert = 1, ywert = 2;
8         string f = "x^2/(y-1)";
9         double fwert;
10
11        parser p;
12        p.DefineVar("x", &xwert);
13        p.DefineVar("y", &ywert);
14        p.SetExpr(f);
15        fwert = p.Eval();
16
17        return 0;
18    }
```

Name	Erklärung
sin	Sinus
cos	Kosinus
tan	Tangens
asin	Arkussinus
acos	Arkuskosinus
atan	Arkustangens
sinh	Sinus Hyperbolicus
cosh	Kosinus Hyperbolicus
tanh	Tangens Hyperbolicus
asinh	Arkussinus Hyperbolicus
acosh	Arkuskosinus Hyperbolicus
atanh	Arkustangens Hyperbolicus
*log*2	Logarithmus zur Basis 2
*log*10 bzw. *log*	Logarithmus zur Basis 10
ln	natürlicher Logarithmus
exp	Exponentialfunktion e^x
sqrt	Quadratwurzel
sign	Signumfunktion
rint	Rundung auf die nähere ganze Zahl
abs	Betrag
min	Minimum aller Argumente
max	Maximum aller Argumente
sum	Summe aller Argumente
avg	arithmetisches Mittel aller Argumente

Tabelle 3.1: muParser: mathematische Funktionen

Gesucht ist hier der Funktionswert einer Funktion $f(x, y)$ an der Stelle $(1, 2)$. Dazu benötigt man eine *string*-Variable f, die die Funktionsvorschrift beinhaltet, zwei *double*-Größen $xwert$ und $ywert$, denen die Werte 1 bzw. 2 zugewiesen werden, sowie eine weitere Variable $fwert$, ebenfalls vom Typ *double*, der der Funktionswert $f(1, 2)$ zugewiesen werden wird. Bei der Verwendung müssen die folgenden Dinge beachtet werden:

- Alle Quellcode-Dateien von *muParser* müssen entweder bereits vor der Erstellung des Programms kompiliert werden, also die Objektdateien erzeugt worden sein, oder man bindet die Kompilierung dieser Dateien direkt in ein Makefile ein, sodass diese bei der Erstellung des Programms mit durchgeführt wird.

- Die Header-Datei "muParser.h" muss eingebunden werden.

- Weiterhin müssen die *muParser*-spezifischen Befehle mit vollständigem Namen angegeben oder mittels des Befehls "using namespace mu;" bekannt gemacht werden. Im ersten Fall muss den entsprechenden Befehlen "mu::" vorangestellt werden.

Operator	Bedeutung	Priorität
=	Zuweisung	-1
&&	logisches und	2
||	logisches oder	4
<=	kleiner gleich	4
>=	größer gleich	4
! =	ungleich	4
==	gleich	4
>	größer gleich	4
<	kleiner gleich	4
+	Addition	5
−	Subtraktion	5
∗	Multiplikation	6
/	Division	6
ˆ	Potenz xˆy	7

Tabelle 3.2: muParser: Operatoren

Nun ist es möglich, einen Parser mittels "parser p", wobei p hier der Name des Parsers sei, zu definieren. Anschließend werden über den Befehl "p.DefineVar("x", &xwert)" verschiedene Variablen des Parsers definiert. Dabei ist x der Name der Variablen im zu evaluierenden Ausdruck und "&xwert" ein Zeiger auf diejenige C++-Variable, dessen Wert der Variablen im Ausdruck zugewiesen werden soll. Mit "p.SetExpr(f)" weist man dem Parser den auszuwertenden Ausdruck zu, in diesem Fall die Funktion f. Daraufhin wertet man mit Hilfe des Befehls "fwert=p.Eval()" den Ausdruck aus und der Variablen $fwert$ vom Typ *double* wird das Ergebnis der Auswertung zugewiesen.

Für weitere Details bezüglich des Funktionsumfangs und der Möglichkeiten, die dieser Parser bietet, sei nochmals auf die Webseite des Parsers verwiesen. Es ist beispielsweise möglich, auch eigene Funktionen und Operationen zu definieren. Außerdem kann man den Parser beim Auftreten eines Fehlers detaillierte Angaben zum Fehler ausgeben lassen.

Bemerkung 3.2.1 Anzumerken ist jedoch, dass der Parser zwar in der Lage ist, sehr schnell viele Formeln auszuwerten[7], die Auswertungen aber trotz allem deutlich mehr Zeit benötigen als die Auswertung einer direkt im Quellcode implementierten Formel. Daher ist es sehr wichtig, dass man in den verwendeten Verfahren versucht, die Anzahl der Funktionsauswertungen so gering wie möglich zu halten.[8] □

[7] Vergleiche dazu den Benchmark auf der Homepage des Parsers.

[8] Unter anderem fiel daher beim Rosenbrock-Typ-Verfahren die Wahl des Koeffizientensatzes auf denjenigen von Shampine. Dieser benötigt zwei identische Funktionsauswertungen, wodurch man in jedem Zeitschritt eine Auswertung einsparen kann. Weiterhin wird bei der Berechnung der Ableitungen geprüft, ob gegebenenfalls einige Ableitungen identisch Null sind, um unnötige Funktionsauswertungen zu vermeiden. Vergleiche zum

3.3 Erstellen von Grafiken mit gnuplot

Im vorliegenden Programm ist die Möglichkeit implementiert, die Lösung eines Problems grafisch darstellen zu lassen. Diese Plots werden mit dem Programm *gnuplot* erstellt und direkt im Programmfenster angezeigt. Dabei stehen dem Nutzer im Wesentlichen zwei verschiedene Optionen zur Erstellung der Graphen zur Verfügung:

1. Es kann die auf der x-Achse dargestellte Variable gewählt werden. Dies sind die Zeitvariable x sowie die Zustandsvariablen $y_i,\ i = 1, \ldots, n$. Dabei sei n die Dimension des gelösten Problems. Bei Wahl dieser Option werden n Grafiken erstellt, wobei bis auf die gewählte Variable alle Variablen in Abhängigkeit zu dieser Variablen geplottet werden.

2. Die zweite Option bietet die Möglichkeit, bis zu zehn Grafiken zu erstellen, die jeweils bis zu drei beliebige Subplots enthalten. Dabei ist es nicht nur möglich, einzelne Variablen plotten zu lassen, sondern beliebige Funktionen, da diese mit *muParser*[9] ausgewertet werden. So kann man beispielsweise Zustands- und Steuerbeschränkungen darstellen lassen, um zu sehen, ob diese auf dem ganzen Intervall eingehalten wurde.

Im folgenden wird beschrieben, wie die Grafiken mit Hilfe von *gnuplot* erstellt werden. Dabei wird vorausgesetzt, dass die Mehrzielmthode erfolgreich gestartet wurde.[10]

1. Um zu verhindern, dass trotz einer fehlgeschlagenen Erstellung der Grafiken alte Plots eines vorher gelösten Problems angezeigt werden, wird der Unterordner *plots* im Programmverzeichnis zunächst geleert.

2. Die Werte der Zustandsvariablen und Steuerungen an den Mehrzielknoten und den Schaltpunkten werden in einem *double*-Feld gespeichert. Diese Werte stammen aus der von der Mehrzielmethode gelieferten Lösung des Problems.

3. In jedem Intervall wird ein Anfangswertproblem gelöst, um auch an Zwischenstellen Werte für die Zustandsvariablen und Steuerungen zu erhalten. Dazu werden spezielle Versionen der Anfangswertproblemlöser verwendet, die die Werte jedes Zeitschritts wiederum in einem Feld vom Typ *double* abspeichern.[11] Um sicherzustellen, dass die entstehenden

Rosenbrock-Typ-Verfahren den Abschnitt 2.2.2.

[9] Vergleiche dazu den vorigen Abschnitt. Mit *muParser* ist es möglich, Strings in Formeln umzuwandeln und diese dann zu evaluieren.

[10] D.h., die auf S.73 beschriebenen Tests lieferten keinen Fehler. Wurde keine Lösung des Problems gefunden, kann die Funktion insbesondere dazu genutzt werden, Anhaltspunkte zu finden, wo Probleme bei der Lösung des Problems aufgetreten sind.

[11] Wurde für die Lösung des Problems eines der *Dormand-Prince*-Verfahren benutzt, so werden die Anfangswertprobleme mit einer modifizierten Version von *Dormand-Prince 5(4)* gelöst, anderenfalls wird eine abgeänderte Version des Rosenbrock-Typ-Verfahrens benutzt.

Plots glatte Kurven darstellen, wird die verwendete Schrittweite nach oben beschränkt, sodass in jedem Intervall mindestens zehn Zeitpunkte mit den dazugehörigen Werten erhalten werden.

4. Nun wird aus der grafischen Oberfläche ermittelt, welche und wie viele Plots erstellt werden sollen und die Daten in einem Vektorfeld vom Typ *string* gespeichert.

5. Anschließend werden die ermittelten Daten dazu verwendet, die zu plottenden Mehrzielknoten, Schaltpunkte und Zwischenpunkte samt den zugehörigen Werten der Zeitvariablen, der Zustands- und Steuervariablen sowie den Werten der ausgewerteten Formeln, sofern von der Möglichkeit beliebiger Plots Gebrauch gemacht wurde, in verschiedenen Textdateien zu speichern.

6. Eine *gnuplot*-Datei wird geschrieben und über einen Konsolenbefehl ausgeführt, die mittels Zugriff auf die im vorhergehenden Schritt erstellten Textdateien die gewünschten Grafiken im *png*-Format erstellt. Dabei werden die Mehrzielknoten und die Schaltpunkte als Punkte geplottet, während die Zwischenpunkte als Kurven dargestellt werden. Enthält eine Grafik mehr als eine Kurve, werden diese in verschiedenen Farben geplottet und außerdem erhält jede Grafik eine Legende. Abschließend wird die erste Grafik im Programm angezeigt. Durch Klicken auf die entsprechenden Buttons kann der Nutzer die weiteren Grafiken anzeigen lassen.

Zur Verwendung bzw. Installation der Erstellung von Grafiken mittels *gnuplot* auf Linux- bzw. Windowssystemen vergleiche die Installationsanweisungen im Abschnitt 3.5.

3.4 Export von Problemstellungen mit pdfLaTeX

Dem Benutzer wird im vorliegenden Programm die Möglichkeit geboten, die Eingabedaten und die Lösung eines Optimalsteuerungsproblems in ein PDF-Dokument zu exportieren. Dazu wird auf Knopfdruck automatisch eine *tex*-Datei erstellt. Anschließend wird diese über einen Konsolenbefehl mit pdfLaTeX kompiliert. Das auf diese Weise erstellte PDF wird unter dem Namen des Projektes im Unterverzeichnis "pdf" abgespeichert. Um dieses Verzeichnis nicht mit Dateien zu überladen, werden nach der Erstellung alle bei der Kompilierung erstellten Dateien automatisch gelöscht. Zur Verfügbarkeit und Installation dieser Exportfunktion vergleiche die Installationsanweisungen im Abschnitt 3.5.

Das Dokument enthält alle relevanten Daten des Problems. Dies sind im Einzelnen:

- Name des Projektes
- Eingabedaten:
  - Anzahl der Differenzialgleichungen, Konstanten, Mehrzielknoten und Schaltpunkte
  - Differenzialgleichungen
  - Steuerungen
  - Rand- und Schaltbedingungen sowie die Sprünge
  - Startschätzungen der Zustandsvariablen, Konstanten und Schaltpunkte
- Daten aus den Newtoniterationen:
  - Iterationsnummer
  - Relaxationsfaktor
  - Ergebnis des Tests auf das Broyden-Update für die nächste Iteration
  - Abbruchkriterien
- Lösungsdaten:
  - Werte der Zustandsvariablen an allen Mehrzielknoten sowie der Konstanten und Schaltpunkte
  - Grafiken, sofern diese vor dem PDF-Export erstellt wurden.

Um die Lesbarkeit des Dokuments zu verbessern, werden alle Daten, die Variablen des Problems beinhalten[12], nicht in der Syntax des Programms dargestellt, sondern aufwendig umgewandelt und im Mathematikmodus in die *tex*-Datei eingebunden.

3.5 Installation

Im Folgenden wird die Installation auf ausgewählten Linux- und Windowssystemen beschrieben. Die für den Parser *muParser*[13] benötigten Dateien sind bereits im Programmverzeichnis enthalten, müssen demzufolge nicht hinzugefügt werden.

[12] Dies betrifft die Differenzialgleichungen, Steuerungen, Rand- und Schaltbedingungen, Sprünge sowie die Angabe von Variablennamen in der Startschätzung und der Lösung.
[13] Siehe Abschnitt 3.2.

3.5.1 Linux

Die Installation des Programms wird beispielhaft anhand der beiden Linux-Distributionen *openSuse* und *Ubuntu* beschrieben. Die Linux-Version enthält sowohl den Quellcode als auch eine unter *openSuse 11.3 64-Bit* kompilierte Variante des Programms.

Ubuntu 10.10 64-Bit

Bei der Installation sind – ausgehend von einer Standardinstallation – mit der Synaptic-Paketverwaltung die folgenden Pakete nachträglich zu installieren sowie die Abhängigkeiten aufzulösen:

- Zunächst wird die Installation des Paketes "libgladem-2.4-dev" vorausgesetzt. Dies wird sowohl benötigt, um die kompilierte Variante starten zu können als auch um den Quellcode des Programms kompilieren zu können. Das Programm kann mittels Ausführung des im Programmverzeichnis bereitgestellten *Makefiles* kompiliert werden.
- Um die Erstellung der Plots zu ermöglichen, ist das Paket "gnuplot" zu installieren.
- Um die Funktion des Exports in ein PDF-Dokument nutzen zu können, wird das Paket "texlive" benötigt. Dieses stellt unter anderem den Befehl *pdflatex* zur Verfügung, welcher zur PDF-Erstellung genutzt wird.

Sind alle Pakete installiert sowie das Programm gegebenenfalls kompiliert worden, kann dieses durch Ausführung der Datei "optcontrolsolver_linux" sowohl über die Konsole als auch einen Dateibrowser gestartet werden.

openSuse 11.4 64-Bit (KDE)

Wird diese Distribution verwendet, sind mittels *YaST* folgende Pakete zu installieren:

- Zunächst ist das Paket "libgladem-2_4-1" zu installieren, welches zur Laufzeit des Programms benötigt wird.
- Für die erfolgreiche Kompilierung des Programms müssen die folgenden Pakete installiert sein:
  - *make*
  - *gtkmm2-devel*
  - *libgladem-devel*
  - *gcc-c++*

  Das Programm kann mittels Ausführung des im Programmverzeichnis bereitgestellten *Makefiles* kompiliert werden.
- Soll die Funktion zur Erstellung von Plots verfügbar sein, so ist das Paket "gnuplot" zu installieren.

- Um die Funktion des Exports in ein PDF-Dokument nutzen zu können, werden die Pakete "texlive" sowie "texlive-latex" benötigt. Diese stellen unter anderem den Befehl *pdflatex* zur Verfügung, welcher zur PDF-Erstellung genutzt wird.

3.5.2 Windows

Bei der Windowsversion des vorliegenden Programms wird – im Gegensatz zur Linuxversion – nicht der Quellcode, sondern lediglich das kompilierte Programm zur Verfügung gestellt. Dies liegt zum Einen daran, dass die Kompilierung aufgrund der nötigen Einbindung der Bibliotheken für die grafische Oberfläche relativ kompliziert ist und zum Anderen daran, dass das Programm auch ohne Neukompilierung auf jedem gängigen Windowssystem lauffähig ist. Die vorliegende Version wurde unter Windows XP mit Microsoft Visual C++ 2008 Express Edition und *gtkmm* in der Version 2.22 kompiliert.

Unterstützte Windows-Versionen:[14]

- Windows XP
- Windows Vista (32 und 64-Bit)
- Windows 7 (32 und 64-Bit)

Installation:

- Zunächst müssen die *gtkmm*-Bibliotheken installiert werden. Diese findet man auf der *gtkmm*-Homepage: "www.gtkmm.org". Es ist ausreichend, die Laufzeitbibliotheken zu installieren[15], da das Programm nicht kompiliert werden muss. Zu beachten ist, dass die 32-Bit-Version von *gtkmm* installiert werden muss, da es sich bei dem vorliegenden Programm um eine 32-Bit-Anwendung handelt.
- Auf der beiliegenden CD befindet sich im Verzeichnis "Windows" der Ordner "optcontrolsolver". Dieser enthält alle benötigten Programmdateien und ist an eine beliebige Stelle zu kopieren.
- Die Unterstützung für die Erstellung von Plots ist bereits gegeben, da die entsprechenden *gnuplot*-Dateien bereits im Programmverzeichnis enthalten sind.

[14] Bei Windows Vista und 7 wurde die Funktionalität des Programms nur unter 64-Bit getestet. Da es sich um ein auf einem 32-Bit-System kompiliertes Programm handelt, kann davon ausgegangen werden, dass dieses auch auf den 32-Bit-Versionen dieser Windowsversionen lauffähig ist.

[15] Bei der Installation kann ausgewählt werden, ob lediglich eine Installation der Laufzeitbibliotheken oder einer Vollinstallation durchgeführt werden soll.

- Um die Funktion für den PDF-Export nutzen zu können, ist das LaTeX-Paket *MiKTeX* zu installieren.[16] Anschließend muss die Datei "pdflatex" in das Programmverzeichnis kopiert werden. Diese befindet sich in "$Installationspfad MiKTeX$\miktex\bin".

- Das Programm wird durch Öffnen der Datei "optcontrolsolver.exe" gestartet.

Bemerkung 3.5.1 (Funktionalität) Das Programm wurde in erster Linie für Linuxsysteme entwickelt. Für die Windowsversion sind lediglich einige notwendige Anpassungen im Quellcode vorgenommen worden ohne dass die Funktionalität des Programms gründlich überprüft wurde. Daher kann nicht garantiert werden, dass diese Version ähnlich fehlerfrei läuft wie die Linuxversion. □

Bemerkung 3.5.2 (Implizite Angabe der Steuerung) Soll die Steuerung implizit angegeben werden, so wird diese in der C++-Datei "control.cc" angegeben bzw. ein Algorithmus zur Berechnung der Steuerung eingefügt. Daher ist es notwendig, das Programm erneut zu kompilieren. Da die Windowsversion lediglich fertig kompiliert bereitgestellt wird, ist hier die implizite Angabe der Steuerung nicht möglich. □

3.6 Funktionen und Bedienung

In diesem Abschnitt wird beschrieben, welche Funktionen das Programm im Einzelnen enthält und wie es bedient wird. Dazu wird zunächst die Eingabe der Projektdaten und im Anschluss die Speicherung bzw. das Laden der Daten eines Projekts näher erläutert. Die dem Anwender bei der Nutzung der Mehrzielmethode zur Verfügung stehenden Optionen sind, nachdem diese bereits bei der Behandlung der einzelnen Verfahren erwähnt wurden, Gegenstand des dritten Teils dieses Abschnitts. Anschließend wird auf die Lösung eines Problems sowie die Verarbeitung der Lösung durch eine grafische Darstellung und den Export in ein PDF-Dokument eingegangen. Der letzte Teil dieses Abschnitts beschäftigt sich mit der Fehlerbehandlung. Dort wird beschrieben, welche Probleme bei der Lösung eines Optimalsteuerungsproblems auftreten können und wie man mit diesen umgehen kann.

3.6.1 Eingabe der Projektdaten

Wird das Programm gestartet[17], so öffnet sich das Hauptfenster (Abb. 3.2). Dieses besteht aus 3 Karteikartenreitern, wobei der erste Tab Angaben zum

[16] Die Installation der Basis-Version ist dabei ausreichend.
[17] Zur Installation und Ausführung des Programms auf verschiedenen Systemen siehe 3.5.

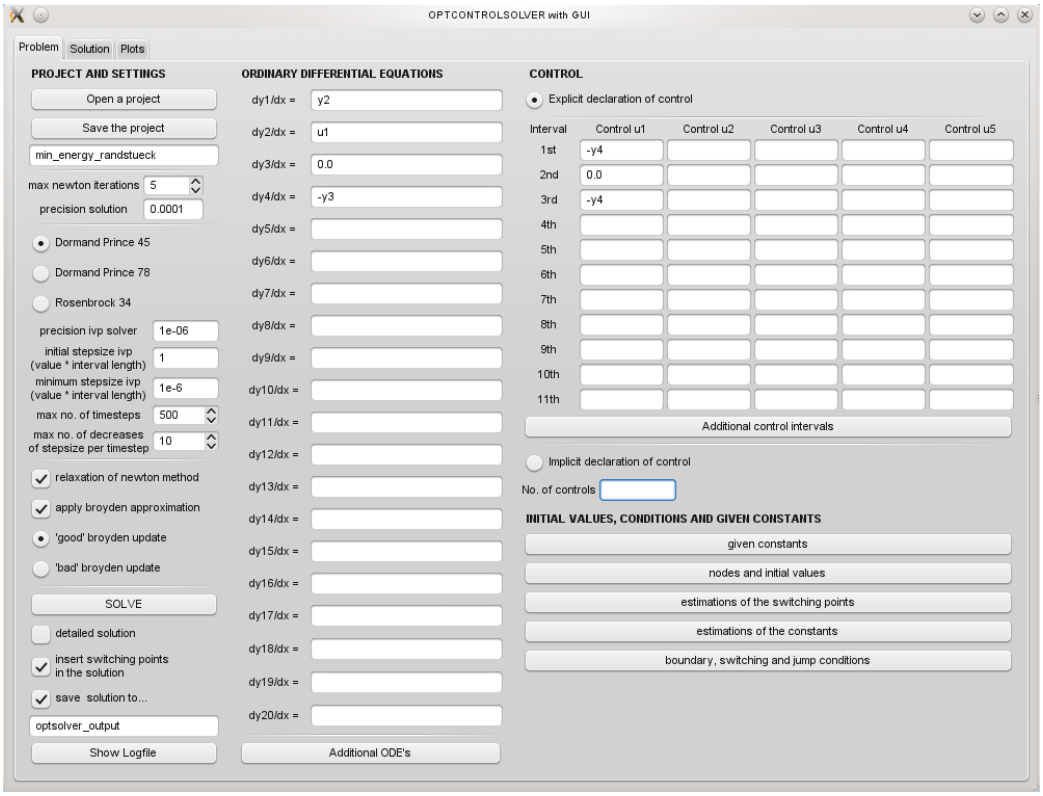


Abbildung 3.2: Hauptfenster

Problem und zu den Optionen der Mehrzielmethode enthält. Hier werden die Differenzialgleichungen (ORDINARY DIFFERENTIAL EQUATIONS) und die Steuerungen (CONTROL) des zu lösenden Problems angegeben. Bei den Steuerungen ist zu beachten, dass diese sowohl explizit in der entsprechenden Tabelle als auch implizit angegeben werden können. Weiterhin können auf Knopfdruck die entsprechenden Fenster für die anderen Eingabedaten angezeigt werden (INITIAL VALUES AND CONDITIONS), unter anderem das Fenster für die Rand- und Schaltbedingungen sowie die Sprünge (Abb. 3.3) als auch das Fenster für die Definition von gegebenen Konstanten (Abb. 3.4).

Bei der Eingabe dieser Daten ist die folgende Syntax zu beachten:

- Die Anzahl der Zustandsvariablen, adjungierten Variablen und Steuerungen wird nicht explizit angegeben. Die Anzahl der Variablen wird über die Anzahl der Differenzialgleichungen ermittelt und analog die Dimension der Steuerung über die eingegebenen Steuerungen in der entsprechenden Tabelle.

- Zu beachten ist, dass bei der Angabe der Daten keine Zeilen freigelassen werden dürfen. Alle nach einer leeren Zeile folgenden Einträge werden nicht berücksichtigt.

- Die Syntax von *muParser* ist bei allen Angaben einzuhalten, da die

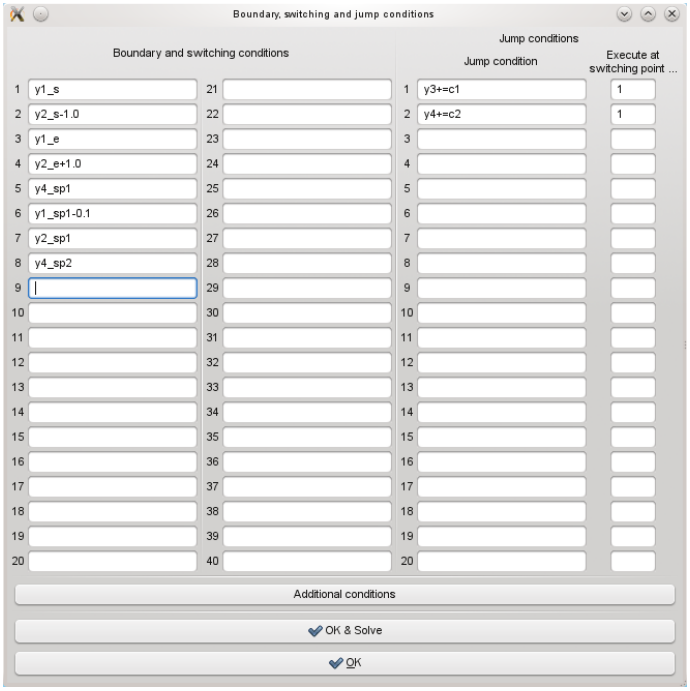


Abbildung 3.3: Eingabe der Bedingungen und Sprünge

eingelesenen Formeln mit diesem Parser ausgewertet werden.[18]

- Die Zustandsvariablen sind fortlaufend in der Form $yi,\ i = 1, \ldots, n$, wobei n die Anzahl der Differenzialgleichungen sei, anzugeben.

- Analog werden die Steuerungen mit $ui,\ i = 1, \ldots, m$, die mit zu optimierenden Konstanten mit $ci,\ i = 1, \ldots, k$ sowie die bekannten Konstanten mit $pi,\ i = 1, \ldots, l$ bezeichnet. Dabei sei m die Anzahl der Steuerungen, k die Anzahl der Konstanten und l die Anzahl der gegebenen Konstanten[19].

- Die Rand- und Schaltbedingungen müssen in die Form $r_i(\cdot) = 0$ gebracht werden, wobei im Programm lediglich $r_i(\cdot)$ angegeben wird. Die i-te Variable erhält, je nachdem an welchem Zeitpunkt der Wert der Variablen eine Bedingung erfüllen muss, verschiedene Anhänge:

  - Anfangswert: yi_s
  - Endwert: yi_e
  - im j-ten Schaltpunkt: yi_spj

- Enthält das zu lösende Problem Sprünge, so sind diese in der Form $yi{+}{=}\ cj$ anzugeben. Dabei ist yi die Variable, die einen Sprung um den Wert cj aufweist. Außerdem ist anzugeben, an welchem Schaltpunkt dieser Sprung auszuführen ist. Dazu wird einfach die Nummer

[18] Vergleiche dazu Abschnitt 3.2.
[19] Vergleiche dazu Bemerkung 3.6.1.

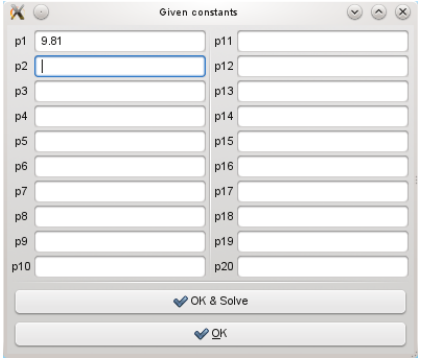


Abbildung 3.4: Definition von gegebenen Konstanten

dieses Schaltpunktes in das entsprechende Feld "Execute at switching point..."[20] eingetragen.

Bemerkung 3.6.1 (bekannte Konstanten) Optimalsteuerungsprobleme enthalten oftmals gegebene Konstanten, also solche, deren Wert bekannt ist. Diese können beispielsweise physikalischer Natur, wie die Gravitationskonstante, oder auch Wachstumskonstanten in biologischen Prozessen sein. Um einem Nutzer die Eingabe der Projektdaten zu erleichtern, können solche Konstanten definiert werden, um statt des Wertes der Konstanten lediglich den Namen der Konstanten eingeben zu müssen. Dabei dürfen diese Konstanten in den folgenden Größen verwendet werden:

- Differenzialgleichungen
- Steuerungen
- Anfangs- und Endbedingungen
- Schaltbedingungen
- Startschätzungen der Zustands- und adjungierten Variablen

Die Werte der Konstanten werden wiederum mit *muParser* ausgewertet, sodass nicht nur Zahlenwerte, sondern auch ganze Terme, die allerdings keine anderen Variablen beinhalten dürfen, angegeben werden können. Die gegebenen Konstanten sind streng von den anderen Konstanten zu unterscheiden. Während bei Ersteren deren Wert bekannt ist, ist bei Letzteren lediglich bekannt, dass sie einen konstanten Wert annehmen, da sie beispielsweise Sprünge oder die Lösung einer Differenzialgleichung, deren rechte Seite identisch Null ist, darstellen. Der Wert dieser Konstanten wird erst durch die Mehrzielmethode bestimmt. □

Die Mehrzielknoten sowie die Startschätzungen für die Zustands- und adjungierten Variablen werden im Fenster "enter nodes and initial values" (Abb. 3.5) eingetragen. Dieses wird durch einen Klick auf den Button "nodes and initial values" im Hauptfenster (Abb. 3.2) geöffnet. Die Anzahl der

[20] Siehe Abb. 3.3.

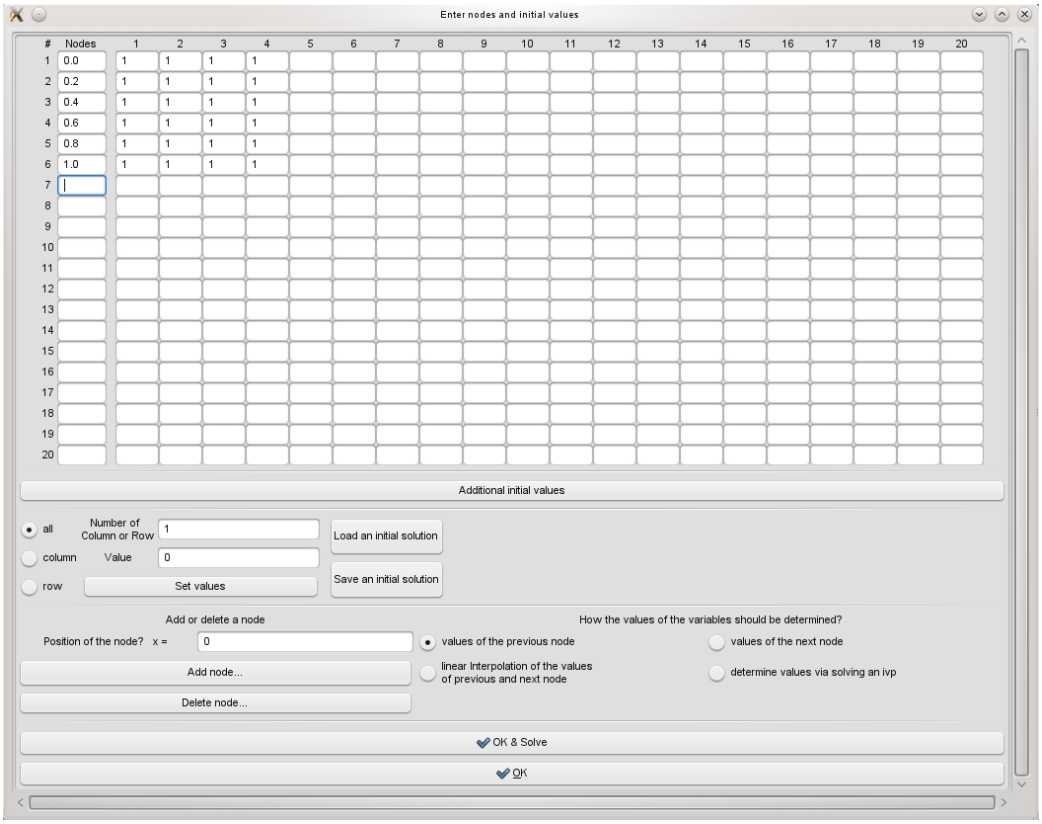


Abbildung 3.5: Fenster: Eingabe der Startschätzungen

Mehrzielknoten wird durch Abfrage der hier eingegebenen Anzahl von Knoten bestimmt. Um bei großen Problemen mit vielen Knoten und/oder Variablen die Eingabe der Startschätzungen zu erleichtern, können ganze Zeilen, Spalten oder auch alle Felder auf Knopfdruck mit einem bestimmten Wert versehen werden, ohne dass der Wert in jedes Feld einzeln eingetragen werden muss. Außerdem besteht die Möglichkeit, eine Startschätzung in einer Datei abzuspeichern und diese auch wieder zu laden. Diese Lösungen befinden sich standardmäßig im Unterordner "solutions" des Programmverzeichnisses, können aber in/aus jedem beliebigen Ordner gespeichert/-geladen werden. Häufig ist es problematisch, sofort das komplexe Mehrpunktrandwertproblem lösen zu wollen, wenn über das Verhalten der Lösung nichts bekannt ist. In solchen Fällen müssen die Startwerte geraten werden und stellen demzufolge in der Regel eine schlechte Näherung dar. Um eine bessere Startnäherung zu erhalten, wird man zunächst ein vereinfachtes Problem, beispielsweise ohne Zustands- und/oder Steuerbeschränkungen, lösen. Die bestimmte Lösung kann dann einfach geladen werden, um sie als Startlösung für das ursprüngliche Problem zu verwenden. Das Laden von Lösungen kann ebenfalls sehr sinnvoll sein, wenn man nicht weiß, ob und wie viele Schaltpunkte die optimale Lösung enthält. Auch hier löst man das Problem zunächst ohne Schaltpunkte und kann dann anhand der Lösung erkennen, ob eine Beschränkung verletzt ist und somit Schaltpunkte eingefügt werden müssen. Ist dies der Fall, kann durch das Laden der Lösung des un-

beschränkten Problems meist auf eine gute Startnäherung für das Problem mit Schaltpunkten zurückgegriffen werden.

Weiterhin können Mehrzielknoten entfernt oder hinzugefügt werden. Die Position des Knotens kann auf zwei Arten bestimmt bzw. angegeben werden. Entweder wird die Position des Knotens direkt angegeben oder durch Klick in einen Plot bestimmt. Bei letzterer Variante ist zu beachten, dass das behandelte Problem bereits gelöst sowie ein Plot mit der Zeitvariablen auf der x-Achse erstellt worden sein muss. Beim Entfernen wird dann derjenige Knoten entfernt, der der angegebenen Position am nächsten liegt. Soll stattdessen ein Knoten hinzugefügt werden, kann der Nutzer bei der Zuweisung der Werte der Variablen am entsprechenden neuen Knoten zwischen 4 Varianten wählen:

1. Die Werte im neuen Knoten erhalten die Werte des vorigen Knotens.
2. Die Werte im neuen Knoten erhalten die Werte des nachfolgenden Knotens.
3. Zur Bestimmung der Werte im neuen Knoten werden die Werte am vorigen und nachfolgenden Knoten linear interpoliert.
4. Es wird ein Anfangswertproblem vom vorigen Knoten bis zum neuen Knoten mit den Werten des vorigen Knotens als Anfangswerte gelöst. Die Lösung sind dann die Werte des neuen Knotens. Dazu wird das Verfahren verwendet, welches im Hauptfenster (Abb. 3.2) aktiviert ist.[21]

Da sowohl die Mehrzielknoten als auch die Startschätzungen mittels *muParser* eingelesen werden, ist es möglich, auch Terme anzugeben, beispielsweise Brüche oder trigonometrische Funktionen. Das Hinzufügen von Mehrzielknoten kann sinnvoll verwendet werden, wenn die Mehrzielmethode das zu lösende Problem nicht auf dem gesamten Zeitintervall lösen kann. Tritt dieser Fall ein, so kann man versuchen, das Problem zunächst auf einem kleineren Zeitintervall lösen. Anschließend wird das Zeitintervall durch Hinzufügen zusätzlicher Knoten vergrößert. Dabei sollte die Lösung des bereits gelösten Problems als Startnäherung verwendet werden.

Die Startschätzungen der Konstanten und Schaltpunkte sind in dem entsprechenden Fenster für die Konstanten (Abb. 3.6) bzw. für die Schaltpunkte (Abb. 3.7) einzugeben. Auch diese können über das Hauptfenster aufgerufen werden. Die Anzahl der jeweiligen Variablen wird wiederum nicht explizit angegeben, sondern indirekt über die Anzahl der eingetragenen Startschätzungen. Hierbei ist zu beachten, dass die Anzahl der Steuerungsintervalle nicht über die Tabelle mit den Steuerungen, sondern über die Anzahl der Schaltpunkte ermittelt wird. Auch diese Werte werden mittels *muParser* eingelesen und ausgewertet.

Wie bereits anfangs dieses Abschnitts erwähnt wurde, können die Steuerungen auch implizit angegeben werden. Dies ist nützlich, wenn sich die

[21] Vergleiche zu den Verfahren zur Lösung von Anfangswertproblem Abschnitt 2.2.

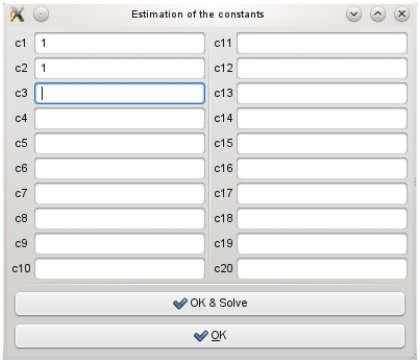


Abbildung 3.6: Konstanten

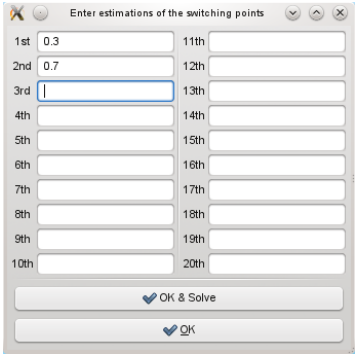


Abbildung 3.7: Schaltpunkte

Steuerungen aus dem Maximumprinzip nicht explizit bestimmen lassen, sondern durch andere Verfahren numerisch bestimmt werden müssen. In diesem Fall kann man im Hauptfenster den Menüpunkt "implicit declaration of control" anwählen und die Dimension der Steuerung angeben. Im Programmverzeichnis befindet sich eine C++-Datei "control.cc", die eine Funktion zur Bestimmung der Steuerungen zur Verfügung stellt:

Listing 3.8: Definition der Steuerung in "control.cc"

```
1  #include "main_window.h"
2  #include <iostream>
3  #include <string>
4  #include <vector>
5  #include "muParser/muParser.h"
6
7  using namespace std;
8  using namespace mu;
9
10 void main_window::calculate_control_implicit(int interval, double
       input_x, vector<double> input_y, vector<double> input_const,
       vector<double>& output)
11 {
12     /*****************************************************************
13     Input-Variablen:    interval     = aktuelles Schaltintervall
           (0,...,no_sp)
14                 input_x      = aktueller Zeitpunkt
15                 input_y      = Zustandsvariablen und Adjungierte am
                       aktuellen Zeitpunkt (Indizes 0,...,no_ode)
16                 input_const = Werte der zu optimierenden Konstanten (
                       Indizes 0,...,no_const)
```

```
17      Output-Variablen:   output       = ausgewertete Steuerung im
            aktuellen Zeitpunkt (Indizes 0,...,no_control)
18      ***************************************************************/
19
20      //hier muss der Algorithmus zur impliziten Berechnung der
            Steuerung angegeben werden.
21
22  }
```

In dieser Funktion kann ein Algorithmus zur Bestimmung der Steuerung angegeben werden. Die Eingabe- und Ausgabevariable der Funktion können im Listing 3.8 nachgelesen werden. Da im Fall der impliziten Angabe der Steuerung die C++-Datei "control.cc" verändert wird, ist anschließend eine Neukompilierung des Programms notwendig. Daher ist diese Funktion nicht mit der Windowsversion des Programms nutzbar.[22]

3.6.2 Laden und Abspeichern eines Projekts

Die Eingabedaten eines Problems können sowohl abgespeichert als auch geladen werden, um jederzeit auf diese zugreifen zu können. Ohne Sicherungsmöglichkeit der Daten wäre dies nicht gegeben, da diese nicht im Quellcode des Programms, sondern nur in der grafischen Oberfläche eingegeben werden. Die Optionen können im Hauptfenster (Abb. 3.2) unter dem Menüpunkt "Project" aufgerufen werden. Dabei werden die Eingabedaten in einer Textdatei mit dem Namen des Problems abgelegt, standardmäßig im Unterverzeichnis "saves" des Programms. Über den implementierten Datei-Browser (Abb. 3.9) ist es aber auch möglich, diese Datei an einem beliebigen Ort zu erstellen. In der Textdatei werden die folgenden Angaben gesichert:

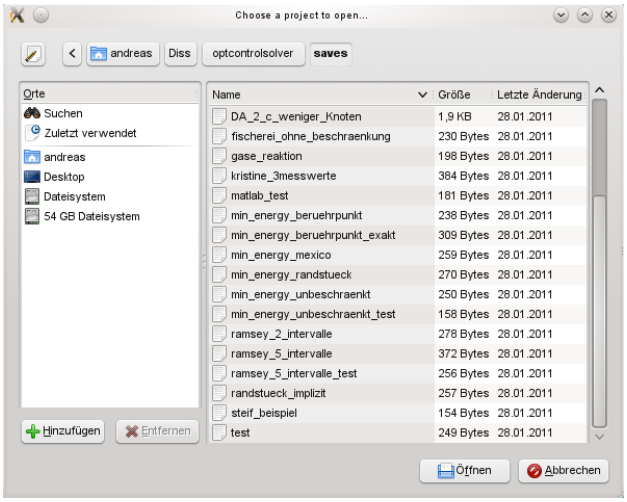


Abbildung 3.9: Öffnen eines Projekts

- Anzahl der Differenzialgleichungen

[22] Vergleiche dazu Abschnitt 3.5.2, Bemerkung 3.5.2.

- Anzahl der Schaltpunkte
- Art der Angabe der Steuerung:

$$\begin{cases} \text{explicit} \\ \text{implicit} \end{cases}$$

- Anzahl der Steuerungen
- Anzahl der Konstanten
- Anzahl der Konstanten mit bekanntem Wert
- Status der Option "Einfügen der Schaltpunkte in der Lösung":

$$\begin{cases} 1, \text{ falls ja} \\ 0, \text{ sonst} \end{cases}$$

- Maximale Anzahl der Newtoniterationen
- geforderte Genauigkeit der Mehrzielmethode
- geforderte Genauigkeit des Anfangswertproblemlösers
- Anfangsschrittweite des Anfangswertproblemlösers
- maximale Anzahl von Zeitschritten des Anfangswertproblemlösers
- maximale Anzahl der Verkleinerungen der Schrittweite in jedem Zeitschritt
- verwendetes Verfahren zur Lösung von Anfangswertproblemen
- Status der Option "Relaxation des Newtonverfahrens":

$$\begin{cases} 1, \text{ falls ja} \\ 0, \text{ sonst} \end{cases}$$

- Status der Option "Broydenapproximation der Jacobimatrix":

$$\begin{cases} 1, \text{ falls ja} \\ 0, \text{ sonst} \end{cases}$$

- Status der Option "Ausgabe einer detaillierten Lösung":

$$\begin{cases} 1, \text{ falls ja} \\ 0, \text{ sonst} \end{cases}$$

- Name der Datei, unter dem die Lösung gespeichert wird

- Differenzialgleichungen
- Steuerungen
- Startschätzungen der Schaltpunkte
- Startschätzungen der Konstanten
- Werte der bekannten Konstanten
- Mehrzielknoten
- Rand- und Schaltbedingungen
- Sprünge
- Startschätzungen
- selbst definierte Plots

Wird ein Projekt geladen, werden die Einträge der Datei eingelesen und in die entsprechenden Felder der grafischen Oberfläche geschrieben. Da die Daten in einer Textdatei gesichert werden, ist es möglich, die Daten eines Projekts direkt in dieser Datei zu ändern. Dies ist beispielsweise bei sehr langen Termen in den Differenzialgleichungen oder Steuerungen sinnvoll, weil die Übersichtlichkeit in der Textdatei besser ist als in den entsprechenden Feldern der grafischen Oberfläche. Dabei ist jedoch darauf zu achten, dass die Struktur der Datei nicht verändert wird, da die Daten anderenfalls nicht ordnungsgemäß eingelesen werden können und das Programm gegebenenfalls abstürzen kann. Daher ist dieses Vorgehen nur in Ausnahmefällen zu empfehlen.

3.6.3 Optionen der Mehrzielmethode

In diesem Abschnitt werden alle Optionen, die dem Anwender im Rahmen der Mehrzielmethode zur Verfügung stehen, zusammengefasst. Diese wurden in der Mehrzahl bereits im Abschnitt 2 angesprochen. Alle Optionen befinden sich in der Tabelle "PROJECT AND SETTINGS" des Hauptfensters (Abb. 3.2).

Newtonverfahren

Der Nutzer kann auf die folgenden Parameter bzw. Optionen des Verfahrens Einfluss nehmen:

- Die maximale Anzahl der Newtoniterationen kann festgelegt werden. Wird die maximale Zahl der Iterationen auf Null gesetzt, so werden lediglich die zur Bestimmung des Vektors F benötigten Anfangswertprobleme gelöst. So ist es möglich, einen Eindruck von der Güte der Startschätzung zu erhalten.

- Ein Benutzer kann die geforderte Genauigkeit des Newtonverfahrens variieren.[23]
- Weiterhin besteht die Möglichkeit, zu entscheiden, ob das gedämpfte Newtonverfahren verwendet wird oder nicht.[24]
- Gleiches gilt für die Anwendung eines Broyden-Updates zur Bestimmung der Jacobimatrix.[25] Da die Jacobimatrix lediglich approximiert und nicht durch das Lösen vieler Anfangswertprobleme bestimmt wird, verringert die Anwendung des Broyden-Updates den Rechenaufwand deutlich. Allerdings kann dies bei empfindlich reagierenden Problemen dazu führen, dass das Newtonverfahren aufgrund der größeren Ungenauigkeit der Jacobimatrix nicht konvergiert.

Anfangswertproblemlöser

Der Nutzer kann zwischen drei verschiedenen Anfangswertproblemlösern wählen. Das Standardverfahren ist *Dormand-Prince 5(4)*, ein eingebettetes Runge-Kutta-Verfahren der Ordnung 5 mit automatischer Schrittweitensteuerung. Wird ein Verfahren mit höherer Genauigkeit benötigt, kann *Dormand-Prince 8(7)* benutzt werden. Liegt hingegen ein steifes Differenzialgleichungssystem vor, fällt die Wahl auf das *Rosenbrock-Typ-Verfahren 4(3)*, einem linear impliziten Einschrittverfahren der Ordnung 4.[26] Weiterhin kann der Nutzer auf die folgenden Parameter der Anfangswertproblemlöser Einfluss nehmen:

- Geforderte Genauigkeit[27]
- Startschrittweite (relativ zum Zeitintervall des Anfangswertproblems)
- Minimale Schrittweite (ebenfalls relativ zum Zeitintervall)
- Maximale Zahl an Zeitschritten
- Maximale Zahl der Verkleinerungen der Schrittweite in einem Zeitschritt

Die voreingestellte Startschrittweite sorgt dafür, dass zu lösende Anfangswertproblem nicht mit einer zu großen Schrittweite gelöst werden. Bei empfindlicheren Problemen kann es aber notwendig sein, die Startschrittweite zu verringern, um eine Lösung zu erhalten. Wird die maximale Anzahl an Zeitschritten erreicht, kann man diese erhöhen oder versuchen, das Problem mit einem anderen besseren Verfahren zu lösen. Gleiches gilt für die maximale Zahl der Verkleinerungen der Schrittweite in einem Zeitschritt. Zusätzlich kann hier die Startschrittweite verringert werden, um die Anzahl der notwendigen Verkleinerungen zu senken.

[23] Zu den Abbruchkriterien des Newtonverfahrens vergleiche Abschnitt 2.1.4.
[24] Vergleiche dazu Abschnitt 2.1.2.
[25] Siehe Abschnitt 2.1.3.
[26] Zu den Verfahren und den folgenden Parametern vergleiche Abschnitt 2.2.
[27] Als Genauigkeit wird hier derjenige Wert bezeichnet, den die Fehlerabschätzung des jeweiligen Verfahrens in jedem Zeitschritt unterschreiten muss.

3.6.4 Lösung eines Problems, Erstellen von Grafiken und PDF-Export

Sind alle Daten des zu lösenden Problems eingegeben sowie alle Optionen der Mehrzielmethode wie gewünscht gesetzt, kann durch einen Klick auf den Button "Solve" im Hauptfenster (Abb. 3.2) die Berechnung der Lösung mittels des Mehrfachschießverfahrens gestartet werden. Bevor die Berechnungen beginnen, werden alle Daten des Problems sowie die Optionen aus der grafischen Oberfläche eingelesen und in Variablen verschiedenen Typs gespeichert. Dabei wird unter anderem sichergestellt, dass die konfigurierbaren Parameter sinnvolle Werte enthalten. Die Parameter dürfen beispielsweise nicht negativ sein. In einem solchen Fall wird dem entsprechenden Parameter ein Standardwert zugewiesen. Dies sind diejenigen, die beim Start des Programms in den jeweiligen Feldern eingetragen sind. Anschließend werden drei Tests durchgeführt:

1. Der erste Test besteht darin, zu überprüfen, ob überhaupt Problemdaten eingegeben wurden. Dies wird beispielhaft anhand der Anzahl der angegebenen Differenzialgleichungen entschieden.

2. Es wird geprüft, ob die Summe der Differenzialgleichungen, Konstanten und Schaltpunkte gleich der angegebenen Bedingungen ist. Dies muss der Fall sein, damit die Möglichkeit der Existenz einer eindeutig bestimmten Lösung besteht.

3. Die Anzahl der Schaltpunkte wird über die Anzahl der angegebenen Startschätzungen für Schaltpunkte ermittelt. Daher muss sichergestellt werden, dass in der Tabelle für die Eingabe der Steuerungen genügend Steuerungsintervalle mit Daten gefüllt wurden. Enthält das Problem i Schaltpunkte, müssen für jede Steuerung $i + 1$ Steuerungsintervalle angegeben werden.

Sofern die drei Tests positiv ausfallen und außerdem eine positive maximale Anzahl von Newtoniterationen angegeben ist, so wird die Mehrzielmethode gestartet und anderenfalls eine Fehlermeldung ausgegeben. Es ist möglich, die Anzahl der Newtoniterationen auf Null zu setzen. In diesem Fall werden lediglich diejenigen Anfangswertprobleme gelöst, deren Lösung für die Bestimmung des Vektors F benötigt wird, und die Mehrzielmethode anschließend beendet. So steht dem Anwender die Möglichkeit offen, eine Einschätzung über die Güte der Startschätzung zu erhalten.

Wird die Berechnung einer Lösung des Problems mittels der Mehrzielmethode gestartet, so wechselt das Programm automatisch auf den Tab "Solution" (Abb. 3.10). Dabei zeigt ein Fortschrittsbalken den ungefähren Fortschritt der Berechnungen innerhalb einer Iteration des Newtonverfahrens. Dort werden in einem Textfenster verschiedene Angaben zum Problem sowie dessen Lösung ausgegeben. Dies sind im Einzelnen:

1. Die Eingabedaten des Problems.

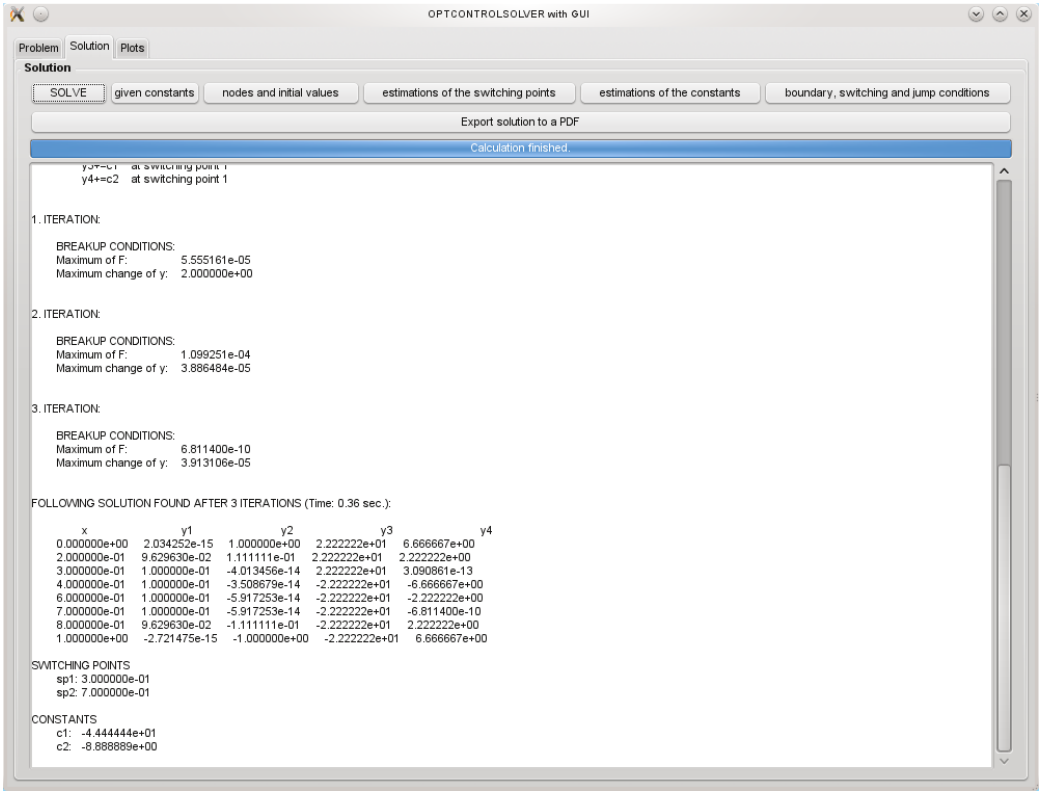


Abbildung 3.10: Ausgabe der Lösung

2. Die Abbruchkriterien und optional zusätzlich der Vektor F sowie die Werte der Zustandsvariablen, Konstanten und Schaltpunkte in jeder Iteration.

3. Die Lösung des Problems mit Angabe der benötigten Iterationen des Newtonverfahrens und der benötigten Zeit[28] bzw. gegebenenfalls der Meldung, dass keine Lösung gefunden wurde.

Die optionale Ausgabe erfolgt nur, wenn im Hauptfenster der Menüpunkt "detailed solution" ausgewählt wurde. Wird außerdem der Menüpunkt "insert switching points in the solution" ausgewählt, werden die Schaltpunkte sowie die zugehörigen Werte der Zustands- und adjungierten Variablen als zusätzliche Mehrzielknoten bei der Ausgabe der Lösung eingefügt. Des Weiteren werden alle Angaben auch in der Konsole, sofern das Programm über eine solche gestartet wurde, ausgegeben. Hier werden zusätzlich die Teilmatrizen der Jacobimatrix sowie eine Aufschlüsselung der benötigten Zeit in die Berechnung des Vektors F, der Jacobimatrix DF sowie der Lösung des linearen Gleichungssystems angegeben. Diese Daten werden automatisch in ein Logfile geschrieben, sodass diese auch in dem Fall, dass das Programm nicht über eine Konsole gestartet wurde, für den Benutzer verfügbar sind.

[28] Die für die Berechnung der Lösung benötigte Zeit wird lediglich auf Linuxsystemen ausgegeben, da die für diese Funktion verwendete Headerdatei unter Windows nicht verfügbar ist.

Dieses Logfile kann entweder über den Button "Show Logfile" oder direkt im Programmverzeichnis durch Öffnen der Datei "logfile.dat" angezeigt werden. Sollte das Programm während der Berechnung aufgrund von Fehlern abstürzen, kann die Ansicht dieses Logfiles Aufschluss darüber geben, bei welchem Teil der Berechnungen ein Fehler aufgetreten ist. Zusätzlich steht dem Benutzer auch die Möglichkeit offen, die Lösung in einer Datei abzuspeichern. Dies wird durch die Auswahl der Option "save solution to.." sowie der Angabe des Dateinamens, unter dem die Lösung gespeichert werden soll, erreicht. Die Datei enthält die folgenden Bestandteile der Lösung:

- Anzahl der Mehrzielknoten, Differenzialgleichungen, Schaltpunkte und zu optimierenden Konstanten
- Positionen der Mehrzielknoten sowie die Funktionswerte der Variablen an diesen Stellen
- Positionen der Schaltpunkte

Die Lösung wird im Unterordner "solutions" gespeichert. Diese kann dann beispielsweise als Startschätzung geladen werden.

Wurde eine Lösung des Problems gefunden, so kann der Benutzer verschiedene Plots der Lösung erstellen lassen. Dies ist im Karteikartenreiter "Plots" (Abb. 3.11) des Hauptfensters möglich. Dabei stehen dem Anwender

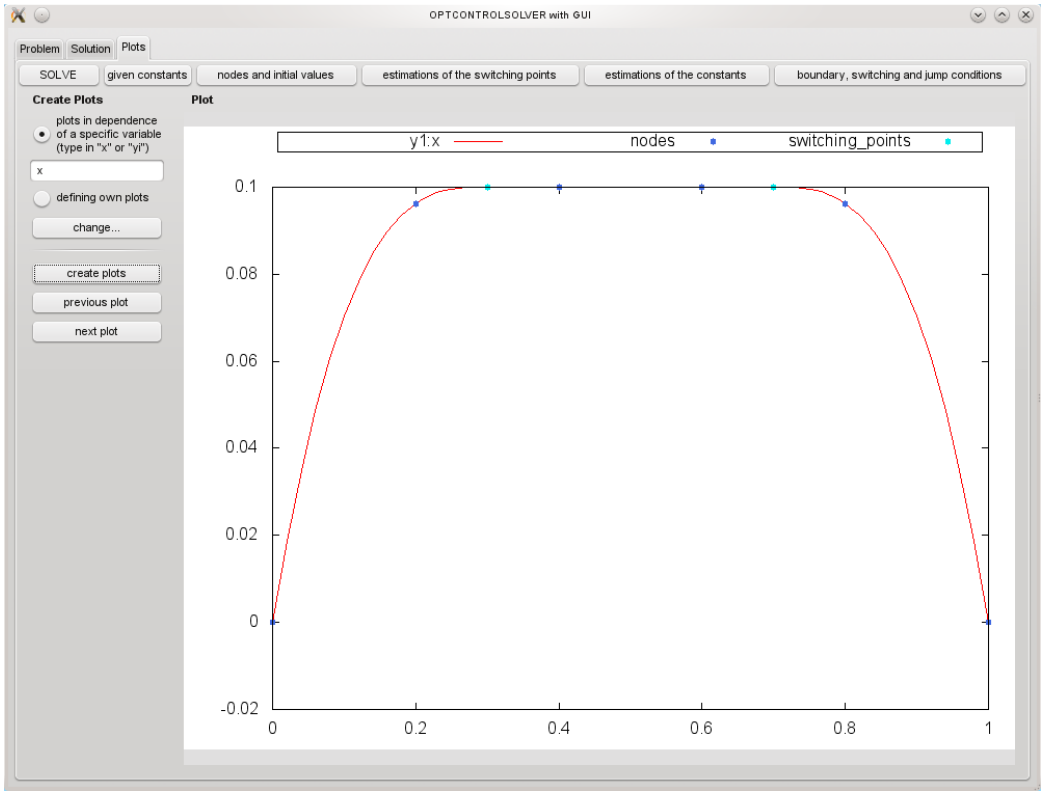


Abbildung 3.11: Erstellung und Ausgabe der Grafiken

die folgenden Optionen zur Verfügung:

- Es werden alle Zustandsvariablen und Adjungierten sowie die Steuerungen in Abhängigkeit von der Zeit geplottet.
- Alle Größen werden in Abhängigkeit von einer frei wählbaren Variablen geplottet.
- Der Nutzer erstellt bis zu zehn eigene Grafiken. Dabei kann jede Grafik bis zu drei Plots enthalten. Vergleiche dazu Abb. 3.12.

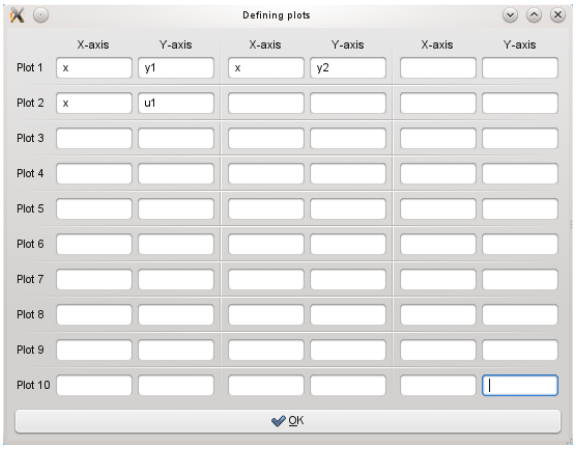


Abbildung 3.12: Definition eigener Grafiken

Außerdem können die Eingabedaten, die Lösung sowie die Plots eines Problems in eine PDF-Datei exportiert werden. Vergleiche dazu die Abschnitte 3.3 bzw. 3.4.

Bemerkung 3.6.2 Die Funktionen für die Erstellung der Grafiken sowie des PDF-Exports stehen sowohl nach dem Starten des Programms als auch nach dem Öffnen eines Projekts nicht zur Verfügung. Dies wird durch die Deaktivierung der entsprechenden Buttons erreicht. Wird die Mehrzielmethode erfolgreich gestartet[29], können anschließend beide Funktionen genutzt werden. □

3.6.5 Fehlerbehandlung

Im Folgenden sollen einige Fehler genannt werden, die bei der Lösung eines Problems mit dem vorliegenden Programm auftreten können. Dabei wird davon ausgegangen, das alle Eingabedaten korrekt eingegeben wurden. Bei der Mehrzielmethode ist stets zu bedenken, dass in der Regel gute Startnäherungen benötigt werden, da diese einen großen Einfluss auf die Konvergenz des Newtonverfahrens haben. Die Güte der Startnäherung kann überprüft werden, indem die Einstellung der maximalen Anzahl der Newtoniterationen

[29] D.h. die auf S.73 angesprochenen Test sind positiv ausgefallen.

auf Null gesetzt wird. Dann wird lediglich das Anfangswertproblem ausgehend von den gegebenen Startwerten gelöst und man kann anhand der Plots Rückschlüsse auf die Güte der Startwerte und/oder Mehrzielknoten ziehen. Sollte das Programm während der Lösung eines Problems abstürzen, so lassen sich durch Ansicht der automatisch erstellten Logdatei[30] Anhaltspunkte finden, an welcher Stelle der Berechnung ein Fehler aufgetreten ist.

Nun wird aufgezeigt, welche Ursache gewisse auftretende Fehler haben können sowie wie mit diesen umgegangen werden kann.

1. Es wird die Meldung ausgegeben, dass der verwendete Anfangswertproblemlöser zu viele Zeitschritte benötigt hat oder die Schrittweite zu oft verkleinert werden musste, um die Genauigkeitsforderung zu erfüllen.
   - Sofern Ersteres aufgetreten ist, kann die maximale Anzahl der Zeitschritte erhöht werden.
   - Die Startschrittweite kann verringert werden.
   - Wurde Dormand-Prince 5(4) verwendet, so kann Dormand-Prince 8(7) gewählt werden, da dieses aufgrund der größeren Ordnung eine höhere Genauigkeit bei der Lösung von Anfangswertproblemen erreicht.
   - Führen die beiden expliziten AWP-Löser nicht zu einem Ergebnis, so kann stattdessen der linear-implizite Anfangswertproblemlöser Rosenbrock 4(3) verwendet werden. Dieser kann auch mit steifen Differenzialgleichungssystemen umgehen.
2. Ein Anfangswertproblem konnte nicht gelöst werden und es treten *nan*- und/oder *inf*-Fehler auf.
   - Die Startwerte an den Mehrzielknoten sollten dahingehend überprüft werden, ob beispielsweise eine Division durch Null auftritt.
   - Die Startschrittweite kann verringert werden. Bei empfindlichen Differenzialgleichungssystemen ist häufig eine kleine Startschrittweite nötig.
   - Ein genauerer Anfangswertproblemlöser kann verwendet werden.
   - Eventuell sind ein oder mehrere Integrationsintervalle zu groß gewählt worden. Dieses Problem kann durch eine Änderung der Position und der Anzahl der Mehrzielknoten behoben werden.
   - Die Genauigkeit des verwendeten Lösers kann erhöht werden.
3. Die Mehrzielmethode konvergiert nicht.
   - Wurde ein Broyden-Update verwendet, so kann die Deaktivierung dieser Funktion die Konvergenz verbessern. Insbesondere empfindliche Differenzialgleichungssysteme sind auf genaue Jacobimatrizen angewiesen, damit das Newtonverfahren konvergiert. Alternativ kann das jeweils andere Broyden-Update verwendet werden, da

[30] Diese befindet sich im Programmverzeichnis.

diese Updates abhängig vom zu lösenden Problem unterschiedlich gute Ergebnisse liefern.

- Die Mehrzielmethode benötigt genaue Lösungen der Anfangswertprobleme. Daher kann sowohl die Erhöhung der geforderten Genauigkeit des verwendeten Lösers als auch die Auswahl eines genaueren Lösers das Konvergenzverhalten positiv beeinflussen.
- Die Relaxation des Newtonverfahrens verbessert im Allgemeinen das Konvergenzverhalten.
- Auch eine Erhöhung der maximalen Anzahl der Iterationen des Newtonverfahrens kann zur Konvergenz führen, sofern die Berechnungen nach dieser Iterationsanzahl angebrochen wurden. Konvergiert das Verfahren beispielsweise aufgrund schlechter Startnäherungen nur sehr langsam, kann durchaus eine große Zahl von Iterationen benötigt werden. Dies kann insbesondere der Fall sein, wenn das gedämpfte Newtonverfahren verwendet wird, da die Newtonschritte dann sehr klein sein können.

4. Es wird die Meldung ausgegeben, dass die Position der Schaltpunkte vom Programm verändert werden musste.

   - Diese Meldung tritt auf, wenn Schaltpunkte nach einem Newtonschritt außerhalb des betrachteten Zeitintervalls lagen oder vertauscht wurden. In einem solchen Fall sollten die Schaltstruktur sowie die Startschätzungen der Schaltpunkte überprüft und gegebenenfalls korrigiert werden.

Kapitel 4

Beispielrechnungen

4.1 Das Brachistochrone-Problem

Als erstes Beispiel soll das in Abschnitt 1.1 vorgestellte Brachistochrone-Problem betrachtet werden. Das zu lösende Optimalsteuerungsproblem lautet wie folgt:[1]

Man bestimme die Fallzeit $t_b > 0$ und eine Steuerung $u \in C^1([t_a, t_b], \mathbb{R})$, sodass das Funktional

$$I(x, v, u, t_b) := t_b = \int_0^{t_b} 1\, dt$$

minimiert wird unter den Nebenbedingungen

$$\begin{aligned}
\dot{x}_1 &= v_1, & x_1(0) &= 0, & x_1(t_b) &= 2,\\
\dot{x}_2 &= v_2, & x_2(0) &= 0, & x_2(t_b) &= -1,\\
\dot{v}_1 &= -u\, v_2, & v_1(0) &= 0,\\
\dot{v}_2 &= -g + u\, v_1, & v_2(0) &= 0.
\end{aligned}$$

Die Endzeit t_b und die Endgeschwindigkeit $(v_1(t_b), v_2(t_b))^T$ sind frei.

Um dieses Optimalsteuerungsproblem in ein Randwertproblem zu überführen, sind mit Hilfe des Maximumprinzips die notwendigen Bedingungen[2] zu ermitteln. Die zugehörige Hamiltonfunktion lautet:

$$\begin{aligned}
H &= 1 + \lambda_1 v_1 + \lambda_2 v_2 - \lambda_3 v_2 u + \lambda_4 (v_1 u - g)\\
&= (1 + \lambda_1 v_1 + \lambda_2 v_2 - \lambda_4 g) + u(\lambda_4 v_1 + \lambda_3 v_2).
\end{aligned} \tag{4.1}$$

Die adjungierten Differenzialgleichungen ergeben sich dann durch Differen-

[1] Vergleiche zur Herleitung Beispiel 1.1.1.
[2] Vergleiche dazu die Ausführungen in Abschnitt 1.3.

zieren der Hamiltonfunktion 4.1:

$$\begin{aligned}
\dot{\lambda}_1 &= -\frac{\partial H}{\partial x_1} = 0, \\
\dot{\lambda}_2 &= -\frac{\partial H}{\partial x_2} = 0, \\
\dot{\lambda}_3 &= -\frac{\partial H}{\partial v_1} = -\lambda_1 - u\lambda_4, \\
\dot{\lambda}_4 &= -\frac{\partial H}{\partial v_2} = -\lambda_2 + u\lambda_3.
\end{aligned} \tag{4.2}$$

Da die Zustandsvariablen v_1 und v_2 keine Endbedingungen zu erfüllen haben, folgt aus den Orthogonalitätsbedingungen:

$$\lambda_3(t_b) = \lambda_4(t_b) = 0. \tag{4.3}$$

Aufgrund der Tatsache, dass der Endzeitpunkt nicht gegeben ist, folgt aus den Transversalitätsbedingungen, dass die Hamiltonfunktion 4.1 im Endzeitpunkt t_b verschwinden muss:

$$H[t_b] = 0. \tag{4.4}$$

Nun ist die optimale Steuerung u ebenfalls durch Ableiten der Hamiltonfunktion 4.1 zu bestimmen. Da die Steuerung linear in das Problem eingeht, hängt die Ableitung $H_u = \lambda_4 v_1 - \lambda_3 v_2$ nicht von der Steuerung ab und liefert somit keine Informationen über diese. In einem solchen Fall spricht man von einem singulären Optimalsteuerungsproblem. Man kann allerdings einen Trick anwenden, um auch in diesem Fall Informationen über die optimale Steuerung zu gewinnen. Da H_u längs der Lösung den Wert Null annimmt, müssen auch sämtliche Zeitableitungen von H_u verschwinden. H_u sowie deren beiden ersten Zeitableitungen lauten:

$$H_u = \lambda_4 v_1 - \lambda_3 v_2 = 0, \tag{4.5}$$

$$\frac{d}{dt} H_u = \lambda_1 v_2 - \lambda_2 v_1 + \lambda_3 g = 0,$$

$$\frac{d^2}{dt^2} H_u = -2\lambda_1 g + u(\lambda_1 v_1 + \lambda_2 v_2 - \lambda_4 g) = 0. \tag{4.6}$$

Die zweite Zeitableitung ist explizit von der Steuerung u abhängig. Da das vorliegende Problem autonom ist, ist die Hamiltonfunktion längs der Lösung konstant[3] und verschwindet aufgrund der Randbedingung (4.4) sogar:

$$H = (1 + \lambda_1 v_1 + \lambda_2 v_2 - \lambda_4 g) + u(\lambda_4 v_1 + \lambda_3 v_2) = 0. \tag{4.7}$$

Einsetzen von (4.5) in (4.7) liefert

$$\lambda_1 v_1 + \lambda_2 v_2 - \lambda_4 g = -1.$$

[3]Siehe Bemerkung 1.3.8.

Dies wiederum setzt man in die zweite Zeitableitung von H_u (4.6) ein und erhält eine Aussage über die optimale Steuerung:

$$u = -2\lambda_1 g.$$

Da die Adjungierte λ_1 konstant ist, folgt somit, dass die optimale Steuerung u ebenfalls konstant ist. Man kann also die Steuerung u durch eine zu bestimmende Konstante ersetzen und das System sogar ohne Verwendung der adjungierten Differenzialgleichungen lösen.

Man erhält - nach einer Transformation des Problems auf das Zeitintervall $[0, 1]$ - das folgende Randwertproblem:

$$\begin{aligned}
\dot{x}_1 &= t_b v_1, & x_1(0) &= 0, & x_1(1) &= 2,\\
\dot{x}_2 &= t_b v_2, & x_2(0) &= 0, & x_2(1) &= -1,\\
\dot{v}_1 &= -t_b u v_2, & v_1(0) &= 0,\\
\dot{v}_2 &= t_b(u v_1 - g), & v_2(0) &= 0,\\
\dot{u} &= 0,\\
\dot{t}_b &= 0. & & & & .
\end{aligned} \tag{4.8}$$

Nun wird das Randwertproblem so verändert, dass es mit dem vorliegenden Programm gelöst werden kann. Die Variablennamen sowie die Randbedingungen werden an die Syntax des Programms angepasst. Weiterhin kann man die Dimension des Differenzialgleichungssystems verringern, indem man die Steuerung u und die Endzeit t_b als Konstanten $c1$ bzw. $c2$, deren Wert unbekannt ist, auffasst.[4]

- Umbenennung der Variablen:

$$\begin{aligned}
y1 &:= x_1 & y2 &:= x_2\\
y3 &:= v_1 & y4 &:= v_2\\
u1 &:= u
\end{aligned}$$

- bekannte Konstante:

$$p1 := g = 9.81$$

- Konstanten mit unbekanntem Wert:

$$c1 := u \qquad c2 := t_b$$

- Differenzialgleichungen:

$$\begin{aligned}
\dot{y1} &= c2 \cdot y3 & \dot{y2} &= c2 \cdot y4\\
\dot{y3} &= -c2 \cdot u1 \cdot y4 & \dot{y4} &= -c2 \cdot (u1 \cdot y3 - p1)
\end{aligned}$$

[4] Vergleiche dazu die Bedienung des Programms in Abschnitt 3.6.

- Randbedingungen:

$$\begin{array}{ll} y1_s & y1_e - 2 \\ y2_s & y2_e + 1 \\ y3_s & y4_s \end{array}$$

- Steuerung:

$$u1 = c1$$

Das Randwertproblem wird mit den folgenden Startwerten (Tabelle 4.1) und Einstellungen gelöst:

t	x_1	x_2	v_1	v_2
0	0	0	0	0
0.25	0	0	0	0
0.5	0	0	0	0
0.75	0	0	0	0
1	0	0	0	0

$$c_1 = 4 \qquad c_2 = 1$$

Tabelle 4.1: Brachistochrone-Problem: Startwerte

- Newtonverfahren:
  - Maximale Anzahl der Newtoniterationen: 30
  - geforderte Genauigkeit der Lösung: 10^{-5}
  - Relaxation des Newtonverfahrens: ja
  - Broyden-Update: ja, gut
- Anfangswertproblemlöser:
  - geforderte Genauigkeit: 10^{-5}
  - Startschrittweite: 1
  - minimale Schrittweite: 10^{-6}
  - maximale Zahl der Zeitschritte: 500
  - maximale Anzahl der Verkleinerungen der Schrittweite in einem Zeitschritt: 10

Nach zehn Iterationen des Newtonverfahrens und 1.15 Sekunden liefert das Programm die in Tabelle 4.2 sowie Abbildung 4.1 dargestellte Lösung.[5]

Nun soll der Einfluss einiger Optionen auf die Iterationszahl und die benötigte Zeit untersucht werden. Die Ergebnisse sind in den Tabellen 4.3

t	x_1	x_2	v_1	v_2
0	$2.37931e-15$	$1.76194e-15$	$-3.90306e-15$	$1.68467e-15$
0.25	0.0559659	-0.186508	0.812273	-1.7319
0.5	0.398737	-0.611517	2.66326	-2.21472
0.75	1.10827	-0.968503	4.218	-1.10024
1	2	-1	4.35517	0.807762

Steuerung: $u = c_1 = 4.35517$

Endzeitpunkt: $t_b = c_2 = 0.805564.$

Tabelle 4.2: Brachistochrone-Problem: optimale Lösung

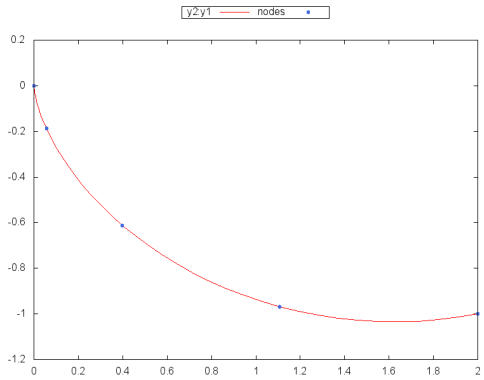


Abbildung 4.1: Brachistochrone-Problem: optimale Kurve

sowie 4.4 dargestellt, wobei die im Vergleich zur ersten Lösung veränderte Option jeweils fett gedruckt ist.

Eine Veränderung der Einstellungen des Newtonverfahrens bewirken bei der Lösung des Brachistochrone-Problems keine gravierenden Änderungen. Das Programm konvergiert in allen Fällen und liefert auch die identische Lösung, lediglich die Genauigkeit der Lösung schwankt. Wird die Relaxation des Newtonverfahrens deaktiviert, so werden deutlich weniger Iterationen (6 statt 10), aber die gleiche Zeit benötigt. Dies lässt darauf schließen, dass das Newtonverfahren hier zu stark gedämpft wird, sodass die Lösung mit vollen Newtonschritten nach weniger Iterationen erreicht wird. Dass die benötigte Zeit dennoch nahezu identisch ist, liegt daran, dass das Broyden-Update nur bei aktivierter Relaxation angewendet werden kann, wodurch die Dauer einer Iteration bei Verwendung eines Broyden-Updates enorm sinkt. Einen großen Unterschied kann man hingegen feststellen, wenn statt dem guten das schlechte Broyden-Update verwendet wird. In der Theorie sind beide Vorgehensweisen gleich gut[6], bei diesem Problem funktioniert das schlechte Broyden-Update allerdings deutlich besser. Zum Einen werden lediglich 7 Ite-

[5]Die Ausgabe des Programms mit deaktivierter Option "detailed solution" kann im Anhang A.1 auf S.119ff nachgelesen werden.

[6]Vergleiche zum Broyden-Update den Abschnitt 2.1.3 auf S. 30.

	1	2	3	4	5	6
Relaxation	ja	**nein**	ja	ja	ja	ja
Broyden-Update	ja	ja	ja	**nein**	ja	ja
Art des Updates	gut	gut	**schlecht**	**-**	gut	gut
Genauigkeit	1e-5	1e-5	1e-5	1e-5	**1e-7**	**1e-3**
Iterationen	10	6	7	6	13	8
Zeit in Sekunden	1.15	1.2	0.62	1.2	1.25	1.08

Tabelle 4.3: Brachistochrone: Vergleich der Optionen des Newtonverfahrens

rationen benötigt und zum Anderen, was viel wichtiger ist, halbiert sich die Rechenzeit. Dieser Effekt kann aber bei anderen Problemstellungen durchaus genau umgekehrt ausfallen. Wird die geforderte Genauigkeit des Newtonverfahrens geändert, zeigt sich der erwartete Effekt. Bei höherer Genauigkeit werden mehr Iterationen benötigt, während eine geringere in weniger Iterationen resultiert. Dies spiegelt sich allerdings kaum in der Rechenzeit nieder, da der Aufwand pro Iteration bei Anwendung des Broyden-Updates sehr gering ist.

	1	2	3	4	5	6
AWP-Löser	DP78	**DP45**	**RB34**	DP78	DP78	DP78
Genauigkeit	1e-5	1e-5	1e-5	**1e-7**	**1e-3**	1e-5
Startschrittweite	1	1	1	1	1	**0.25**
Iterationen	10	10	10	10	10	10
Zeit in Sekunden	0.86	8.5	22.1	2.27	1.02	2.1

Tabelle 4.4: Brachistochrone: Vergleich der Optionen der AWP-Löser

Die verschiedenen die Anfangswertproblemlöser betreffenden Optionen haben bei dem vorliegenden Optimalsteuerungsproblem vor allem Auswirkungen auf die benötigte Zeit, um eine Lösung zu finden, nicht aber auf die Zahl der Newtoniterationen. Wird statt DP78 der weniger genaue AWP-Löser DP45 verwendet, so vervielfacht sich die Rechenzeit, obwohl der Aufwand pro Zeitschritt deutlich geringer ist.[7] Der Grund dafür ist, dass der geringere Aufwand auch eine geringere Genauigkeit bedeutet, die durch eine kleinere Schrittweite ausgeglichen werden muss. Der Anfangswertproblemlöser DP78 ist also auch bei einfachen Problemen gegebenenfalls vorzuziehen, da der Nachteil des höheren Aufwands pro Zeitschritt durch größere Schrittweiten kompensiert werden und sich sogar ins Gegenteil umkehren kann. Wird das Rosenbrock-Verfahren für die Lösung der Anfangswertprobleme angewendet, so erhöht sich die Rechenzeit noch einmal deutlich. Dies liegt im deutlich höheren Aufwand, den dieses Verfahren erfordert, begründet. Liegt ein steifes Differenzialgleichungssystem vor, so ist dieser Aufwand berechtigt, da das

[7] Hier musste zusätzlich die maximale Anzahl der Verkleinerungen der Schrittweite pro Zeitschritt auf den Wert 20 erhöht werden, um eine Lösung zu erhalten.

Verfahren in solchen Fällen mit viel größeren Schrittweiten arbeiten kann. Man sollte dieses Verfahren also nur verwenden, wenn ein steifes System vorliegt, anderenfalls ist der Aufwand und damit auch die Rechenzeit viel zu groß. Eine Änderung der geforderten Genauigkeit des Lösers hat den erwarteten Effekt - lediglich die benötigte Zeit fällt höher bzw. geringer aus. Beim einfachen Brachistochrone-Problem hat die Genauigkeit keinen Einfluss auf das Newtonverfahren. Dies ist aber im Allgemeinen anders, da die Mehrzielmethode enorm von genauen Anfangswertproblemlösern profitiert. Eine höhere Genauigkeit kann also durchaus eine geringere Zahl der Newtoniterationen bewirken als auch entscheidend für die Konvergenz des Newtonverfahrens sein. Eine Verringerung der Startschrittweite resultiert hier in einer Erhöhung der benötigten Zeit. Daraus kann man schließen, dass der Anfangswertproblemlöser bei diesem Problem mit einer großen Schrittweite arbeiten kann. Wird dieser mit einer kleineren Schrittweite gestartet, vergrößert sich diese durch die automatische Schrittweitensteuerung nur allmählich, wodurch mehr Zeitschritte benötigt werden und der Aufwand steigt. In anderen Problemstellungen kann eine Verringerung der Startschrittweite allerdings auch in einem verbesserten Aufwand münden[8], aber auch notwendig sein, um überhaupt eine Lösung zu erhalten.

4.2 Das Min-Energy-Problem

Im zweiten Beispiel soll nun das sogenannte Min-Energy-Problem

$$\begin{aligned} I(t,x,v,u) &:= \int_0^1 \frac{1}{2}u(t)^2 dt \to \min, \\ \dot{x} &= v, \\ \dot{v} &= u, \\ x(0) &= x(1) = 0, \\ v(0) &= 1,\ v(1) = -1, \\ s(t,x(t),v(t)) &= x(t) - l \leq 0, \quad l > 0,\ t \in [0,1], \end{aligned} \tag{4.9}$$

betrachtet werden.[9] Die Ungleichungsnebenbedingung ist eine globale, reine Zustandsbeschränkung, diese muss also auf dem gesamten Zeitintervall erfüllt sein und ist unabhängig von der Steuerung u. Wie sich in den folgenden Unterabschnitten zeigen wird, ist es von der konstanten Größe l abhängig, ob die Lösung des Problems unbeschränkt ist (die Ungleichungsnebenbedingung wird nicht aktiv), einen Berührpunkt (die Nebenbedingung ist in einem Zeitpunkt aktiv) oder aber ein Randstück (die Nebenbedingung ist auf einem Zeitintervall mit Gleichheit erfüllt) aufweist.

[8] Dies ist der Fall, wenn eine kleine Schrittweite benötigt wird, um die geforderte Genauigkeit zu erreichen.

[9] Dieses wird beispielsweise in [1] S.123-125 und [11] S.46ff behandelt.

4.2.1 Das unbeschränkte Min-Energy-Problem

Zunächst wird das Problem ohne Berücksichtigung der Beschränkung

$$s(t, x(t), v(t)) = x(t) - l \leq 0, \quad l > 0, \quad t \in [0, 1], \tag{4.10}$$

gelöst und überprüft, welche Werte l annehmen darf, sodass die Bedingung (4.10) trotz der Nichtberücksichtigung erfüllt ist.

Die Hamiltonfunktion ist gegeben durch:

$$H(t, x, v, \lambda, u) = \frac{1}{2}u^2 + \lambda_x v + \lambda_v u \tag{4.11}$$

und die adjungierten Differenzialgleichungen lauten wie folgt:

$$\dot{\lambda}_x = -\frac{\partial H}{\partial x} = 0, \qquad \dot{\lambda}_v = -\frac{\partial H}{\partial v} = -\lambda_x. \tag{4.12}$$

Die Steuerung u ergibt sich durch Nullsetzen der Ableitung der Hamiltonfunktion (4.11):

$$H_u = u + \lambda_v = 0 \Rightarrow u = -\lambda_v, \tag{4.13}$$

$$H_{uu} = 1 > 0. \tag{4.14}$$

Aus (4.14) folgt, dass die Hamiltonfunktion bezüglich u strikt konvex ist und daher ein eindeutig bestimmtes Minimum besitzt. Die Hamiltonfunktion ist also regulär und die Steuerung damit stetig.

Da für die Zustandsvariablen jeweils sowohl im Start- als auch im Endzeitpunkt Bedingungen vorgeschrieben sind, ergeben sich aus den Orthogonalitätsbedingungen keine weiteren Bedingungen für die adjungierten Variablen.[10]

Die Dimension des Differenzialgleichungssystems wird durch Einführung einer gesuchten Konstanten $c1 := \lambda_x$ verringert, das zu lösende Randwertproblem aufgestellt

- Umbenennung der Variablen:

$$y1 := x \qquad y2 := v$$
$$y3 := \lambda_v \qquad u1 := u$$

- Konstanten mit unbekanntem Wert:

$$c1 := \lambda_x$$

- Differenzialgleichungen:

$$\dot{y1} = y2 \qquad \dot{y2} = u1$$
$$\dot{y3} = -c1$$

[10] Siehe Satz 1.3.4 und die nachfolgenden Ausführungen auf S.12f.

- Steuerung:

$$u1 = -y3$$

- Randbedingungen:

$$\begin{array}{ll} y1_s & y1_e \\ y2_s - 1 & y2_e + 1 \end{array}$$

und mit den folgenden Startwerten (Tabelle 4.5) und Einstellungen gelöst:

t	y_1	y_2	y_3
0	0	0	0
0.2	0	0	0
0.4	0	0	0
0.6	0	0	0
0.8	0	0	0
1	0	0	0

$$c_1 = 0$$

Tabelle 4.5: Min-Energy-Problem (unbeschränkt): Startwerte

- Newtonverfahren:
  - Maximale Anzahl der Newtoniterationen: 10
  - geforderte Genauigkeit der Lösung: 10^{-5}
  - Relaxation des Newtonverfahrens: ja
  - Broyden-Update: ja, gut
- Anfangswertproblemlöser:
  - geforderte Genauigkeit: 10^{-5}
  - Startschrittweite: 1
  - minimale Schrittweite: 10^{-6}
  - maximale Zahl der Zeitschritte: 500
  - maximale Anzahl der Verkleinerungen der Schrittweite in einem Zeitschritt: 10

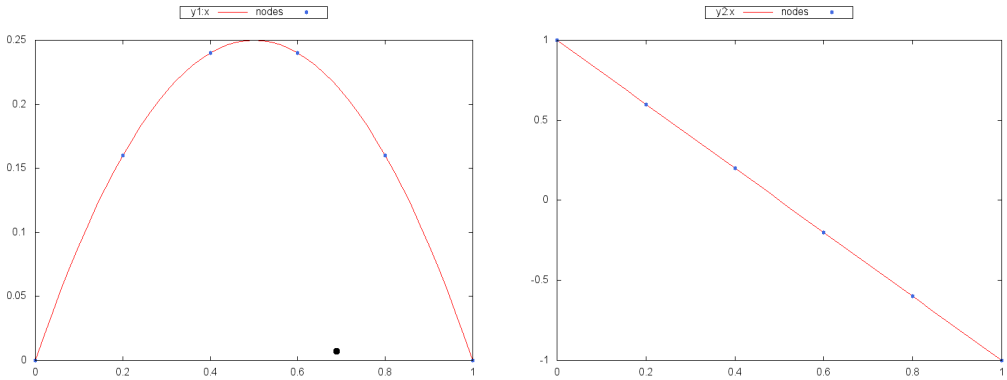
Die Mehrzielmethode findet nach zwei Iterationen (0.15 Sekunden) die in Tabelle 4.6 dargestellte Lösung.[11] Die Abbildungen 4.2 und 4.3 zeigen die optimalen Trajektorien für die Zustandsvariablen x und v sowie die optimale Steuerung u. Man kann erkennen, dass die Beschränkung (4.10) keinen Einfluss auf die optimale Lösung hat, sofern $l > 0.25$ gilt.

[11] Die Ausgabe des Programms mit deaktivierter Option "detailed solution" kann im Anhang A.2 auf S.121ff nachgelesen werden.

t	y_1	y_2	y_3
0	0	1	2
0.2	0.16	0.6	2
0.4	0.24	0.2	2
0.6	0.24	-0.2	2
0.8	0.16	-0.6	2
1	0	-1	2

Steuerung:	$u = -y3 = -2$
gesuchte Konstante:	$c1 = 0$

Tabelle 4.6: Min-Energy-Problem (unbeschränkt): optimale Lösung



(a) x (b) v

Abbildung 4.2: optimale Trajektorien der Zustandsvariablen

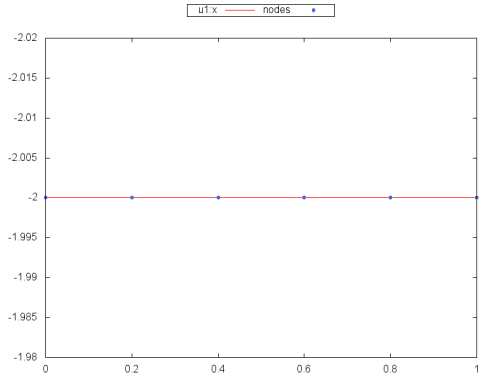


Abbildung 4.3: optimale Steuerung

4.2.2 Das Min-Energy-Problem mit Berührpunkt

Wie bereits gesehen, hat die Nebenbedingung (4.10) lediglich einen Einfluss auf die optimale Lösung des Min-Energy-Problems, wenn $l <= 0.25$ gilt. Sei nun $l = 0.2$ gegeben. Man gehe zunächst davon aus, dass die Ungleichungsbedingung lediglich in einem Zeitpunkt und nicht in einem ganzen Zeitintervall mit Gleichheit erfüllt ist. In diesem Fall enthält die optimale Lösung einen Schaltpunkt t_b, in dem die Steuerung umgeschaltet wird. In diesem Schaltpunkt gilt also:

$$x(t_b) - 0.2 = 0. \tag{4.15}$$

Des Weiteren folgt aufgrund der Stetigkeit der Geschwindigkeit v:

$$v(t_b) = 0. \tag{4.16}$$

Angenommen, die Geschwindigkeit sei im Schaltpunkt positiv, dann würde das Fahrzeug die Nebenbedingung nach dem Schaltpunkt verletzen. Nimmt man stattdessen an, dass die Geschwindigkeit im Schaltpunkt negativ ist, dann wäre die Nebenbedingung vor dem Schaltpunkt nicht erfüllt. Die Hamiltonfunktion (4.11), die adjungierten Differenzialgleichungen (4.12) und die Steuerung (4.13) bleiben unverändert. Enthält ein Optimalsteuerungsproblem eine lokale Ungleichungsnebenbedingung, so kann diese Unstetigkeiten der adjungierten Variablen und der Hamiltonfunktion hervorrufen. Die Gleichung (1.14) liefert:

$$\begin{aligned}\lim_{t\to t_b^-} \lambda^o(t) &= \lim_{t\to t_b^+} \lambda^o(t) + \mu_1 \left.\begin{pmatrix} \frac{\partial}{\partial x}(x(t)-0.2) \\ \frac{\partial}{\partial v}(x(t)-0.2) \end{pmatrix}\right|_{t=t_b} + \mu_2 \left.\begin{pmatrix} \frac{\partial}{\partial x} v(t) \\ \frac{\partial}{\partial v} v(t) \end{pmatrix}\right|_{t=t_b} \\ &= \lim_{t\to t_b^+} \lambda^o(t) + \mu_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \mu_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix}.\end{aligned}$$

Es gilt also:

$$\begin{aligned}\lim_{t\to t_b^+} \lambda_x^o &= \lim_{t\to t_b^-} \lambda_x^o - \mu_1, \\ \lim_{t\to t_b^+} \lambda_v^o &= \lim_{t\to t_b^-} \lambda_v^o - \mu_2.\end{aligned} \tag{4.17}$$

Da die Steuerung aber durch $u(t) = -\lambda_v(t)$ gegeben und außerdem stetig ist[12], darf λ_v im Zeitpunkt t_b keinen Sprung aufweisen und es muss gelten: $\mu_2 = 0$.

Man erhält somit das folgende Mehrpunktrandwertproblem:

- Differenzialgleichungen (siehe (4.9) und (4.12)):

$$\begin{aligned}\dot{x} &= v & \qquad \dot{v} &= u \\ \dot{\lambda}_x &= 0 & \qquad \dot{\lambda}_v &= -\lambda_x\end{aligned}$$

[12] Dies folgt aus (4.14).

- Steuerung (siehe (4.13)):

$$u = \begin{cases} -\lambda_v, & 0 \leq t \leq t_b \\ -\lambda_v, & t_b \leq t \leq 1 \end{cases} \quad = -\lambda_v$$

- Schaltpunkt:

$$t_b$$

- Rand- und Schaltbedingungen (siehe (4.9), (4.15) sowie (4.16)):

$$\begin{aligned} x(0) &= 0 & \qquad x(1) &= 0 \\ v(0) &= 1 & \qquad v(1) &= -1 \\ x(t_b) &= 0.2 & \qquad v(t_b) &= 0 \end{aligned}$$

- Sprung (siehe (4.17)):

$$\lambda_x(t_b^+) = \lambda_x(t_b^-) - \mu_1.$$

Dieses wird anschließend an die Syntax des Programms angepasst. Im Gegensatz zum unbeschränkten Problem kann hier die Dimension des Differenzialgleichungssystems nicht durch Ersetzen der Adjungierten λ_x durch eine zu bestimmende Konstante reduziert werden, weil λ_x aufgrund des Sprunges im Schaltpunkt t_b nur noch stückweise konstant ist.

- Umbenennung der Variablen:

$$\begin{aligned} y1 &:= x & \qquad y2 &:= v \\ y3 &:= \lambda_x & \qquad y4 &:= \lambda_v \\ u1 &:= u \end{aligned}$$

- Konstanten mit unbekanntem Wert:

$$c1 := -\mu_1$$

- Differenzialgleichungen:

$$\begin{aligned} \dot{y1} &= y2 & \qquad \dot{y2} &= u1 \\ \dot{y3} &= 0 & \qquad \dot{y4} &= -y3 \end{aligned}$$

- Steuerung:

$$u1 = -y4, \quad \text{(in beiden Intervallen)}$$

- Randbedingungen:

$$\begin{aligned} & y1_s & \qquad & y1_e \\ & y2_s - 1 & \qquad & y2_e + 1 \\ & y1_sp1 - 0.2 & \qquad & y2_sp1 \end{aligned}$$

- Sprung:

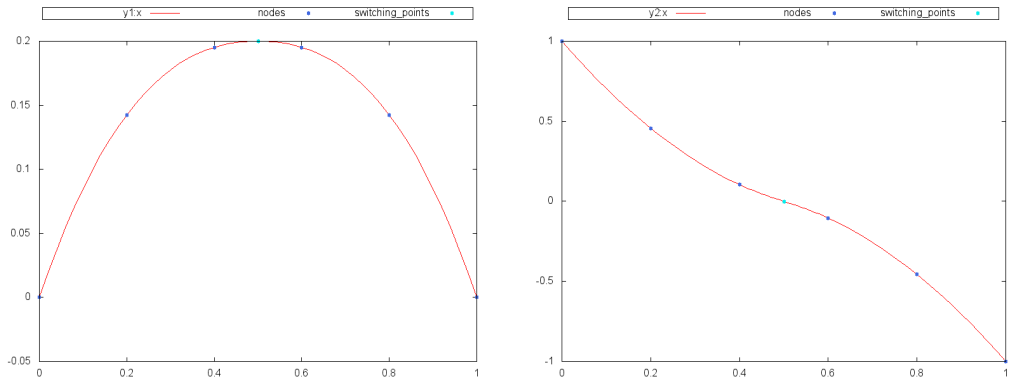
$$y3+ = c1 \quad \text{(durchzuführen im Schaltpunkt Nr. 1)}$$

Zur Lösung des Problems werden als Startschätzung die optimale Lösung des unbeschränkten Problems (Tabelle 4.6) sowie $t_b = 0.5$ und $c1 = 1$ und die gleichen Einstellungen wie beim unbeschränkten Problem verwendet. Die Abbruchbedingungen der Mehrzielmethode sind nach 2 Iterationen ($0.21s$) erfüllt und man erhält die in Tabelle 4.7 sowie den Abbildungen 4.4-4.6 dargestellte Lösung.[13]

t	y_1	y_2	y_3	y_4
0	0	1	4.8	3.2
0.2	0.1424	0.456	4.8	2.24
0.4	0.1952	0.104	4.8	1.28
0.6	0.1952	−0.104	4.8	1.28
0.8	0.1424	−0.456	4.8	2.24
1	0	−1	2	3.2

Schaltpunkt:	$t_b = 0.5$
gesuchte Konstante:	$c1 = -9.6$

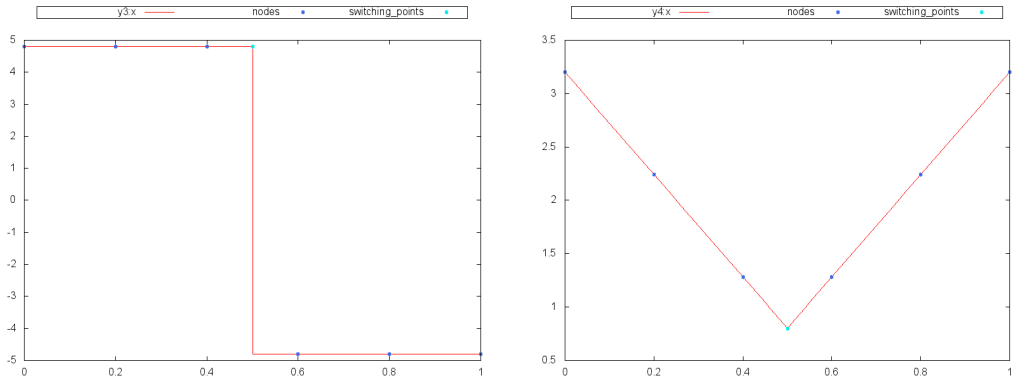
Tabelle 4.7: Min-Energy-Problem mit Berührpunkt: optimale Lösung



(a) y_1 (b) y_2

Abbildung 4.4: optimale Trajektorien der Zustandsvariablen

[13]Die Ausgabe des Programms mit deaktivierter Option "detailed solution"kann im Anhang A.3 auf S.122ff nachgelesen werden.



(a) y_3 (b) y_4

Abbildung 4.5: optimale Trajektorien der Adjungierten

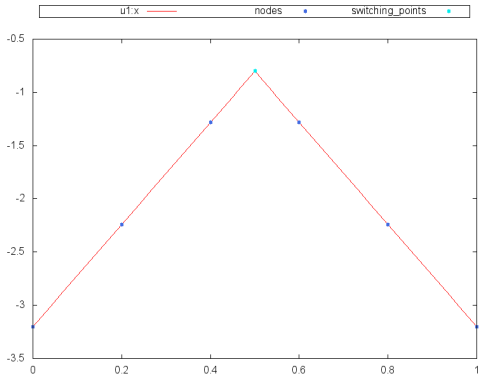


Abbildung 4.6: optimale Steuerung

4.2.3 Das Min-Energy-Problem mit Randstück

Man betrachte das Min-Energy-Problem erneut mit einer veränderten Konstanten $l := 0.1$, die Ungleichungsbedingung lautet also nun

$$s(t, x(t), v(t)) = x(t) - 0.1 \leq 0, \quad t \in [0, 1]. \tag{4.18}$$

Geht man weiterhin davon aus, dass das Problem genau einen Schaltpunkt enthält, so ändert sich an den aus dem Maximumprinzip folgenden notwendigen Bedingungen lediglich eine der Schaltbedingungen:

$$x(t_b) = 0.1.$$

Löst man dieses Problem, so wird wiederum nach 2 Iterationen eine Lösung ermittelt. Wie man in Abbildung 4.7 erkennen kann, erfüllt die Lösung die Beschränkung (4.18) zwar im Schaltpunkt $t_b = 0.5$, verletzt diese jedoch in Zeitintervallen links- und rechtsseitig des Schaltpunktes. Daraus kann man schließen, dass die optimale Lösung nun 2 Schaltpunkte t_1 und t_2 enthalten wird. Dabei wird die Ungleichungsbeschränkung (4.10) im Intervall $[t_1, t_2]$ mit Gleichheit erfüllt sein:

$$\begin{aligned} w(t, x, v) &< 0, \quad t \in [0, t_1[\cup]t_2, 1], \\ w(t, x, v) &= 0, \quad t \in [t_1, t_2]. \end{aligned} \tag{4.19}$$

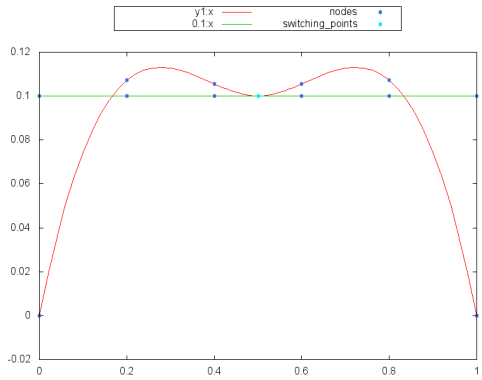


Abbildung 4.7: Zustandsvariable x und Ungleichungsbeschränkung (4.18)

Im Intervall $[t_1, t_2]$ wird die Beschränkung also wie eine globale Gleichungsbeschränkung behandelt.[14] Da die Beschränkung von der Ordnung 2 ist, ist diese äquivalent zu den 3 Bedingungen

$$w(t, x(t), v(t), u(t))|_{t=\tau} = x(\tau) - 0.1 = 0, \quad \tau \in [t_1, t_2], \tag{4.20}$$

$$\left.\frac{dw(t, x(t), v(t), u(t))}{dt}\right|_{t=\tau} = \dot{x}(\tau) = v(\tau) = 0, \quad \tau \in [t_1, t_2], \tag{4.21}$$

$$\frac{d^2 w(t, x(t), v(t), u(t))}{dt^2} = \dot{v}(t) = u(t) = 0, \quad t_1 \leq t \leq t_2. \tag{4.22}$$

Aus (4.20) und (4.21) folgen die Schaltbedingungen

$$\begin{aligned} x(t_1) - 0.1 &= 0, \\ v(t_1) &= 0. \end{aligned} \tag{4.23}$$

Die Hamiltonfunktion (4.11) sowie die adjungierten Differenzialgleichungen (4.12) bleiben unverändert. Daraus und aus der Bedingung (4.22) folgt für die Steuerung u:

$$u = \begin{cases} -\lambda_v, & 0 \leq t \leq t_1 \\ 0, & t_1 \leq t \leq t_2 \\ -\lambda_v, & t_2 \leq t \leq 1 \end{cases}. \tag{4.24}$$

Weiterhin folgt aufgrund der Stetigkeit der Steuerung[15] aus (4.24), dass in den Schaltpunkten die folgenden Bedingungen erfüllt sein müssen:

$$\lambda_v(t_1^-) = \lambda_v(t_2^+) = 0. \tag{4.25}$$

Zuletzt muss noch untersucht werden, ob die adjungierten Variablen Sprünge aufweisen. Diese können wahlweise an einem der beiden Schaltpunkte auf-

[14] Vergleiche dazu Abschnitt 1.3.
[15] Dies folgt aus (4.14).

treten.[16]

$$\begin{aligned}\lim_{t\to t_1^-}\lambda^o(t) &= \lim_{t\to t_1^+}\lambda^o(t) + \mu_1\left.\begin{pmatrix}\frac{\partial}{\partial x}w(t,x,v)\\ \frac{\partial}{\partial v}w(t,x,v)\end{pmatrix}\right|_{t=t_1} + \mu_2\left.\begin{pmatrix}\frac{\partial}{\partial x}\frac{d}{dt}w(t,x,v)\\ \frac{\partial}{\partial v}\frac{d}{dt}w(t,x,v)\end{pmatrix}\right|_{t=t_1}\\ &\quad + \mu_3\left.\begin{pmatrix}\frac{\partial}{\partial x}\frac{d^2}{dt^2}w(t,x,v)\\ \frac{\partial}{\partial v}\frac{d^2}{dt^2}w(t,x,v)\end{pmatrix}\right|_{t=t_1}\\ &= \lim_{t\to t_1^+}\lambda^o(t) + \mu_1\begin{pmatrix}1\\0\end{pmatrix} + \mu_2\begin{pmatrix}0\\1\end{pmatrix} + \mu_3\begin{pmatrix}0\\0\end{pmatrix}.\end{aligned}$$

Demzufolge weisen beide Adjungierten im Schaltpunkt t_1 einen Sprung auf und es gilt:

$$\begin{aligned}\lim_{t\to t_b^+}\lambda_x^o &= \lim_{t\to t_b^-}\lambda_x^o - \mu_1,\\ \lim_{t\to t_b^+}\lambda_v^o &= \lim_{t\to t_b^-}\lambda_v^o - \mu_2.\end{aligned} \tag{4.26}$$

Man erhält also das folgende bereits an die Syntax des Programms angepasste Mehrpunktrandwertproblem:

- Umbenennung der Variablen:

$$\begin{aligned} y1 &:= x & y2 &:= v\\ y3 &:= \lambda_x & y4 &:= \lambda_v\\ u1 &:= u \end{aligned}$$

- Konstanten mit unbekanntem Wert (Sprünge aus (4.26)):

$$c1 := -\mu_1 \qquad c2 := -\mu_2$$

- Differenzialgleichungen (siehe (4.9) sowie (4.12)):

$$\begin{aligned} \dot{y1} &= y2 & \dot{y2} &= u1\\ \dot{y3} &= 0 & \dot{y4} &= -y3 \end{aligned}$$

- 2 Schaltpunkte: t_1 und t_2
- Steuerung (siehe (4.24)):

$$u1 = \begin{cases} -y4, & 0 \le t \le t_1\\ 0, & t_1 \le t \le t_2\\ -y4, & t_2 \le t \le 1 \end{cases}$$

- Rand- und Schaltbedingungen (siehe (4.9), (4.23) sowie (4.25)):

$$\begin{aligned} &y1_s && y1_e\\ &y2_s - 1 && y2_e + 1\\ &y1_sp1 - 0.2 && y2_sp1\\ &y4_sp1 && y4_sp2 \end{aligned}$$

[16] O.B.d.A. wird hier der erste Schaltpunkt t_1 gewählt.

- Sprünge (siehe (4.26)):

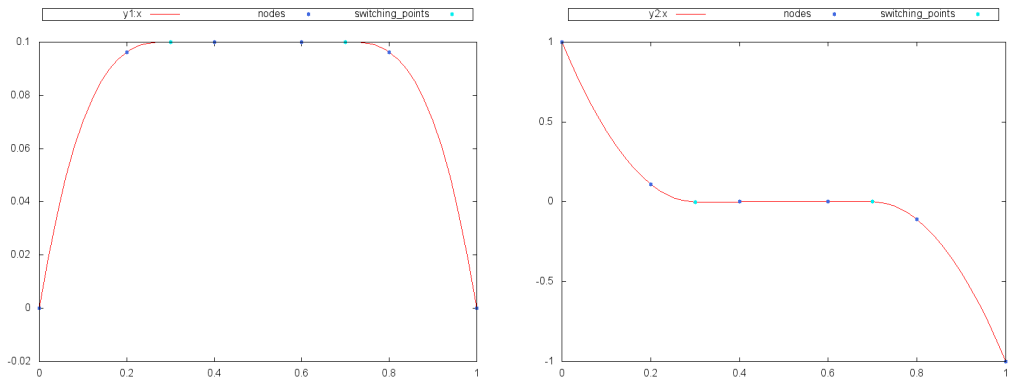
$$y3+ = c1 \quad \text{(durchzuführen im Schaltpunkt Nr. 1)}$$
$$y4+ = c2 \quad \text{(durchzuführen im Schaltpunkt Nr. 1)}$$

Zur Lösung des Problems werden als Startschätzung die optimale Lösung des unbeschränkten Problems (Tabelle 4.7) sowie $t_1 = 0.25$, $t_2 = 0.75$ und $c1 = c2 = 1$ verwendet. Des Weiteren bleiben die Einstellungen der Verfahren weitestgehend unverändert. Es wird lediglich statt des guten das schlechte Broyden-Update angewendet, da das Newtonverfahren bei Ersterem in diesem Fall nicht konvergiert. Die Abbruchbedingungen der Mehrzielmethode sind nach 7 Iterationen ($0.45s$) erfüllt und man erhält die in Tabelle 4.8 sowie den Abbildungen 4.8-4.10 dargestellte Lösung.[17]

t	y_1	y_2	y_3	y_4
0	0	1	4.8	3.2
0.2	0.1424	0.456	4.8	2.24
0.4	0.1952	0.104	4.8	1.28
0.6	0.1952	−0.104	4.8	1.28
0.8	0.1424	−0.456	4.8	2.24
1	0	−1	2	3.2

Schaltpunkte:	$t_1 = 0.3,\ t_2 = 0.7$
gesuchte Konstanten:	$c1 = -4.\overline{4},\ c2 = -8.\overline{8}$

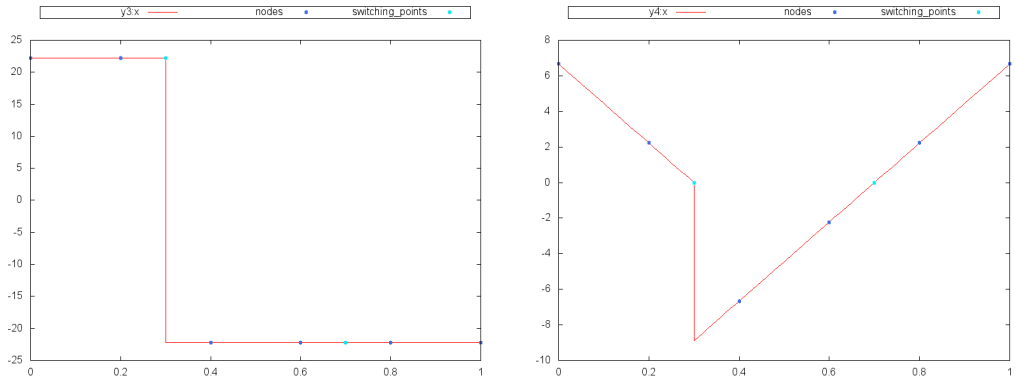
Tabelle 4.8: Min-Energy-Problem mit Randstück: optimale Lösung



(a) y_1 (b) y_2

Abbildung 4.8: optimale Trajektorien der Zustandsvariablen

[17] Die Ausgabe des Programms mit deaktivierter Option "detailed solution" kann im Anhang A.4 auf S.123ff nachgelesen werden.



(a) y_3 (b) y_4

Abbildung 4.9: optimale Trajektorien der Adjungierten

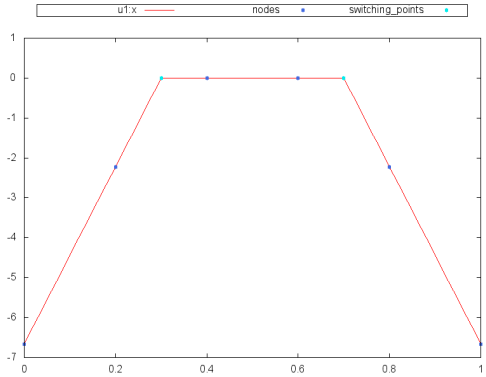


Abbildung 4.10: optimale Steuerung

4.3 Das Rayleigh-Problem

Im Folgenden soll das sogenannte Rayleigh-Problem untersucht werden. An diesem Beispiel sollen einerseits die Behandlung einer gemischten Steuer-Zustandsbeschränkung und andererseits die Funktionen des Programms, die die Lösung eines Optimalsteuerungsproblems erleichtern können, demonstriert werden.

Man betrachte das folgende Optimalsteuerungsproblem:

$$I(t,x,v,u) := \int_0^T u(t)^2 + x_1(t)^2 dt \to \min, \tag{4.27}$$

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -x_1 + 1.4x_2 - 0.14x_2^3 + 4u, \end{aligned} \tag{4.28}$$

$$x_1(0) = x_2(0) = -5, \tag{4.29}$$

$$T = 4.5,$$

$$s(t,x(t),v(t)) = u(t) + \frac{1}{6}x_1(t) \leq 0, \;\; t \in [0,T]. \tag{4.30}$$

Das Problem soll zunächst in seiner unrestringierten Form, d.h. ohne Berücksichtigung der Beschränkung (4.30), gelöst werden. Dazu werden die Hamil-

tonfunktion sowie die adjungierten Differenzialgleichungen aufgestellt:

$$H(t,x,\lambda,u) := u^2 + x_1^2 + \lambda_1 x_2 + \lambda_2(-x_1 + 1.4x_2 - 0.14x_2^3 + 4u), \tag{4.31}$$

$$\begin{aligned} \dot{\lambda}_1 &:= -\frac{\partial H}{\partial x_1} = -2x_1 + \lambda_2, \\ \dot{\lambda}_2 &:= -\frac{\partial H}{\partial x_2} = -\lambda_1 - 1.4\lambda_2 + 0.42x_2^2\lambda_2. \end{aligned} \tag{4.32}$$

Aus den Orthogonalitätsbedingungen folgen Endbedingungen für die Adjungierten:

$$\lambda_1(T) = \lambda_2(T) = 0. \tag{4.33}$$

Die Steuerung u ergibt sich durch Nullsetzen der Ableitung der Hamiltonfunktion (4.31):

$$H_u = 2u + 4\lambda_2 = 0 \Rightarrow u = -2\lambda_2, \tag{4.34}$$

$$H_{uu} = 1 > 0. \tag{4.35}$$

Aus (4.35) folgt, dass die Hamiltonfunktion bezüglich u strikt konvex ist und daher ein eindeutig bestimmtes Minimum besitzt. Die Hamiltonfunktion ist also regulär und die Steuerung damit stetig. Diese Aussage wird bei der Lösung des restringierten Problems eine Rolle spielen.

Zusätzlich zu den Differenzialgleichungen für die Zustands- und adjungierten Variablen wird dem Problem eine weitere Differenzialgleichung (4.36) hinzugefügt, um die Lösungen des Problems anhand des optimalen Zielfunktionswertes vergleichen zu können:

$$x_3(t) := \int_0^t u(\tau)^2 + x_1(\tau)^2 d\tau \Rightarrow \dot{x}_3 = u^2 + x_1^2,\ x_3(0) = 0. \tag{4.36}$$

Für die eigentliche Lösung des unrestringierten Rayleigh-Problems wird diese Differenzialgleichung nicht benötigt.

Man erhält demzufolge das folgende, bereits an die Syntax des Programms angepasste, Randwertproblem:

- Umbenennung der Variablen:

$$\begin{aligned} y1 &:= x_1 & y2 &:= x_2 \\ y3 &:= \lambda_1 & y4 &:= \lambda_2 \\ y5 &:= x_3 & u1 &:= u \end{aligned}$$

- Differenzialgleichungen (siehe (4.28), (4.32) sowie (4.36)):

$$\begin{aligned} \dot{y1} &= y2 \\ \dot{y2} &= -y1 + 1.4 \cdot y2 - 0.14 \cdot y2\hat{}3 + 4 \cdot u1 \\ \dot{y3} &= -2 \cdot y1 + y4 \\ \dot{y4} &= -y3 - 1.4 \cdot y4 + 0.42 \cdot y2\hat{}2 \cdot y4 \\ \dot{y5} &= u1\hat{}2 + x1\hat{}2 \end{aligned}$$

- Randbedingungen (siehe (4.29) und (4.33)):

$$\begin{array}{ll} y1_s + 5 & y2_s + 5 \\ y3_e & y4_e \\ y5_s & \end{array}$$

- Steuerung (siehe (4.34)):

$$u1 = -2 \cdot y4$$

Das Randwertproblem wird mit den folgenden Startwerten (Tabelle 4.9) und Einstellungen gelöst:

t	x_1	x_2	λ_1	λ_2	x_3
0	0	0	0	0	0
0.5	0	0	0	0	0
1.5	0	0	0	0	0
2.5	0	0	0	0	0
3.5	0	0	0	0	0
4.5	0	0	0	0	0

Tabelle 4.9: Rayleigh-Problem (unrestringiert): Startwerte

- Newtonverfahren:
  - Maximale Anzahl der Newtoniterationen: 20
  - geforderte Genauigkeit der Lösung: 10^{-4}
  - Relaxation des Newtonverfahrens: ja
  - Broyden-Update: nein
- Anfangswertproblemlöser:
  - geforderte Genauigkeit: 10^{-5}
  - Startschrittweite: 1
  - minimale Schrittweite: 10^{-6}
  - maximale Zahl der Zeitschritte: 500
  - maximale Anzahl der Verkleinerungen der Schrittweite in einem Zeitschritt: 10

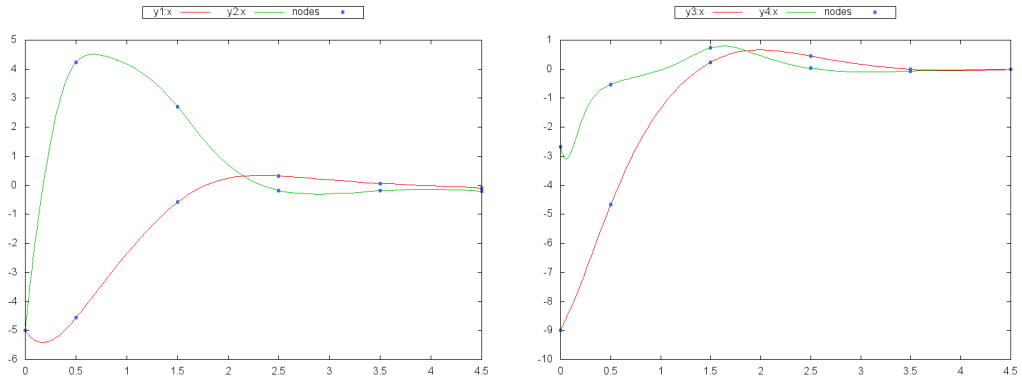
Nach 10 Iterationen und rund 28 Sekunden erhält man die in Tabelle 4.10 sowie den Abbildungen (4.11) und (4.12) dargestellte Lösung.[18]

Man untersucht nun, ob die Lösung des unrestringierten Rayleigh-Problems die Beschränkung (4.30) erfüllt. Anhand der Abbildung 4.13 kann

t	x_1	x_2	λ_1	λ_2	x_3
0	-5	-5	-9.001243	-2.672616	0
0.5	-4.564134	4.232434	-4.674163	-5.255322	20.64724
1.5	-0.5870499	2.714466	0.2336765	0.7292466	28.49424
2.5	0.327439	-0.1756616	0.4540488	0.0268984	29.67675
3.5	0.06671313	-0.1932589	-0.0012403	-0.06751068	29.74243
4.5	-0.09571141	-0.2043943	0	0	29.74893

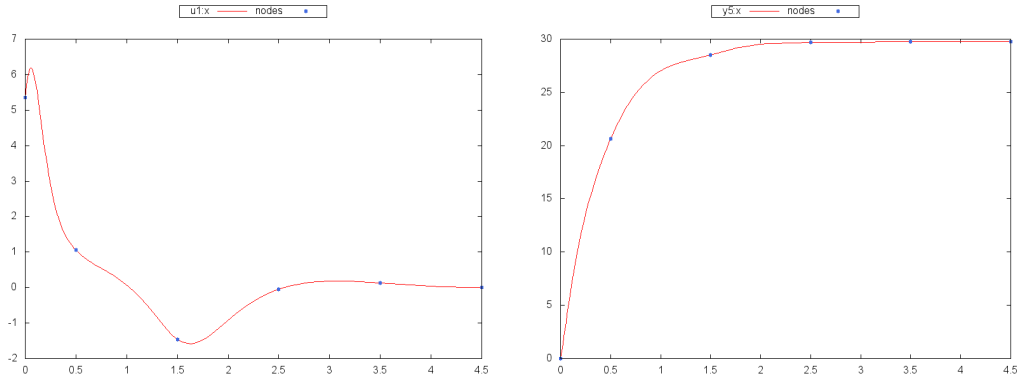
optimaler Zielfunktionswert: 29.74893

Tabelle 4.10: Rayleigh-Problem (unrestringiert): optimale Lösung



(a) y_1 (b) y_2

Abbildung 4.11: optimale Trajektorien der Zustandsvariablen & Adjungierten



(a) y_3 (b) y_4

Abbildung 4.12: optimale Steuerung und Zielfunktional

man erkennen, dass die Beschränkung am Anfang sowie am Ende des betrachteten Zeitintervalls verletzt ist. Demzufolge muss die optimale Lösung

[18]Die Ausgabe des Programms mit deaktivierter Option "detailed solution"kann im An-

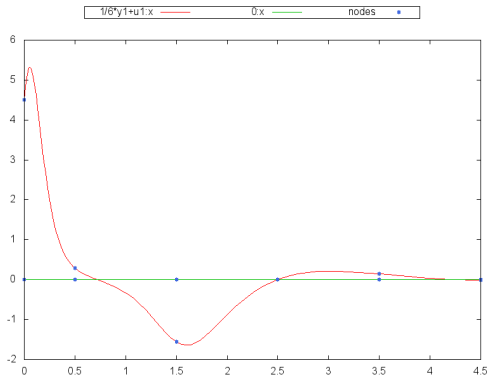


Abbildung 4.13: Nebenbedingung $u + \frac{1}{6}x_1$ (unrestringiert)

des restringierten Problems Schaltpunkte enthalten. Da die Nebenbedingung etwa im Intervall $[0, 0.75]$ nicht erfüllt ist, wird ein Schaltpunkt t_1 mit der Startschätzung 0.75 eingefügt. Die Randsteuerung im ersten Intervall ergibt sich durch Umstellen der Beschränkung (4.30) und man erhält die folgende Schaltstruktur:

$$u(t) = \begin{cases} -\frac{1}{6}x_1(t), & t \in [0, t_1] \\ -2\lambda_2(t), & t \in [t_1, 4.5] \end{cases} . \tag{4.37}$$

Des Weiteren wird eine Schaltbedingung benötigt. Da die Steuerung, wie bereits erwähnt, aufgrund der Regularität der Hamiltonfunktion bezüglich der Steuerung stetig ist, gilt im Schaltpunkt $-\frac{1}{6}x_1(t_1) = -2\lambda_2(t_1)$. Somit muss die folgende, bereits an die Syntax des Programms angepasste, Schaltbedingung im Zeitpunkt t_1 erfüllt sein:

$$2 \cdot y4_sp1 - 1/6 \cdot y1_sp1. \tag{4.38}$$

Das unrestringierte Rayleigh-Problem wird also um einen Schaltpunkt t_1, eine Randsteuerung im ersten Steuerungsintervall (4.37) sowie eine im Schaltpunkt zu erfüllende Bedingung (4.38) erweitert und als Startschätzung die Lösung des unrestringierten Problems (Tabelle 4.10) verwendet. Die Abbruchkriterien des Newtonverfahrens sind nach 17 Iterationen ($\approx$ 42 Sekunden) erfüllt und die in Tabelle 4.11 dargestellte Lösung wurde ermittelt.[19]

Betrachtet man nun die in der Abbildung 4.14 geplottete Beschränkung (4.30), so erkennt man, dass diese weiterhin nicht im gesamten Zeitintervall erfüllt ist. Demzufolge muss ein weiterer Schaltpunkt t_2 eingefügt werden. Analog zu den vorhergehenden Ausführungen verändert sich erneut die Schaltstruktur (4.39) und eine weitere Schaltbedingung (4.40), die im zweiten

hang A.5 auf S.125ff nachgelesen werden.

[19] Die Ausgabe des Programms mit deaktivierter Option "detailed solution"kann im Anhang A.6 auf S.127ff nachgelesen werden.

t	x_1	x_2	λ_1	λ_2	x_3
0	-5	-5	-12.809	-4.54501	0
0.5	-6.16449	-0.0579288	-10.2002	-3.94572	17.1837
1.5	-2.6567	4.28214	-1.68464	-0.107894	41.6148
2.5	0.198893	0.967521	0.657963	0.544572	44.4377
3.5	0.25505	-0.329132	0.133965	-0.0646212	44.7437
4.5	-0.224029	-0.747758	0	0	44.7717

Schaltpunkt: $t_1 = 1.2815$
optimaler Zielfunktionswert: 44.7717

Tabelle 4.11: Rayleigh-Problem (1 Schaltpunkt): optimale Lösung

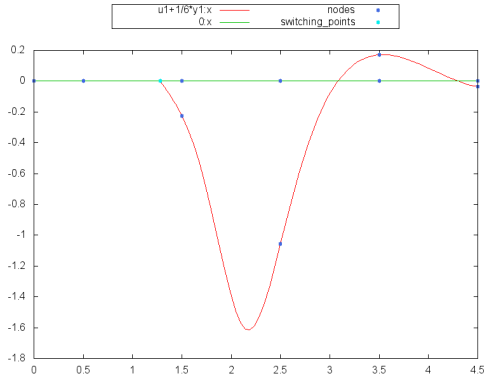


Abbildung 4.14: Nebenbedingung $u + \frac{1}{6}x_1$ (1 Schaltpunkt)

Schaltpunkt erfüllt sein muss, wird hinzugefügt:

$$u(t) = \begin{cases} -\frac{1}{6}x_1(t), & t \in [0, t_1] \\ -2\lambda_2(t), & t \in [t_1, t_2] \\ -\frac{1}{6}x_1(t), & t \in [t_2, 4.5] \end{cases}, \tag{4.39}$$

$$2 \cdot y4_sp2 - 1/6 \cdot y1_sp2. \tag{4.40}$$

Als Startschätzung wird die optimale Lösung des Problems mit einem Schaltpunkt (Tabelle 4.11) und $t_2 = 3$ benutzt. Das Programm liefert nach 3 Iterationen (6.19 Sekunden) eine Lösung (Tabelle 4.12)[20], die die Beschränkung (4.30) im gesamten Zeitintervall $[0, 4.5]$ erfüllt (siehe Abbildung 4.15).

Diese Lösung muss aber nicht die optimale Lösung des restringierten Rayleigh-Problems sein, da nicht ausgeschlossen ist, dass die optimale Lösung durchaus noch einen weiteren Schaltpunkt nahe des Endzeitpunktes besitzen kann. Dies würde bedeuten, dass die Randsteuerung nicht bis zum Endzeitpunkt gilt, sondern stattdessen noch ein weiteres Steuerungsintervall mit einer freien Steuerung existiert. Diese Vermutung legt die Abbildung 4.14

[20]Die Ausgabe des Programms mit deaktivierter Option "detailed solution"kann im Anhang A.7 auf S.129ff nachgelesen werden.

t	x_1	x_2	λ_1	λ_2	x_3
0	-5	-5	-12.7933	-4.53416	0
0.5	-6.16449	-0.0579288	-10.1752	-3.93181	17.1837
1.5	-2.65765	4.27369	-1.65284	-0.0958034	41.6123
2.5	0.150564	0.927777	0.743556	0.516169	44.4926
3.5	0.335036	-0.212686	0.182438	-0.120357	44.7551
4.5	-0.393869	-1.24987	0	0	44.8059

Schaltpunkte: $t_1 = 1.26917,\ t_2 = 2.98404$
optimaler Zielfunktionswert: 44.8059

Tabelle 4.12: Rayleigh-Problem (2 Schaltpunkte): optimale Lösung

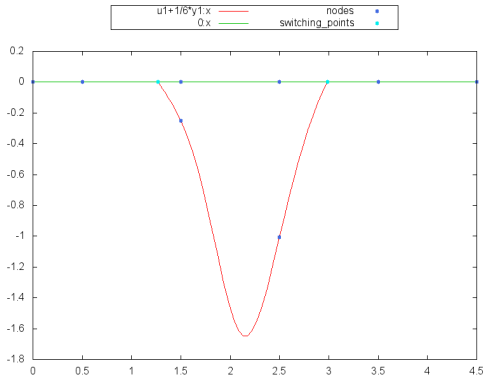


Abbildung 4.15: Nebenbedingung $u + \frac{1}{6}x_1$ (2 Schaltpunkte)

nahe. Das Problem ist also um einen weiteren Schaltpunkt t_3 zu erweitern. Damit einhergehend wird wiederum sowohl die Schaltstruktur verändert als auch eine zusätzliche Schaltbedingung hinzugefügt:

$$u(t) = \begin{cases} -\frac{1}{6}x_1(t), & t \in [0, t_1] \\ -2\lambda_2(t), & t \in [t_1, t_2] \\ -\frac{1}{6}x_1(t), & t \in [t_2, t_3] \\ -2\lambda_2(t), & t \in [t_3, 4.5] \end{cases}, \tag{4.41}$$

$$2 \cdot y4_sp3 - 1/6 \cdot y1_sp3. \tag{4.42}$$

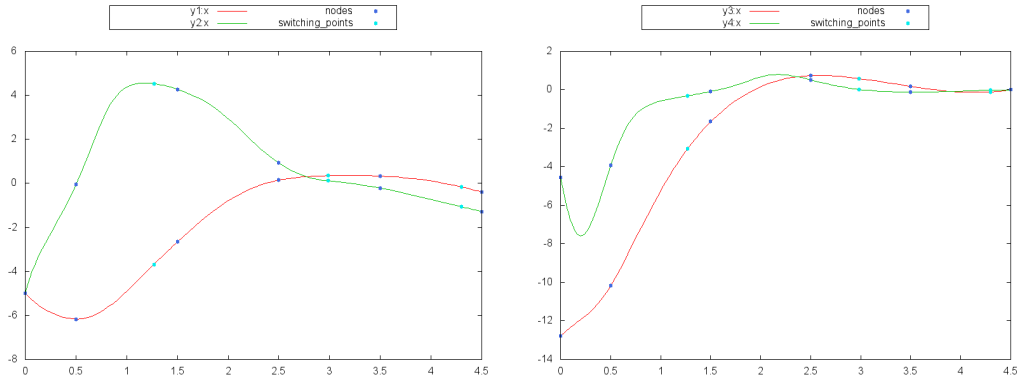
Nach dem Lösen dieses Problems muss untersucht werden, ob die Beschränkung (4.30) weiterhin erfüllt ist und der optimale Zielfunktionswert darüber hinaus auch verbessert wurde. Das Problem wird nun mit der optimalen Lösung des restringierten Problems mit 2 Schaltpunkten (Tabelle 4.12) als Startwerte sowie der Startschätzung $t_3 = 4.2$ für den Schaltpunkt gelöst. Nach 3 Iterationen (6.12 Sekunden) wird die in Tabelle 4.13 sowie den Abbildungen 4.16 und 4.17 dargestellte Lösung[21] erhalten.

[21] Die Ausgabe des Programms mit deaktivierter Option "detailed solution" kann im Anhang A.8 auf S.131ff nachgelesen werden.

t	x_1	x_2	λ_1	λ_2	x_3
0	-5	-5	-12.7933	-4.53416	0
0.5	-6.16449	-0.0579288	-10.1752	-3.9318	17.1837
1.5	-2.65765	4.27369	-1.65283	-0.0957951	41.6123
2.5	0.150513	0.927707	0.743619	0.516151	44.4927
3.5	0.335046	-0.212494	0.182545	-0.120419	44.7551
4.5	-0.395708	-1.28033	0	0	44.8056

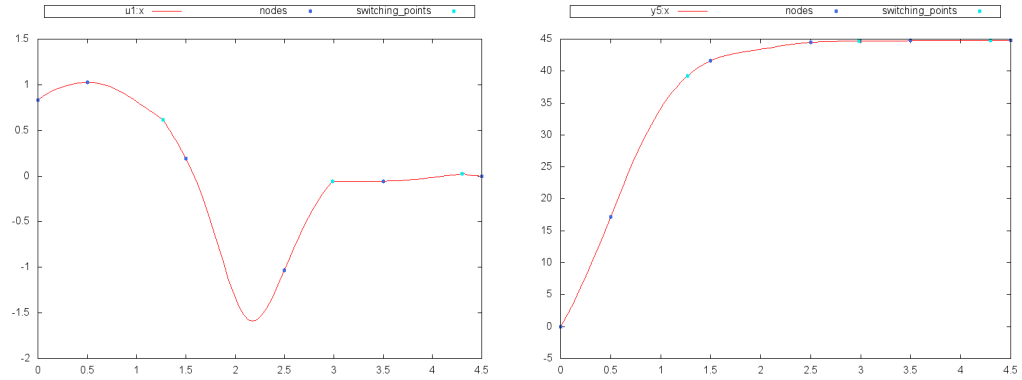
Schaltpunkte: $t_1 = 1.26916,\ t_2 = 2.98398,\ t_3 = 4.29926$
optimaler Zielfunktionswert: 44.8056

Tabelle 4.13: Rayleigh-Problem (3 Schaltpunkte): optimale Lösung



(a) y_1 (b) y_2

Abbildung 4.16: optimale Trajektorien der Zustandsvariablen & Adjungierten



(a) y_3 (b) y_4

Abbildung 4.17: optimale Steuerung und Zielfunktionswert

Betrachtet man die geplottete Beschränkung (4.30) in Abbildung 4.18, so kann man erkennen, dass diese im gesamten Intervall $[0, 4.5]$ erfüllt ist.

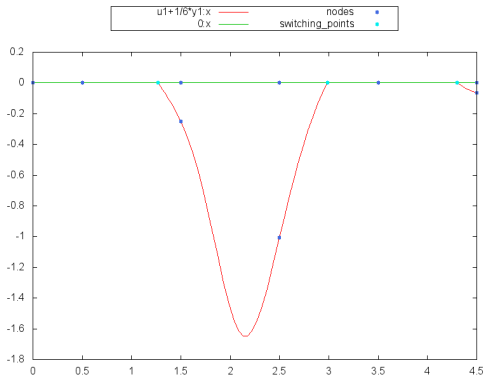


Abbildung 4.18: Nebenbedingung $u + \frac{1}{6}x_1$ (3 Schaltpunkte)

Die Lösung ist demzufolge genauso zulässig wie diejenige, die lediglich zwei Schaltpunkte enthält. Darüber hinaus hat wurde der Zielfunktionswert auch leicht verbessert[22], sodass die Lösung mit zwei Schaltpunkten nicht optimal sein kann.

4.4 Das Re-Entry-Problem

Das folgende Problem entstammt der Raumfahrt und modelliert das optimale Bremsmanöver eines Raumfahrzeuges in der Erdatmosphäre.[23] Bei der Modellierung des Problems wird vereinfachend angenommen, dass die Erde eine ruhende Kugel ist, die Flugbahn in einer Großkreisebene liegt und keine obere Grenze für die Belastbarkeit der Astronauten existiert. Die Bahnkoordinaten eines Apollo-Raumfahrzeugs genügen beim Flug durch die Lufthülle der Erde den folgenden Differenzialgleichungen:

$$\begin{aligned}
\dot{v} &= -\frac{S\rho v^2}{2m} C_W(u) - \frac{g\sin(\gamma)}{(1+\xi)^2}, \\
\dot{\gamma} &= \frac{S\rho v}{2m} C_A(u) + \frac{v\cos(\gamma)}{R(1+\xi)} - \frac{g\cos(\gamma)}{v(1+\xi)^2}, \\
\dot{\xi} &= \frac{v\sin(\gamma)}{R}.
\end{aligned} \tag{4.43}$$

Die Variablen sowie deren Bedeutung und die gegebenen Konstanten des Problems sind in den Tabellen 4.14 bzw. 4.15 dargestellt. Den größten Einfluss haben dabei aufgrund der hohen Geschwindigkeit die Luftkräfte. Dies sind die mit $C_W(u)$ und $C_A(u)$ multiplizierten Terme. Diese Kräfte lassen sich über den Steuerungsparameter u beeinflussen. Die mit der Gravitationskonstanten g multiplizierten Glieder stellen die auf das Raumfahrzeug wirkenden Gravitationskräfte der Erde dar. Die Differenzialgleichungen müssen die fol-

[22] 44.8056 gegenüber 44.8059.
[23] Dieses Problem wurde aus [18] S.214ff. entnommen.

Bezeichnung	Bedeutung
v	Tangentialgeschwindigkeit
γ	Bahnneigungswinkel
h	Höhe über Erdoberfläche
R	Erdradius
$\xi = h/R$	normalisierte Höhe
$\rho = \rho_0 e^{-\beta R \xi}$	Luftdichte
$C_W(u) = 1.174 - 0.9\cos(u)$	aerodynamischer Widerstandskoeffizient
$C_A(u) = 0.6\sin(u)$	aerodynamischer Auftriebskoeffizient
u	Steuerung (beliebig zeitabhängig wählbar)
g	Erdbeschleunigung
S/m	Frontfläche/Fahrzeugmasse

Tabelle 4.14: Re-Entry-Problem: Variablen

Konstante	Wert
R	$209(\widehat{=}\ 209 \cdot 10^5 ft)$
β	4.26
ρ_0	0.002704
g	$3.2172 \cdot 10^{-4}$
S/m	53200

Tabelle 4.15: Re-Entry-Problem: Konstanten

genden Randbedingungen erfüllen:

$$\begin{aligned} v(0) &= 0.36(\widehat{=}\ 36000 ft/sec), & v(T) &= 0.27(\widehat{=}\ 27000 ft/sec), \\ \gamma(0) &= -8.1^\circ \frac{\pi}{180^\circ}, & \gamma(T) &= 0, \\ \xi(0) &= \frac{4}{R}, & \xi(T) &= \frac{2.5}{R}. \end{aligned} \tag{4.44}$$

Des Weiteren ist die beim Flug durch die Lufthülle der Erde auftretende Aufheizung des Raumfahrzeugs durch das zu minimierende Integral

$$J = \int_0^T 10 v^3 \sqrt{\rho}\, dt \tag{4.45}$$

gegeben. Die für den Flug durch die Lufthülle benötigte Zeit ist unbekannt, die Endzeit T ist dementsprechend frei.

Die Hamiltonfunktion sowie die adjungierten Differenzialgleichungen ergeben sich dann wie in (4.46) bzw. (4.47)-(4.49) angegeben.

$$\begin{aligned} H(t, v, \gamma, \xi, \lambda_v, \lambda_\gamma, \lambda_\xi) := 10 v^3 \sqrt{\rho} + \lambda_v \left(-\frac{S\rho v^2}{2m} C_W(u) - \frac{g \sin(\gamma)}{(1+\xi)^2} \right) \\ + \lambda_\gamma \left(\frac{S\rho v}{2m} C_A(u) + \frac{v\cos(\gamma)}{R(1+\xi)} - \frac{g\cos(\gamma)}{v(1+\xi)^2} \right) + \lambda_\xi \left(\frac{v \sin(\gamma)}{R} \right) \end{aligned} \tag{4.46}$$

$$\begin{aligned}\dot{\lambda}_v := -\frac{\partial H}{\partial v} &= -30v^2\sqrt{\rho} + \lambda_v\left(\frac{S\rho v}{m}C_W(u)\right) - \lambda_\xi\frac{\sin\gamma}{R} \\ &- \lambda_\gamma\left(\frac{S\rho}{2m}C_A(u) + \frac{\cos\gamma}{R(1+\xi)} + \frac{g\cos\gamma}{v^2(1+\xi)^2}\right)\end{aligned} \tag{4.47}$$

$$\begin{aligned}\dot{\lambda}_\gamma := -\frac{\partial H}{\partial \gamma} &= \lambda_v\frac{g\cos\gamma}{(1+\xi)^2} + \lambda_\gamma\left(\frac{v\sin\gamma}{R(1+\xi)} - \frac{g\sin\gamma}{v(1+\xi)^2}\right) \\ &- \lambda_\xi\frac{v\cos\gamma}{R}\end{aligned} \tag{4.48}$$

$$\begin{aligned}\dot{\lambda}_\xi := -\frac{\partial H}{\partial \xi} &= 5v^3\beta R\sqrt{\rho} - \lambda_v\left(\frac{Sv^2\beta R\rho}{4m}C_W(u) + 2\frac{g\sin\gamma}{(1+\xi)^3}\right) \\ &+ \lambda_\gamma\left(\frac{Sv\beta R\rho}{4m}C_A(u) + \frac{v\cos\gamma}{R(1+\xi)^2} - 2\frac{g\cos\gamma}{v(1+\xi)^3}\right)\end{aligned} \tag{4.49}$$

Die partielle Ableitung der Hamiltonfunktion (4.46) nach u liefert durch Nullsetzen und Auflösen nach u die optimale Steuerung:

$$\begin{aligned}\frac{\partial H}{\partial u} &= -0.9\frac{S\rho^2}{v}\lambda_v\sin u + 0.6\frac{S\rho}{v}\lambda_\gamma\cos u \\ &= -\frac{S\rho}{2m}v\,(0.9v\lambda_v\sin u - 0.6\lambda_\gamma\cos u) = 0 \\ &= \underbrace{-\frac{S\rho_0}{2m}e^{-\beta R\xi}v}_{\neq 0}\,(0.9v\lambda_v\sin u - 0.6\lambda_\gamma\cos u) = 0 \\ &\Rightarrow\ 0.9v\lambda_v\sin u = 0.6\lambda_\gamma\cos u \\ &\Rightarrow \tan u = \frac{0.6\lambda_\gamma}{0.9v\lambda_v} \\ &\Rightarrow\ u = \arctan\left(\frac{2\lambda_\gamma}{3v\lambda_v}\right)\end{aligned} \tag{4.50}$$

Da für die Zustandsvariablen v, γ und ξ jeweils Anfangs- und Endbedingung vorgegeben sind, treten für die Adjungierten λ_v, λ_γ und λ_ξ keine weiteren Bedingungen auf.

Des Weiteren ist, wie bereits erwähnt, der Endzeitpunkt T frei. Daher muss im Endzeitpunkt T die Transversalitätsbedingung (4.51) erfüllt sein.

$$H(T^o, v^o(T^o), \gamma^o(T^o), \xi^o(T^o), u^o(T^o), \lambda_v^o(T^o), \lambda_\gamma^o(T^o), \lambda_\xi^o(T^o)) = 0 \tag{4.51}$$

Außerdem wird das Optimalsteuerungsproblem auf das feste Zeitintervall $[0, 1]$ normiert.[24] Dazu werden die rechten Seiten der Differenzialgleichungen mit der Endzeit T multipliziert und dem Problem die zusätzliche unbekannte Konstante T hinzugefügt.

[24] Vergleiche dazu die Ausführungen auf S.8f.

Das vorliegende Optimalsteuerungsproblem wurde damit auf ein Randwertproblem mit 6 Differenzialgleichungen (4.43), (4.47), (4.48) und (4.49), 7 Randbedingungen (4.44) und (4.51), einer unbekannten Konstanten T sowie einer Steuerung (4.50) zurückgeführt. Eine geschlossen angebbare Lösung für dieses Problem existiert nicht, stattdessen müssen numerische Methoden angewandt werden. Weiterhin existieren aufgrund der Gefährlichkeit des Bremsmanövers nur für einen sehr schmalen Bereich von Randdaten differenzierbare Lösungen des Problems.[25]

Um dieses Randwertproblem mit dem vorliegenden Programm lösen zu können, werden zunächst einige Anpassungen am Problem vorgenommen:[26]

- Umbenennung der Variablen:

$$\begin{aligned} y1 &:= v & y2 &:= \gamma \\ y3 &:= \xi & y4 &:= \lambda_v \\ y5 &:= \lambda\gamma & y6 &:= \lambda_\xi \\ u1 &:= u \end{aligned}$$

- Einführung von Konstanten mit bekanntem Wert:[27]

$$\begin{aligned} p1 &:= S/m = 53200 & p2 &:= \rho_0 = 0.002704 \\ p3 &:= R = 209 & p4 &:= \beta = 4.26 \\ p5 &:= g = 0.00032172 \end{aligned}$$

- unbekannte Konstante:

$$c1 := T$$

Für die Startwerte (Tabelle 4.16) wird eine Schätzung dieser anhand der Abbildungen der Lösung in [18] verwendet. Die folgenden Einstellungen für das Newtonverfahren bzw. den Anfangswertproblemlöser werden benutzt.

- Newtonverfahren:
  - Maximale Anzahl der Newtoniterationen: 10
  - geforderte Genauigkeit der Lösung: 10^{-4}
  - Relaxation des Newtonverfahrens: ja
  - Broyden-Update: ja, schlecht
- Anfangswertproblemlöser:

[25] Vergleiche dazu [18] S.217.

[26] Für die vollständige Angabe der angepassten Eingabedaten des Problems vergleiche man (A.9) auf S.132ff.

[27] Dies erleichtert die Eingabe der Daten, da statt der Zahlenwerte lediglich die Bezeichnungen der Konstanten angegeben werden müssen.

t	v	γ	ξ	λ_v	λ_γ	λ_ξ
0	0.36	-0.14	4/209	-0.5	-0.7	-2.3
0.09	0.36	-0.122	3/209	-0.45	-0.6	-2.2
0.27	0.32	0	1.8/209	-0.4	-0.2	-13.5
0.5	0.285	0.017	2.3/209	-0.6	0.3	-4.5
0.75	0.28	0.01	2.4/209	-0.9	0.7	-7
1	0.27	0	2.5/209	-1.4	1.4	-13

unbekannte Konstante: $T = 225$

Tabelle 4.16: Re-Entry-Problem: Startlösung

- geforderte Genauigkeit: 10^{-6}
- Startschrittweite: 1
- minimale Schrittweite: 10^{-6}
- maximale Zahl der Zeitschritte: 500
- maximale Anzahl der Verkleinerungen der Schrittweite in einem Zeitschritt: 30

Wird statt des schlechten das gute Broyden-Update verwendet, konvergiert das Newtonverfahren deutlich langsamer.[28] Wenn als Lösungsverfahren für die Anfangswertprobleme beispielsweise Dormand-Prince 5(4) ausgewählt wird oder die geforderte Genauigkeit des Verfahrens verringert wird, so konvergiert die Mehrzielmethode nicht.[29] Dies zeigt, dass das Re-Entry-Problem hohe Anforderungen an die Genauigkeit der verwendeten Methoden stellt. Hier macht sich die Vielzahl der zur Verfügung gestellten Optionen bezahlt, da aufgrund dieser gebotenen Konfigurationsmöglichkeiten der Methoden eine Lösung des Problems ermittelt werden kann.

Nach 9 Iterationen (ca. 63 Sekunden) wird eine Lösung (Tabelle 4.17) des Randwertproblems gefunden.[30] Die Abbildungen (4.19) - (4.22) zeigen die optimalen Trajektorien der Zustandsvariablen, Adjungierten und der Steuerung.

[28] Es werden ca. 110 Sekunden benötigt, um eine Lösung zu finden.

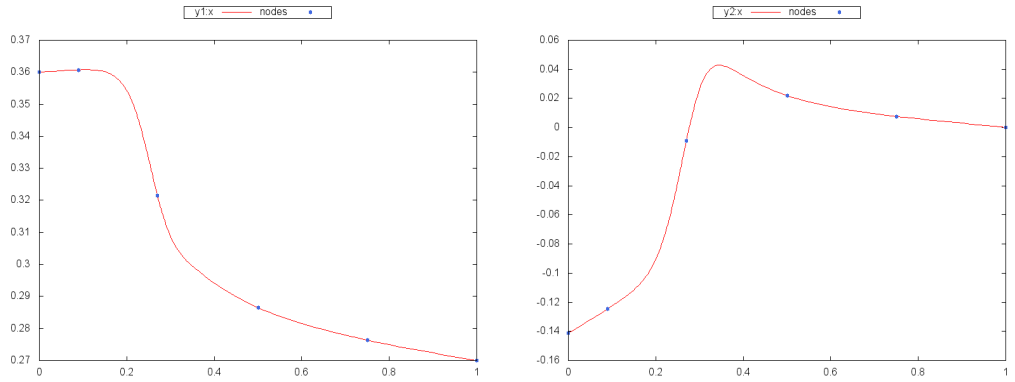
[29] Es treten nan-Fehler auf, d.h. unendlich große Werte oder beispielsweise eine Division durch Null. Dies tritt in der Regel auf, wenn die Genauigkeit des Verfahrens zu gering ist.

[30] Die Ausgabe des Programms mit aktivierter Option "detailed solution" kann im Anhang A.9 auf S.132ff nachgelesen werden.

t	v	γ	ξ	λ_v	λ_γ	λ_ξ
0	0.36	−0.14137	0.019139	−0.50814	−0.70862	−3.1193
0.09	0.36074	−0.12461	0.01454	−0.45558	−0.60419	−3.0309
0.27	0.32154	−0.008961	0.008434	−0.42841	−0.24147	−12.269
0.5	0.28653	0.022006	0.010661	−0.50764	0.25465	−3.943
0.75	0.27635	0.007718	0.011682	−0.7106	0.58978	−5.8587
1	0.27	0	0.011962	−1.19617	1.17008	−11.2111

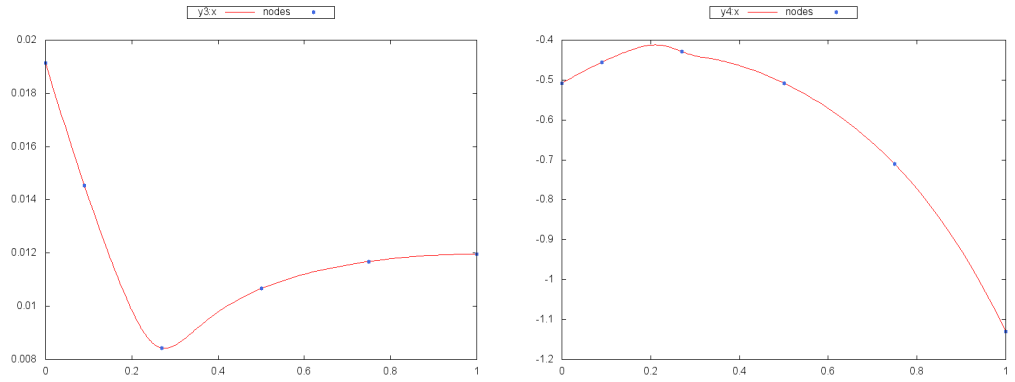
unbekannte Konstante: $T = 224.637$

Tabelle 4.17: Re-Entry-Problem: optimale Lösung



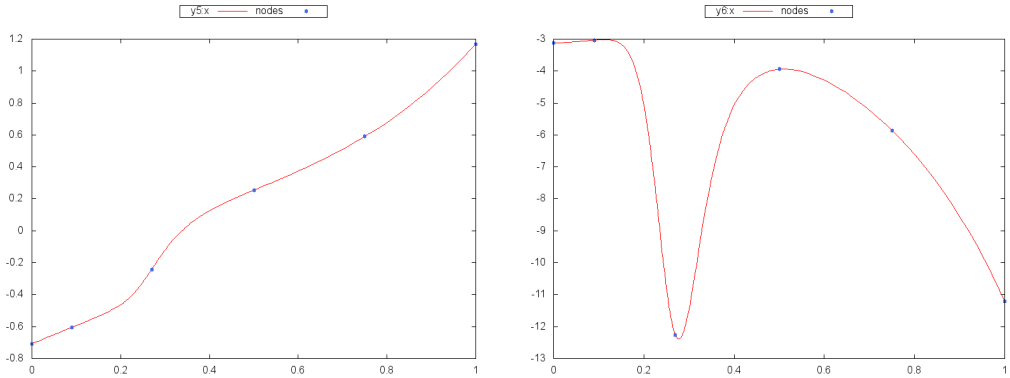
(a) v (b) γ

Abbildung 4.19: optimale Trajektorien der Zustandsvariablen v und γ



(a) ξ (b) λ_v

Abbildung 4.20: optimale Trajektorien der Variablen ξ und λ_v



(a) λ_γ (b) λ_ξ

Abbildung 4.21: optimale Trajektorien der Adjungierten λ_γ und λ_ξ

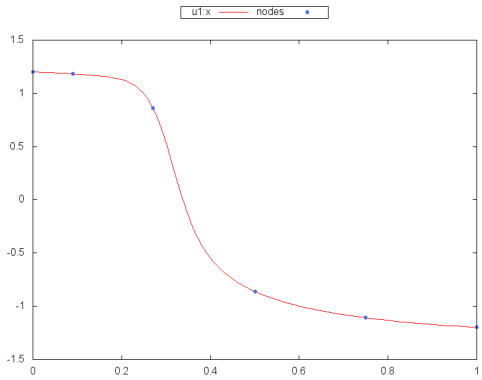


Abbildung 4.22: optimale Trajektorie der Steuerung u

Kapitel 5

Zusammenfassung

Es wurde ein Programm entwickelt, das zur Lösung von Optimalsteuerungsproblemen mit Hilfe der Mehrzielmethode eingesetzt werden kann. Dabei ist es gelungen, die eingangs genannten Anforderungen erfolgreich umzusetzen. Die implementierte grafische Oberfläche sowie die dadurch ermöglichte Eingabe, Speicherung und das Einlesen aller relevanten Daten über die grafische Oberfläche mit Hilfe des Tools *muParser* bieten unter Anderem die folgenden Vorteile:

- Eine grafische Oberfläche erleichtert im Allgemeinen die Bedienung, da sie intuitiver zu handhaben ist.
- Der Quellcode des Programms kann unverändert bleiben.
- Bei Änderungen der Eingabedaten sind keine Neukompilierungen nötig.
- Da die Problemdaten nicht im Quellcode gespeichert werden, entfällt ebenso der Nachteil, dass für jedes Problem ein eigener Quellcode vorhanden sein muss. Stattdessen ist es nun möglich, die Eingabedaten eines Problems in einer einfachen Textdatei zu speichern.
- Änderungen der Eingabedaten müssen nicht mehr vor der Lösung des Problems durch eine erneute Kompilierung gespeichert werden, sondern können zunächst getestet werden.
- Ein Benutzer benötigt keinerlei Programmierkenntnisse. Es muss lediglich dafür gesorgt werden, dass alle benötigten Pakete installiert werden und das Programm gegebenenfalls einmal kompiliert wird.

Wie eingangs der Arbeit erwähnt, ist der Funktionsumfang existierender Programme beschränkt bzw. ist es aufgrund nötiger Änderungen am Quellcode schwierig, diesen zu nutzen. Im vorliegenden Programm wurde eine Vielzahl von Optionen implementiert:

- Das Hinzufügen und/oder Entfernen von Mehrzielknoten wurde verbessert.

- Der Anwender kann beispielsweise zwischen mehreren Anfangswertproblemlösern wählen, sodass ein spezifischer für das Problem geeigneter Löser verwendet werden kann.
- Das der Mehrzielmethode zugrunde liegende Newtonverfahren sowie die Anfangswertproblemlöser können mittels verschiedener Parameter an die Anforderungen des zu lösenden Problems angepasst werden.
- Die grafische Darstellung einer Lösung mittels *gnuplot* wurde verbessert. Es können zum Beispiel verschiedenste Plots erstellt werden, die direkt in der grafischen Oberfläche des Programm angezeigt werden. Dabei ist es nicht nötig, *gnuplot* explizit zu benutzen, da die Grafiken vom Programm auf Knopfdruck automatisch erstellt werden.
- Dem Benutzer wird die Möglichkeit zur Verfügung gestellt, die Daten eines Problems mitsamt der Lösung und der Grafiken mit Hilfe von pdfLaTeX in ein PDF-Dokument zu exportieren. Analog zur Erstellung der Grafiken wird auch hier das Dokument automatisch vom Programm ohne die explizite Nutzung von pdfLaTeX erstellt.

Ebenso wurde die Anforderung, dass ausschließlich frei verfügbare Programme und Tools verwendet werden, erfüllt, sodass die Nutzung des Programms uneingeschränkt erfolgen kann.

Abschließend wurden einige Optimalsteuerungsprobleme, die sich in der Stabilität der Differenzialgleichungen und den auftretenden Nebenbedingungen deutlich unterscheiden, erfolgreich gelöst. Dabei konnte sowohl die Funktionsfähigkeit des Programms gezeigt als auch die sinnvolle Verwendung der diversen zur Verfügung gestellten Funktionen demonstriert werden.

Insgesamt ist zu sagen, dass erfolgreich ein Programm entwickelt wurde, das zur Lösung von Optimalsteuerungsproblemen geeignet ist und die eingangs gestellten Anforderungen bezüglich einer anwenderfreundlichen Bedienung und der Erweiterung des Funktionsumfangs erfüllt.

Abbildungsverzeichnis

1.1 Brachistochrone 3

3.1 Beispiel *muParser* 54
3.2 Hauptfenster 63
3.3 Eingabe der Bedingungen und Sprünge 64
3.4 Definition von gegebenen Konstanten 65
3.5 Fenster: Eingabe der Startschätzungen 66
3.6 Konstanten 68
3.7 Schaltpunkte 68
3.8 Definition der Steuerung in "control.cc" 68
3.9 Öffnen eines Projekts 69
3.10 Ausgabe der Lösung 74
3.11 Erstellung und Ausgabe der Grafiken 75
3.12 Definition eigener Grafiken 76

4.1 Brachistochrone-Problem: optimale Kurve 83
4.2 Min-Energy-Problem (unbeschränkt): optimale Trajektorien der Zustandsvariablen 88
4.3 Min-Energy-Problem (unbeschränkt): optimale Steuerung . . . 88
4.4 Min-Energy-Problem mit Berührpunkt: optimale Trajektorien der Zustandsvariablen 91
4.5 Min-Energy-Problem mit Berührpunkt: optimale Trajektorien der Adjungierten 92
4.6 Min-Energy-Problem mit Berührpunkt: optimale Steuerung . . 92
4.7 Min-Energy-Problem mit Berührpunkt: Zustandsvariable x & Ungleichungsbeschränkung 93
4.8 Min-Energy-Problem mit Randstück: optimale Trajektorien der Zustandsvariablen 95
4.9 Min-Energy-Problem mit Randstück: optimale Trajektorien der Adjungierten 96

4.10 Min-Energy-Problem mit Randstück: optimale Steuerung . . . 96
4.11 Rayleigh-Problem (unrestringiert): optimale Trajektorien der Zustandsvariablen & Adjungierten 99
4.12 Rayleigh-Problem (unrestringiert): optimale Steuerung & Zielfunktional 99
4.13 Rayleigh-Problem (unrestringiert): Nebenbedingung 100
4.14 Rayleigh-Problem (1 Schaltpunkt): Nebenbedingung 101
4.15 Rayleigh-Problem (2 Schaltpunkte): Nebenbedingung 102
4.16 Rayleigh-Problem (3 Schaltpunkte): optimale Trajektorien der Zustandsvariablen & Adjungierten 103
4.17 Rayleigh-Problem (3 Schaltpunkte): optimale Steuerung und Zielfunktional 103
4.18 Rayleigh-Problem (3 Schaltpunkte): Nebenbedingung 104
4.19 Re-Entry-Problem: optimale Trajektorien der Zustandsvariablen v und γ 109
4.20 Re-Entry-Problem: optimale Trajektorien der Variablen ξ und λ_v 109
4.21 Re-Entry-Problem: optimale Trajektorien der Adjungierten λ_γ und λ_ξ 110
4.22 Re-Entry-Problem: optimale Trajektorie der Steuerung u . . . 110

A.1 Brachistochrone: Ausgabe der Lösung 119
A.2 Min-Energy-Problem (unbeschränkt): Ausgabe der Lösung . . 121
A.3 Min-Energy-Problem mit Berührpunkt: Ausgabe der Lösung . 122
A.4 Min-Energy-Problem mit Randstück: Ausgabe der Lösung . . 123
A.5 Rayleigh-Problem (unrestringiert): Ausgabe der Lösung 125
A.6 Rayleigh-Problem (1 Schaltpunkt): Ausgabe der Lösung 127
A.7 Rayleigh-Problem (2 Schaltpunkte): Ausgabe der Lösung . . . 129
A.8 Rayleigh-Problem (3 Schaltpunkte): Ausgabe der Lösung . . . 131
A.9 Re-Entry-Problem: Ausgabe der Lösung 132

Tabellenverzeichnis

2.1 Butcher-Tableau Dormand-Prince 5(4) 34
2.2 Butcher-Tableau Dormand-Prince 8(7) 35

3.1 muParser: mathematische Funktionen 55
3.2 muParser: Operatoren 56

4.1 Brachistochrone-Problem: Startwerte 82
4.2 Brachistochrone-Problem: optimale Lösung 83
4.3 Brachistochrone: Vergleich der Optionen des Newtonverfahrens 84
4.4 Brachistochrone: Vergleich der Optionen der AWP-Löser . . . 84
4.5 Min-Energy-Problem (unbeschränkt): Startwerte 87
4.6 Min-Energy-Problem (unbeschränkt): optimale Lösung 88
4.7 Min-Energy-Problem mit Berührpunkt: optimale Lösung . . . 91
4.8 Min-Energy-Problem mit Randstück: optimale Lösung 95
4.9 Rayleigh-Problem (unrestringiert): Startwerte 98
4.10 Rayleigh-Problem (unrestringiert): optimale Lösung 99
4.11 Rayleigh-Problem (1 Schaltpunkt): optimale Lösung 101
4.12 Rayleigh-Problem (2 Schaltpunkte): optimale Lösung 102
4.13 Rayleigh-Problem (3 Schaltpunkte): optimale Lösung 103
4.14 Re-Entry-Problem: Variablen 105
4.15 Re-Entry-Problem: Konstanten 105
4.16 Re-Entry-Problem: Startlösung 108
4.17 Re-Entry-Problem: optimale Lösung 109

Literaturverzeichnis

[1] Bryson, Jr., A.E.; Ho, Yu-Chi: *Applied Optimal Control*, Hemisphere Publishing Corporation, Washington, 1975.

[2] Deuflhard, P.: *Newton Methods for Nonlinear Problems*, Springer Verlag, Berlin - Heidelberg - New York, 2004.

[3] Deuflhard, P.; Hohmann, A.: *Numerische Mathematik 1*, Walter de Gruyter Verlag, Berlin - New York, 4.Auflage , 2008.

[4] Dormand, J.R.; Prince, P.J.: *A family of embedded Runge-Kutta formulae*, Journal of Computational and Applied Mathematics Volume 6, Issue 1: S.19–26, 1980.

[5] Dormand, J.R.; Prince, P.J.: *High order embedded Runge-Kutta formulae*, Journal of Computational and Applied Mathematics Volume 7, Issue 1: S.67-75, 1981.

[6] Gerdts, M.: *Skript Optimale Steuerung, Uni Würzburg WiSe 2009/2010*, www.unibw.de/lrt1/gerdts/lehre/optimale_steuerung.pdf

[7] Hairer, E.; Wanner, G.: *Solving Ordinary Differential Equations II*, Springer Verlag, Berlin - Heidelberg - New York, 1991.

[8] Hanke-Bourgeois, M.: *Grundlagen der numerischen Mathematik und des Wissenschaftlichen Rechnens*, Vieweg+Teubner Verlag, Wiesbaden, 3.Auflage, 2009.

[9] Hartl, R.F.; Sethi, S.P.; Vickson, R.G.: *A survey of the maximum principles for optimal control problems with state constraints*, SIAM Review, Vol.37, No.2, S.181-218, Juni 1995.

[10] Hermann, M.: *Numerische Mathematik*, Oldenbourg Verlag, München - Wien, 2.Auflage, 2006.

[11] Oberle, H.J.; Grimm, W.: *BNDSCO - A Program for the Numerical Solution of Optimal Control Problems*, http://www.math.uni-hamburg.de/home/oberle/software/bndsco.pdf.

[12] Oberle, H.J.: *Skript Variationsrechnung und Optimale Steuerung*, www.math.uni-hamburg.de/home/oberle/skripte.html.

[13] Oberle, H.J.: *Skript Optimierung*, www.math.uni-hamburg.de/home/oberle/skripte.html.

[14] Locatelli, A.: *Optimal control: an introduction*, Birkhäuser Verlag, Basel - Boston - Berlin, 2001.

[15] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P.: *Numerical Recipes in C++: The Art of Scientific Computing*, Cambridge University Press, Second Edition, Cambridge 2002.

[16] Schwarz, H.R.; Köckler, N.: *Numerische Mathematik*, Vieweg+Teubner Verlag, Wiesbaden, 7. Auflage, 2009.

[17] Sethi, S.P.; Thompson, G.L.: *Optimal Control Theory - Applications to Management Science and Economics*, Kluwer Academic Publishers, Boston - Dordrecht - London, Second Edition, 2003.

[18] Stoer, J.; Bulirsch, R.: *Numerische Mathematik 2*, Springer Verlag, Berlin - Heidelberg - New York, 4.Auflage, 2000.

[19] Watkins, D.S.: *Fundamentals of Matrix Computations*, Wiley-Interscience, New York, Second Edition, 2002.

Anhang A

Lösungsausgaben des Programms bei den Beispielen

A.1 Das Brachistochrone-Problem

Listing A.1: Brachistochrone: Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   4
NUMBER OF CONSTANTS:                2
NUMBER OF NODES:                    5
NUMBER OF SWITCHING POINTS:         0
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          1

NODES and INITIAL VALUES:
    0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00
    2.500000e-01    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00
    5.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00
    7.500000e-01    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00
    1.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00

ODES:
    dy1 = c2*y3
    dy2 = c2*y4
    dy3 = -c2*u1*y4
    dy4 = c2*(u1*y3-p1)

CONTROL:
    u1:     c1

ESTIMATION OF THE CONSTANTS:
    c1:     4.000000e+00
    c2:     1.000000e+00

CONDITIONS:
    6 Boundary and switching conditions
        y1(s) = 0
        y2(s) = 0
        y3(s) = 0
        y4(s) = 0
        y1(e)-2.0 = 0
        y2(e)+1.0 = 0
```

```
GIVEN CONSTANTS:
    p1:     9.81


1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           8.860181e-01
    Maximum change of y:    1.000000e+00


2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           6.071949e-01
    Maximum change of y:    5.794979e-01


3. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           5.882033e-01
    Maximum change of y:    3.043752e-02


4. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           5.160708e-02
    Maximum change of y:    7.139203e-01


5. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           7.032171e-03
    Maximum change of y:    1.606610e-02


6. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.827260e-04
    Maximum change of y:    1.043637e-02


7. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.663694e-04
    Maximum change of y:    1.423491e-03


8. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.647968e-05
    Maximum change of y:    6.831540e-05


9. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.404445e-06
    Maximum change of y:    3.482807e-05


10. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           5.854810e-07
```

```
    Maximum change of y:    5.156115e-06


FOLLOWING SOLUTION FOUND AFTER 10 ITERATIONS (Time: 1.14 sec.):

        x                y1               y2               y3
                y4
    0.000000e+00     2.379314e-15     1.761941e-15     -3.903056e-15
            1.684672e-15
    2.500000e-01     5.596585e-02     -1.865076e-01     8.122727e-01
            -1.731904e+00
    5.000000e-01     3.987368e-01     -6.115174e-01     2.663264e+00
            -2.214724e+00
    7.500000e-01     1.108268e+00     -9.685034e-01     4.217999e+00
            -1.100238e+00
    1.000000e+00     2.000000e+00     -1.000000e+00     4.355173e+00
            8.077619e-01

CONSTANTS
    c1:     4.355171e+00
    c2:     8.055638e-01
```

A.2 Das Min-Energy-Problem

Listing A.2: Min-Energy-Problem (unbeschränkt): Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   3
NUMBER OF CONSTANTS:                1
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         0
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          0

NODES and INITIAL VALUES:
    0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00
    2.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
    4.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
    6.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
    8.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
    1.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00

ODES:
    dy1 = y2
    dy2 = u1
    dy3 = -c1

CONTROL:
    u1:     -y3

ESTIMATION OF THE CONSTANTS:
    c1:     0.000000e+00

CONDITIONS:
    4 Boundary and switching conditions
        y1(s) = 0
        y2(s)-1.0 = 0
        y1(e) = 0
        y2(e)+1.0 = 0


1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           4.011439e-10
    Maximum change of y:    1.000000e+00
```

```
2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.220446e-16
    Maximum change of y:    9.188823e-10


FOLLOWING SOLUTION FOUND AFTER 2 ITERATIONS (Time: 0.16 sec.):

        x               y1              y2              y3
    0.000000e+00     1.499601e-19     1.000000e+00     2.000000e+00
    2.000000e-01     1.600000e-01     6.000000e-01     2.000000e+00
    4.000000e-01     2.400000e-01     2.000000e-01     2.000000e+00
    6.000000e-01     2.400000e-01     -2.000000e-01     2.000000e+00
    8.000000e-01     1.600000e-01     -6.000000e-01     2.000000e+00
    1.000000e+00     2.569006e-19     -1.000000e+00     2.000000e+00

CONSTANTS
    c1:     2.179050e-15
```

Listing A.3: Min-Energy-Problem mit Berührpunkt: Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   4
NUMBER OF CONSTANTS:                1
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         1
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          0

NODES and INITIAL VALUES:
    0.000000e+00    0.000000e+00    1.000000e+00    0.000000e+00
        2.000000e+00
    2.000000e-01    1.600000e-01    6.000000e-01    0.000000e+00
        2.000000e+00
    4.000000e-01    2.400000e-01    2.000000e-01    0.000000e+00
        2.000000e+00
    6.000000e-01    2.400000e-01    -2.000000e-01   0.000000e+00
        2.000000e+00
    8.000000e-01    1.600000e-01    -6.000000e-01   0.000000e+00
        2.000000e+00
    1.000000e+00    0.000000e+00    -1.000000e+00   0.000000e+00
        2.000000e+00

ODES:
    dy1 = y2
    dy2 = u1
    dy3 = 0.0
    dy4 = -y3

CONTROL:
    u1:     -y4     -y4

ESTIMATION OF THE SWITCHING POINTS:
    sp1:    5.000000e-01

ESTIMATION OF THE CONSTANTS:
    c1:     1.000000e+00

CONDITIONS:
    6 Boundary and switching conditions
        y1(s) = 0
        y2(s)-1.0 = 0
        y1(e) = 0
        y2(e)+1.0 = 0
        y1(sp1)-0.2 = 0
        y2(sp1) = 0
    1 Jump condition
        y3+=c1  at switching point 1


1. ITERATION:
```

```
    BREAKUP CONDITIONS:
    Maximum of F:           7.283091e-08
    Maximum change of y:    1.104167e+00


2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           7.979572e-09
    Maximum change of y:    2.467811e-07


FOLLOWING SOLUTION FOUND AFTER 2 ITERATIONS (Time: 0.21 sec.):

        x                y1               y2               y3
                y4
    0.000000e+00     -1.113968e-13     1.000000e+00     4.800000e+00
            3.200000e+00
    2.000000e-01     1.424000e-01     4.560000e-01     4.800000e+00
            2.240000e+00
    4.000000e-01     1.952000e-01     1.040000e-01     4.800000e+00
            1.280000e+00
    6.000000e-01     1.952000e-01     -1.040000e-01     -4.800000e+00
            1.280000e+00
    8.000000e-01     1.424000e-01     -4.560000e-01     -4.800000e+00
            2.240000e+00
    1.000000e+00     9.065329e-14     -1.000000e+00     -4.800000e+00
            3.200000e+00

SWITCHING POINTS
    sp1:    5.000000e-01

CONSTANTS
    c1:     -9.600000e+00
```

Listing A.4: Min-Energy-Problem mit Randstück: Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   4
NUMBER OF CONSTANTS:                2
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         2
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          0

NODES and INITIAL VALUES:
    0.000000e+00    5.225390e-16    1.000000e+00    4.800000e+00
        3.200000e+00
    2.000000e-01    1.424000e-01    4.560000e-01    4.800000e+00
        2.240000e+00
    4.000000e-01    1.952000e-01    1.040000e-01    4.800000e+00
        1.280000e+00
    6.000000e-01    1.952000e-01    -1.040000e-01   -4.800000e+00
        1.280000e+00
    8.000000e-01    1.424000e-01    -4.560000e-01   -4.800000e+00
        2.240000e+00
    1.000000e+00    -5.066440e-16   -1.000000e+00   -4.800000e+00
        3.200000e+00

ODES:
    dy1 = y2
    dy2 = u1
    dy3 = 0.0
    dy4 = -y3

CONTROL:
    u1:     -y4     0.0     -y4

ESTIMATION OF THE SWITCHING POINTS:
    sp1:    2.500000e-01
    sp2:    7.500000e-01
```

```
ESTIMATION OF THE CONSTANTS:
    c1:     1.000000e+00
    c2:     1.000000e+00

CONDITIONS:
    8 Boundary and switching conditions
        y1(s) = 0
        y2(s)-1.0 = 0
        y1(e) = 0
        y2(e)+1.0 = 0
        y4(sp1) = 0
        y1(sp1)-0.1 = 0
        y2(sp1) = 0
        y4(sp2) = 0
    2 Jump conditions
        y3+=c1  at switching point 1
        y4+=c2  at switching point 1


1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           8.610930e-01
    Maximum change of y:    1.294807e+00


2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.662515e-01
    Maximum change of y:    3.792567e-01


3. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.201837e-02
    Maximum change of y:    3.268217e-01


4. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.298684e-02
    Maximum change of y:    2.692368e-02


5. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           5.392741e-04
    Maximum change of y:    4.002246e-02


6. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           8.012606e-06
    Maximum change of y:    3.510788e-04


7. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.113947e-08
    Maximum change of y:    4.914515e-06


FOLLOWING SOLUTION FOUND AFTER 7 ITERATIONS (Time: 0.45 sec.):
```

```
        x                y1               y2               y3
                y4
    0.000000e+00     -2.372332e-16     1.000000e+00     2.222222e+01
            6.666667e+00
    2.000000e-01     9.629630e-02     1.111111e-01     2.222222e+01
            2.222222e+00
    3.000000e-01     1.000000e-01     -2.791653e-09     2.222222e+01
            1.113786e-08
    4.000000e-01     1.000000e-01     1.930595e-17     -2.222222e+01
            -6.666667e+00
    6.000000e-01     1.000000e-01     6.890362e-16     -2.222222e+01
            -2.222222e+00
    7.000000e-01     1.000000e-01     6.890362e-16     -2.222222e+01
            1.113947e-08
    8.000000e-01     9.629630e-02     -1.111111e-01     -2.222222e+01
            2.222222e+00
    1.000000e+00     -4.274289e-17     -1.000000e+00     -2.222222e+01
             6.666667e+00

SWITCHING POINTS
    sp1:    3.000000e-01
    sp2:    7.000000e-01

CONSTANTS
    c1:     -4.444444e+01
    c2:     -8.888889e+00
```

A.3 Das Rayleigh-Problem

Listing A.5: Rayleigh-Problem (unrestringiert): Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   5
NUMBER OF CONSTANTS:                0
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         0
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          0

NODES and INITIAL VALUES:
    0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00    0.000000e+00
    5.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00    0.000000e+00
    1.500000e+00    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00    0.000000e+00
    2.500000e+00    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00    0.000000e+00
    3.500000e+00    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00    0.000000e+00
    4.500000e+00    0.000000e+00    0.000000e+00    0.000000e+00
        0.000000e+00    0.000000e+00

ODES:
    dy1 = y2
    dy2 = -y1+1.4*y2-0.14*y2^3+4*u1
    dy3 = -2*y1+y4
    dy4 = -y3-1.4*y4+0.42*y2^2*y4
    dy5 = u1^2+y1^2

CONTROL:
    u1:     -2*y4

CONDITIONS:
    5 Boundary and switching conditions
        y1(s)+5 = 0
        y2(s)+5 = 0
        y3(e) = 0
```

```
        y4(e) = 0
        y5(s) = 0


1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           6.112671e+00
    Maximum change of y:    1.000000e+00


2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           6.188844e+00
    Maximum change of y:    1.000000e+00


3. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           4.852229e+00
    Maximum change of y:    3.320910e-01


4. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.836158e+00
    Maximum change of y:    3.768481e-01


5. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           7.201262e-01
    Maximum change of y:    2.830983e-01


6. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.015596e-01
    Maximum change of y:    1.335034e-01


7. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           5.072946e-02
    Maximum change of y:    8.238792e-02


8. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.028582e-03
    Maximum change of y:    2.320957e-02


9. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.237224e-06
    Maximum change of y:    7.265320e-04


10. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           8.655365e-11
```

```
    Maximum change of y:    1.413287e-06


FOLLOWING SOLUTION FOUND AFTER 10 ITERATIONS (Time: 27.87 sec.):

        x                y1               y2               y3
                y4               y5
    0.000000e+00     -5.000000e+00     -5.000000e+00     -9.001243e+00
             -2.672616e+00     2.037179e-16
    5.000000e-01     -4.564134e+00     4.232434e+00     -4.674163e+00
            -5.255322e-01     2.064724e+01
    1.500000e+00     -5.870499e-01     2.714466e+00     2.336765e-01
            7.292466e-01     2.849424e+01
    2.500000e+00     3.274390e-01     -1.756616e-01     4.540488e-01
            2.689840e-02     2.967675e+01
    3.500000e+00     6.671313e-02     -1.932589e-01     -1.240326e-03
            -6.751068e-02     2.974243e+01
    4.500000e+00     -9.571141e-02     -2.043943e-01     1.088751e-15
            -3.386098e-16     2.974893e+01
```

Listing A.6: Rayleigh-Problem (1 Schaltpunkt): Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   5
NUMBER OF CONSTANTS:                0
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         1
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          0

NODES and INITIAL VALUES:
    0.000000e+00    -5.000000e+00   -5.000000e+00   -9.001240e+00
        -2.672620e+00   2.037640e-16
    5.000000e-01    -4.564130e+00   4.232430e+00    -4.674160e+00
        -5.255320e-01   2.064720e+01
    1.500000e+00    -5.870500e-01   2.714470e+00    2.336760e-01
        7.292470e-01    2.849420e+01
    2.500000e+00    3.274390e-01    -1.756620e-01   4.540490e-01
        2.689840e-02    2.967680e+01
    3.500000e+00    6.671310e-02    -1.932590e-01   -1.240330e-03
        -6.751070e-02   2.974240e+01
    4.500000e+00    -9.571140e-02   -2.043940e-01   1.088650e-15
        -3.386000e-16   2.974890e+01

ODES:
    dy1 = y2
    dy2 = -y1+1.4*y2-0.14*y2^3+4*u1
    dy3 = -2*y1+y4
    dy4 = -y3-1.4*y4+0.42*y2^2*y4
    dy5 = u1^2+y1^2

CONTROL:
    u1:     -1/6*y1     -2*y4

ESTIMATION OF THE SWITCHING POINTS:
    sp1:    7.500000e-01

CONDITIONS:
    6 Boundary and switching conditions
        y1(s)+5 = 0
        y2(s)+5 = 0
        y3(e) = 0
        y4(e) = 0
        y5(s) = 0
        2*y4(sp1)-1/6*y1(sp1) = 0


1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.540181e+00
    Maximum change of y:    1.027754e+00
```

```
2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.512012e+00
    Maximum change of y:    5.463970e-01


3. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.773493e+00
    Maximum change of y:    3.980284e-01


4. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.368596e+00
    Maximum change of y:    6.466288e-01


5. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.098823e+00
    Maximum change of y:    2.352559e-01


6. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           4.517412e+00
    Maximum change of y:    9.044264e-01


7. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.070050e+00
    Maximum change of y:    7.281596e-01


8. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.018258e+00
    Maximum change of y:    2.054386e-01


9. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.526673e+00
    Maximum change of y:    1.183936e+00


10. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.835351e+00
    Maximum change of y:    1.174500e-01


11. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.541031e+00
    Maximum change of y:    1.114824e-01
```

```
12. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.249972e+00
    Maximum change of y:    6.349724e-01


13. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           5.935329e-01
    Maximum change of y:    2.710632e-02


14. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           7.502797e-02
    Maximum change of y:    1.339977e-01


15. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           5.837873e-03
    Maximum change of y:    6.378351e-02


16. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           6.408015e-06
    Maximum change of y:    1.567389e-03


17. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           8.033041e-11
    Maximum change of y:    5.725704e-06


FOLLOWING SOLUTION FOUND AFTER 17 ITERATIONS (Time: 41.87 sec.):

        x               y1              y2              y3
              y4              y5
    0.000000e+00     -5.000000e+00     -5.000000e+00     -1.280895e+01
             -4.545008e+00     4.642967e-16
    5.000000e-01     -6.164486e+00     -5.792882e-02     -1.020016e+01
             -3.945716e+00     1.718369e+01
    1.500000e+00     -2.656695e+00     4.282137e+00     -1.684644e+00
            -1.078939e-01     4.161477e+01
    2.500000e+00     1.988929e-01     9.675205e-01     6.579628e-01
            5.445724e-01     4.443767e+01
    3.500000e+00     2.550501e-01     -3.291321e-01     1.339649e-01
            -6.462117e-02     4.474367e+01
    4.500000e+00     -2.240287e-01     -7.477577e-01     -6.608549e-16
             1.686377e-16     4.477173e+01

SWITCHING POINTS
    sp1:    1.281497e+00
```

Listing A.7: Rayleigh-Problem (2 Schaltpunkte): Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   5
NUMBER OF CONSTANTS:                0
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         2
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          0
```

```
NODES and INITIAL VALUES:
    0.000000e+00    -5.000000e+00   -5.000000e+00   -1.280900e+01
        -4.545010e+00   4.365230e-16
    5.000000e-01    -6.164490e+00   -5.792880e-02   -1.020020e+01
        -3.945720e+00   1.718370e+01
    1.500000e+00    -2.656700e+00   4.282140e+00    -1.684640e+00
        -1.078940e-01   4.161480e+01
    2.500000e+00    1.988930e-01    9.675210e-01    6.579630e-01
        5.445720e-01    4.443770e+01
    3.500000e+00    2.550500e-01    -3.291320e-01   1.339650e-01
        -6.462120e-02   4.474370e+01
    4.500000e+00    -2.240290e-01   -7.477580e-01   -6.430840e-16
        1.662310e-16    4.477170e+01

ODES:
    dy1 = y2
    dy2 = -y1+1.4*y2-0.14*y2^3+4*u1
    dy3 = -2*y1+y4
    dy4 = -y3-1.4*y4+0.42*y2^2*y4
    dy5 = u1^2+y1^2

CONTROL:
    u1:     -1/6*y1     -2*y4       -1/6*y1

ESTIMATION OF THE SWITCHING POINTS:
    sp1:    1.281500e+00
    sp2:    3.000000e+00

CONDITIONS:
    7 Boundary and switching conditions
        y1(s)+5 = 0
        y2(s)+5 = 0
        y3(e) = 0
        y4(e) = 0
        y5(s) = 0
        2*y4(sp1)-1/6*y1(sp1) = 0
        2*y4(sp2)-1/6*y1(sp2) = 0


1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           6.482712e-03
    Maximum change of y:    4.037554e-01


2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.295598e-05
    Maximum change of y:    3.385337e-03


3. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           8.944221e-11
    Maximum change of y:    6.008170e-06


FOLLOWING SOLUTION FOUND AFTER 3 ITERATIONS (Time: 6.19 sec.):

        x               y1              y2              y3
                y4              y5
    0.000000e+00    -5.000000e+00   -5.000000e+00   -1.279332e+01
            -4.534162e+00   2.336755e-16
    5.000000e-01    -6.164486e+00   -5.792882e-02   -1.017521e+01
            -3.931807e+00   1.718369e+01
    1.500000e+00    -2.657654e+00   4.273691e+00    -1.652836e+00
            -9.580342e-02   4.161231e+01
```

```
    2.500000e+00     1.505641e-01     9.277770e-01     7.435558e-01
            5.161694e-01     4.449263e+01
    3.500000e+00     3.350362e-01     -2.126862e-01     1.824378e-01
            -1.203568e-01     4.475508e+01
    4.500000e+00     -3.938694e-01     -1.249874e+00     1.741128e-15
            -3.668817e-16     4.480593e+01

SWITCHING POINTS
    sp1:    1.269165e+00
    sp2:    2.984039e+00
```

Listing A.8: Rayleigh-Problem (3 Schaltpunkte): Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   5
NUMBER OF CONSTANTS:                0
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         3
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          0

NODES and INITIAL VALUES:
    0.000000e+00    -5.000000e+00   -5.000000e+00   -1.279330e+01
        -4.534160e+00   2.334850e-16
    5.000000e-01    -6.164490e+00   -5.792880e-02   -1.017520e+01
        -3.931810e+00   1.718370e+01
    1.500000e+00    -2.657650e+00   4.273690e+00    -1.652840e+00
        -9.580340e-02   4.161230e+01
    2.500000e+00    1.505640e-01    9.277770e-01    7.435560e-01
        5.161690e-01    4.449260e+01
    3.500000e+00    3.350360e-01    -2.126860e-01   1.824380e-01
        -1.203570e-01   4.475510e+01
    4.500000e+00    -3.938690e-01   -1.249870e+00   1.796810e-15
        -3.886890e-16   4.480590e+01

ODES:
    dy1 = y2
    dy2 = -y1+1.4*y2-0.14*y2^3+4*u1
    dy3 = -2*y1+y4
    dy4 = -y3-1.4*y4+0.42*y2^2*y4
    dy5 = u1^2+y1^2

CONTROL:
    u1:     -1/6*y1     -2*y4       -1/6*y1     -2*y4

ESTIMATION OF THE SWITCHING POINTS:
    sp1:    1.269170e+00
    sp2:    2.984040e+00
    sp3:    4.200000e+00

CONDITIONS:
    8 Boundary and switching conditions
        y1(s)+5 = 0
        y2(s)+5 = 0
        y3(e) = 0
        y4(e) = 0
        y5(s) = 0
        2*y4(sp1)-1/6*y1(sp1) = 0
        2*y4(sp2)-1/6*y1(sp2) = 0
        2*y4(sp3)-1/6*y1(sp3) = 0


1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.032304e-02
    Maximum change of y:    3.079786e-02


2. ITERATION:

    BREAKUP CONDITIONS:
```

```
    Maximum of F:           2.274098e-05
    Maximum change of y:    7.212099e-03


3. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           6.031700e-10
    Maximum change of y:    1.687421e-05


FOLLOWING SOLUTION FOUND AFTER 3 ITERATIONS (Time: 6.12 sec.):

        x               y1              y2              y3
              y4              y5
    0.000000e+00     -5.000000e+00     -5.000000e+00     -1.279332e+01
             -4.534159e+00     -2.499370e-17
    5.000000e-01     -6.164486e+00     -5.792882e-02     -1.017520e+01
             -3.931803e+00     1.718369e+01
    1.500000e+00     -2.657654e+00     4.273686e+00     -1.652826e+00
            -9.579512e-02     4.161231e+01
    2.500000e+00     1.505133e-01     9.277072e-01     7.436186e-01
            5.161506e-01     4.449270e+01
    3.500000e+00     3.350465e-01     -2.124942e-01     1.825451e-01
            -1.204185e-01     4.475510e+01
    4.500000e+00     -3.957080e-01     -1.280331e+00     -7.304258e-17
             -1.439924e-16     4.480559e+01

SWITCHING POINTS
    sp1:    1.269160e+00
    sp2:    2.983980e+00
    sp3:    4.299258e+00
```

A.4 Das Re-Entry-Problem

Listing A.9: Re-Entry-Problem: Ausgabe der Lösung

```
NUMBER OF DIFFERENTIAL EQUATIONS:   6
NUMBER OF CONSTANTS:                1
NUMBER OF NODES:                    6
NUMBER OF SWITCHING POINTS:         0
NUMBER OF CONTROLS:                 1
NUMBER OF GIVEN CONSTANTS:          5

NODES and INITIAL VALUES:
    0.000000e+00    3.600000e-01    -1.400000e-01   1.913876e-02
        -5.000000e-01   -7.000000e-01   -2.300000e+00
    9.000000e-02    3.600000e-01    -1.220000e-01   1.435407e-02
        -4.500000e-01   -6.000000e-01   -2.200000e+00
    2.700000e-01    3.200000e-01    0.000000e+00    8.612440e-03
        -4.000000e-01   -2.000000e-01   -1.350000e+01
    5.000000e-01    2.850000e-01    1.700000e-02    1.100478e-02
        -6.000000e-01   3.000000e-01    -4.500000e+00
    7.500000e-01    2.800000e-01    1.000000e-02    1.148325e-02
        -9.000000e-01   7.000000e-01    -7.000000e+00
    1.000000e+00    2.700000e-01    0.000000e+00    1.196172e-02
        -1.400000e+00   1.400000e+00    -1.300000e+01

ODES:
    dy1 = c1*(-0.5*p1*p2*exp(-p4*p3*y3)*y1^2*(1.174-0.9*cos(u1))-p5*
        sin(y2)/(1+y3)^2)
    dy2 = c1*(0.5*p1*p2*exp(-p4*p3*y3)*y1*0.6*sin(u1)+y1*cos(y2)/(p3
        *(1+y3))-p5*cos(y2)/(y1*(1+y3)^2))
    dy3 = c1*(y1*sin(y2)/p3)
    dy4 = c1*(-30*y1^3*sqrt(p2*exp(-p4*p3*y3))+y4*(p1*p2*exp(-p4*p3*y3
        )*y1*(1.174-0.9*cos(u1)))-y5*(0.5*p1*p2*exp(-p4*p3*y3)*0.6*sin
```

```
        (u1)+cos(y2)/(p3*(1+y3))+p5*cos(y2)/(y1^2*(1+y3)^2))-y6*sin(y2
        )/p3)
    dy5 = c1*(y4*p5*cos(y2)/(1+y3)^2+y5*(y1*sin(y2)/(p3*(1+y3))-p5*sin
        (y2)/(y1*(1+y3)^2))-y6*y1*cos(y2)/p3)
    dy6 = c1*(5*y1^3*p3*p4*sqrt(p2*exp(-p4*p3*y3))-y4*(0.5*p1*p2*p3*p4
        *exp(-p4*p3*y3)*y1^2*(1.174-0.9*cos(u1))+2*p5*sin(y2)/(1+y3)
        ^3)+y5*(0.5*p1*p2*p3*p4*exp(-p4*p3*y3)*y1*0.6*sin(u1)+y1*cos(
        y2)/(p3*(1+y3)^2)-2*p5*cos(y2)/(y1*(1+y3)^3)))

CONTROL:
    u1:     atan(2*y5/(3*y1*y4))

ESTIMATION OF THE CONSTANTS:
    c1:     2.250000e+02

CONDITIONS:

1. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.581781e-01
    Maximum change of y:    6.524690e-01


2. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.573859e-02
    Maximum change of y:    2.932745e-01


3. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.453770e-02
    Maximum change of y:    5.737951e-02


4. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           6.883923e-03
    Maximum change of y:    9.907147e-03


5. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           9.109337e-04
    Maximum change of y:    1.132365e-02


6. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           2.245200e-04
    Maximum change of y:    1.513522e-03


7. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           1.168943e-04
    Maximum change of y:    1.211553e-03


8. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           3.736472e-06
    Maximum change of y:    1.343231e-03
```

```
9. ITERATION:

    BREAKUP CONDITIONS:
    Maximum of F:           7.115659e-07
    Maximum change of y:    3.957904e-06


FOLLOWING SOLUTION FOUND AFTER 9 ITERATIONS (Time: 63.14 sec.):

        x                y1               y2               y3
                y4               y5               y6
    0.000000e+00     3.600000e-01     -1.413720e-01     1.913876e-02
            -5.081402e-01     -7.086216e-01     -3.119301e+00
    9.000000e-02     3.607364e-01     -1.246096e-01     1.453955e-02
            -4.555757e-01     -6.041886e-01     -3.030884e+00
    2.700000e-01     3.215374e-01     -8.960894e-03     8.433512e-03
            -4.284077e-01     -2.414743e-01     -1.226857e+01
    5.000000e-01     2.865279e-01     2.200586e-02     1.066110e-02
            -5.076426e-01     2.546496e-01     -3.942957e+00
    7.500000e-01     2.763511e-01     7.717901e-03     1.168228e-02
            -7.105604e-01     5.897758e-01     -5.858717e+00
    1.000000e+00     2.700000e-01     2.020641e-14     1.196172e-02
            -1.130092e+00     1.170080e+00     -1.121106e+01

CONSTANTS
    c1:     2.234013e+02

    7 Boundary and switching conditions
        y1(s)-0.36 = 0
        y2(s)+8.1*3.1416/180 = 0
        y3(s)-4/p3 = 0
        y1(e)-0.27 = 0
        y2(e) = 0
        y3(e)-2.5/p3 = 0
        10*y1(e)^3*sqrt(p2*exp(-p4*p3*y3(e)))+y4(e)*(-0.5*p1*p2*exp(-
            p4*p3*y3(e))*y1(e)^2*(1.174-0.9*cos(u1(e)))-p5*sin(y2(e))
            /(1+y3(e))^2)+y5(e)*(0.5*p1*p2*exp(-p4*p3*y3(e))*y1(e)
            *0.6*sin(u1(e))+y1(e)*cos(y2(e))/(p3*(1+y3(e)))-p5*cos(y2(
            e))/(y1(e)*(1+y3(e))^2))+y6(e)*(y1(e)*sin(y2(e))/p3) = 0

GIVEN CONSTANTS:
    p1:     53200
    p2:     0.002704
    p3:     209
    p4:     4.26
    p5:     0.00032172
```