Direkte Optimierungsmethoden zur numerischen Berechnung optimaler Steuerungen

Diplomarbeit
zur Erlangung des Mathematik-Diploms
am
Institut für numerische und instrumentelle Mathematik
der
Westfälischen Wilhelms-Universität Münster
Prof. Dr. H. Maurer

vorgelegt von
Christof Büskens

31. März 1993

Inhaltsverzeichnis

1 Einleitung **1**

2 Zur Theorie optimaler Steuerprozesse **4**

2.1 Grundlagen 4
2.2 Das unbeschränkte Steuerproblem 8
2.3 Rückführung optimaler Steuerprozesse auf ein Ausgangsproblem 11
2.4 Bang-bang und singuläre Steuerungen –Probleme mit linear eingehender Steuerung 13
2.5 Reguläre Hamiltonprobleme, –Probleme mit nichtlinear auftretender Steuerung 15
2.6 Das zustandsbeschränkte Steuerproblem 19
2.7 Weitere Beschränkungen 26

3 Numerische Lösungsverfahren für Randwertprobleme **27**

3.1 Das Randwertproblem 28
3.2 Das Einfach-Schießverfahren 28
3.3 Die Mehrzielmethode 31
3.4 Das Kollokationsverfahren 33

4 Numerische Lösung optimaler Steuerprozesse durch direkte Verfahren **37**

4.1 Das Standardproblem der nichtlinearen Optimierung 38
4.2 Direkte Verfahren durch Diskretisierung der Steuer- und Zustandsfunktionen 39
4.2.1 Diskretisierungsansätze nach Betts 39
4.2.2 Spezielle Eigenschaften der beschriebenen Diskretisierungen 42
4.2.3 Lösung eines optimalen Steuerprozesses durch direkte Kollokation 46
4.2.4 Spezielle Eigenschaften der direkten Kollokation 51
4.2.5 Bestimmung einer Anfangsschätzung für die adjungierten Variablen 53

4.2.6 Nachteile und Schwierigkeiten bei der praktischen Realisierung direkter Verfahren durch Diskretisierung der Steuer- und Zustandsfunktionen 58
4.3 Direkte Verfahren durch Diskretisierung der Steuerfunktion kombiniert mit dem Lösen von Anfangswertproblemen 60
4.3.1 Die direkte Mehrzielmethode nach Bock 60
4.3.2 Spezielle Eigenschaften der direkten Mehrzielmethode . 61
4.3.3 Lösung eines optimalen Steuerprozesses durch Problemsharing 63
4.3.4 Spezielle Eigenschaften des direkten Verfahrens durch Problemsharing 72
4.4 Allgemeine Modifikationstechniken der beschriebenen Diskretisierungsmethoden 74
4.4.1 Variationsmöglichkeiten der Zustands- und Steuerbeschränkungen 75
4.4.2 Optimale Diskretisierungsschrittweite 78
4.4.3 Eine 'sichere' Methode zur genaueren Bestimmung von speziellen Schaltpunkten 80
4.4.4 Eine spezielle 'Schlauchtechnik' 82

5 Anwendungsbeispiele und deren Auswertung 85
5.1 Grundlagen/Voraussetzungen 85
5.2 Anwendungsbeispiel 1: Ein zeitoptimales Problem mit Steuer- und Zustandsbeschränkungen und linear eingehender Steuerung 87
5.2.1 Zeitoptimale Steuerung einer Verladebrücke 87
5.2.2 Numerische Auswertungen 89
5.3 Anwendungsbeispiel 2: Ein Problem mit regulärer Hamilton-Funktion und kompaktem Steuerbereich 93
5.3.1 Optimale Temperaturverteilung in einem chemischen Reaktor 93
5.3.2 Numerische Auswertungen 94
5.4 Anwendungsbeispiel 3: Ein Problem mit regulärer Hamilton-Funktion und einer Beschränkung im Zustandsraum 96
5.4.1 Ein Spline-Problem 96
5.4.2 Numerische Auswertungen 98
5.5 Anwendungsbeispiel 4: Ein Problem mit regulärer Hamilton-Funktion, Zustandsbeschränkungen und 'Pseudominima' . . . 104
5.5.1 Knickstab 104
5.5.2 Numerische Auswertungen 106
5.5.3 Das Auftreten von 'Pseudominima' 111
5.6 Zusammenfassung der numerischen Ergebnisse 111

6 Schlußbemerkung **116**

A Beschreibung der Subroutine BDSCO **118**
A.1 Problemstellung 118
A.2 Die Parameterliste der Routine BDSCO 119
A.3 Vom Anwender bereitzustellende Unterprogramme 123
A.3.1 Die Subroutine `F` 123
A.3.2 Die Subroutine `R` 124
A.3.3 Die Subroutine `METHOD` 125
A.4 Anmerkungen 126

B Beschreibung der Subroutine NUDOCCCS **127**
B.1 Problemstellung 127
B.2 Die NUDOCCCS-Parameterliste 129
B.3 Vom Anwender bereitzustellende Unterprogramme 140
B.3.1 Die Subroutine `MINFKT` 140
B.3.2 Die Subroutine `DGLSYS` 141
B.3.3 Die Subroutine `ANFANGSW` 142
B.3.4 Die Subroutine `RANDBED` 142
B.3.5 Die Subroutine `NEBENBED` 143
B.4 Anmerkungen 144

C Beschreibung der Subroutine KOLLOKAT **145**
C.1 Problemstellung 145
C.2 Die KOLLOKAT-Parameterliste 145
C.3 Vom Anwender bereitzustellende Unterprogramme 147
C.4 Anmerkungen 147

D Beschreibung der Subroutine DIROPTIM **148**
D.1 Problemstellung 148
D.2 Die DIROPTIM-Parameterliste 148
D.3 Vom Anwender bereitzustellende Unterprogramme 148
D.4 Anmerkungen 149

E Beschreibung der Subroutine DIRMESCH **150**
E.1 Problemstellung 150
E.2 Die DIRMESCH-Parameterliste 150
E.3 Vom Anwender bereitzustellende Unterprogramme 151
E.4 Anmerkungen 151

F Beispiele einiger FORTRAN-Programme **152**
F.1 Anwendungsbeispiel 1 152
F.2 Anwendungsbeispiel 2 167
F.3 Anwendungsbeispiel 3 173
F.4 Anwendungsbeispiel 4 190

Literaturverzeichnis **211**

Tabellenverzeichnis

2.1 Rückführung von Steuerprozessen 12
2.2 Abhängigkeit von der Ordnung p der Zustandsbeschränkung . 25

4.1 Vergleich der Diskretisierungs-Verfahren 46
4.2 Maximale Fehlerordnung von expliziten Runge-Kutta-Verfahren 69
4.3 Realisierte Kombinationen der Routine NUDOCCCS 71

5.1 Lösungsdaten der unbeschränkten Verladebrücke 90
5.2 Lösungsdaten der durch $x_2 \leq 0.4$ beschränkten Verladebrücke 91
5.3 Lösungsdaten des chemischen Reaktors für $u_{max} = 1$ 95
5.4 Rechenzeiten der direkten Verfahren in Sekunden 95
5.5 Lösungsstrukturen des Spline-Problems 97
5.6 Lösungsdaten des durch $x_1 \leq 0.2$ beschränkten Spline-Problems 101
5.7 Lösungsdaten des durch $x_1 \leq 0.1$ beschränkten Spline-Problems 102
5.8 Lösungsstrukturen des Knickstab-Problems 106
5.9 Lösungsdaten des Knickstab-Problems für $\alpha = 9.90027$ 107
5.10 Lösungsdaten des Knickstab-Problems für $\alpha = 39.3291$ 109
5.11 Lösungsdaten des Knickstab-Problems für $\alpha = 105.190$ 110
5.12 Zusammenfassung der numerischen Ergebnisse 113
5.13 Konvergenzordnungen 115

B.1 Realisierte Kombinationen der Routine NUDOCCCS 131

Abbildungsverzeichnis

2.1 Ein- bzw. Austrittspunkt 20
2.2 Kontaktpunkt 21

3.1 Das Einfach-Schießverfahren 29
3.2 Das Mehrfach-Schießverfahren 32
3.3 Diskretes Gebiet 34

4.1 Blockstruktur der Jacobi-Matrix 43
4.2 Zuverlässigkeit von Verfahren höherer Ordnung 45
4.3 Lineare Interpolation der Steuerung 47
4.4 Kubische Interpolation des Zustandes 48
4.5 Bidiagonalblockstruktur der Jacobi-Matrix 52
4.6 Kombination der Lösungsansätze 53
4.7 Blockstruktur der Jacobi-Matrix 62
4.8 Direktes Verfahren durch Problemsharing 64
4.9 Eine spezielle 'Schlauchtechnik' 84

5.1 Gittersystem $\mathcal{D}_k$ 86
5.2 Verladebrücke (Erzentlader) 87
5.3 Optimale Trajektorien der unbeschränkten Verladebrücke . . . 90
5.4 Optimale Trajektorien der durch $x_2 \leq 0.4$ beschränkten Verladebrücke 92
5.5 'Chattering-control' 92
5.6 Chemischer Reaktor 93
5.7 Optimale Trajektorien des chemischen Reaktors für $u_{max} = 1$. 94
5.8 Optimale Steuerung des chemischen Reaktors für $u_{max} = 1$ . . 95
5.9 Optimale Trajektorien des unbeschränkten Spline-Problems . . 99
5.10 Optimale Steuerung des unbeschränkten Spline-Problems . . . 99
5.11 Optimale Trajektorien des durch $x_1 \leq 0.2$ beschränkten Spline-Problems 101
5.12 Optimale Steuerung des durch $x_1 \leq 0.2$ beschränkten Spline-Problems 101
5.13 Optimale Trajektorien des durch $x_1 \leq 0.1$ beschränkten Spline-Problems 103

5.14 Optimale Steuerung des durch $x_1 \leq 0.1$ beschränkten Spline-Problems . . . 103
5.15 Knickstab . . . 104
5.16 Optimale Trajektorien des Knickstabes für $\alpha = 9.90027$ . . . 107
5.17 Optimale Steuerung des Knickstabes für $\alpha = 9.90027$ . . . 107
5.18 Optimale Trajektorien des Knickstabes für $\alpha = 39.3291$ . . . 108
5.19 Optimale Steuerung des Knickstabes für $\alpha = 39.3291$ . . . 109
5.20 Optimale Trajektorien des Knickstabes für $\alpha = 105.190$ . . . 110
5.21 Optimale Steuerung des Knickstabes für $\alpha = 105.190$ . . . 111
5.22 Koordinatensystem der drei dimensionalen Darstellungen . . . 112
5.23 Das Auftreten von Nebenminima in Abhängigkeit von der Anfangsschätzung . . . 112

Kapitel 1

Einleitung

Da aber die Gestalt des ganzen Universums höchst vollkommen ist, entworfen von dem weisesten aller Schöpfer, so geschieht in der Welt nichts, ohne das sich irgendwie eine Maximums- oder Minimumsregel zeigt.
Leonhard Euler (1707-1783)

In vielen Gebieten gewinnt die mathematische Modellierung realer Prozesse mit Systemen von Differentialgleichungen immer größere Bedeutung. Die Ziele, einen hohen Aufwand für Experimente am realen Prozeß zu ersetzen, sowie eine wissenschaftliche Einsicht in die Natur des Prozesses zu bekommen, werden hierbei durch kontrollierte Simulationen der mathematischen Modelle auf Rechenanlagen erreicht. Oftmals besteht dabei die Möglichkeit, den Zustand eines solchen Systems von außen zu beeinflussen. Allerdings soll in der Regel eine solche 'Steuerung' nicht ziellos und unkontrolliert erfolgen, vielmehr steht die Optimierung eines Leistungsmaßes unter Einhaltung gewisser Nebenbedingungen im Vordergrund. Viele Optimierungsprobleme in den Natur-, Ingenieur- und Wirtschaftswissenschaften können mathematisch durch optimale Steuerprobleme beschrieben werden.

Typische Anwendungen sind z.B. die Steuerung von Raumfahrzeugen, chemischen Reaktionen, Robotern oder wirtschaftswissenschaftlichen Vorgängen.

Die mathematisch-analytische Behandlung mit Hilfsmitteln der Theorie optimaler Steuerungen (ein Teilgebiet der klassischen Variationsrechnung) und die Herleitung der aus den notwendigen Bedingungen dieser Theorie folgenden Mehrpunktrandwertprobleme, stellen auch heute noch sehr hohe Anforderungen an den Anwender. Aus historischer Sicht motivierte dabei insbesondere das Aufkommen elektronischer Rechenanlagen die rasche Entwicklung der optimalen Steuerungstheorie.

Grundsätzlich lassen sich die numerischen Methoden zur Lösung optimaler Steuerprozesse in zwei Klassen aufteilen: Indirekte und direkte Verfahren.

Bei den indirekten Verfahren werden die notwendigen Bedingungen aus der Theorie der optimalen Steuerprozesse genutzt, um den Prozeß auf ein Mehrpunktrandwertproblem zurückzuführen. Für die Lösung solcher, in der Regel sehr komplizierten Probleme, steht mit der Fortran-Routine BDSCO ein leistungsfähiges Verfahren zur Verfügung. Die Anwendung dieser Methode erfordert umfangreiche Kenntnisse der Theorie optimaler Steuerprozesse, liefert aber andererseits eine Fülle von Informationen und damit tiefe Einsichten in die Struktur der optimalen Lösung. Dabei stellt die Benutzung der adjungierten Variablen auf der einen Seite eine gewisse Benutzerunfreundlichkeit dar, auf der anderen Seite gestattet sie aber eine Überprüfung vieler notwendiger Bedingungen und damit eine Aussondierung nichtoptimaler Lösungen. Insbesondere zeichnet sich die so gefundene Lösung durch eine hohe Genauigkeit aus.

Bei den direkten Verfahren wird das unendlich dimensionale Optimalsteuerungsproblem auf ein endlich dimensionales Optimierungsproblem zurückgeführt, zu dessen Lösung eine umfangreiche Anzahl numerischer Optimierungsverfahren zur Verfügung steht. Erste Formulierungen optimaler Steuerprozesse als abstrakte Optimierungsprobleme sind bereits seit den 60'er Jahren bekannt. Desweiteren zeichnen sich die direkten Verfahren durch eine gewisse Anwenderfreundlichkeit aus, da sie, bis auf einige Ausnahmen, keine Kenntnisse der Theorie optimaler Steuerungen erfordern. Außerdem besitzen sie einen im Verhältnis zu den indirekten Verfahren relativ großen Konvergenzradius, während jedoch die erzielbare Genauigkeit der Lösung oftmals nur gering ist.

In dieser Arbeit sollen verschiedene direkte Verfahren (allesamt Iterationsverfahren) vorgestellt und anhand unterschiedlicher Anwendungsbeispiele miteinander verglichen werden. Dazu werden im Anschluß an diese Einleitung im zweiten Kapitel die für die weiteren Untersuchungen wichtigen Ergebnisse der Theorie optimaler Steuerprozesse bereitgestellt. Kapitel 3 beschäftigt sich mit verschiedenen Verfahren zur Lösung von Randwertproblemen, die zum einen eine wesentliche Bedeutung bei den indirekten Verfahren haben, andererseits aber auch bei den direkten Verfahren ihre Anwendung finden, die in Kapitel 4 beschrieben werden sollen. Ergänzend werden in diesem Kapitel auch einige Modifikationstechniken des Ausgangsproblems dargestellt. In Kapitel 5 werden schließlich einige optimale Steuerprozesse vorgestellt und sowohl mit einem indirekten als auch den verschiedenen direkten Ansätzen gelöst und bezüglich unterschiedlicher Kriterien wie Effizienz, Zuverlässigkeit und Genauigkeit der Lösungen miteinander verglichen. Die Beschreibungen aller benutzten Fortran-Routinen befindet sich im Anhang.

Alle hier realisierten direkten Verfahren wurden von mir erstellt. Ebenso wurde eigens ein Programm zur Erstellung der Grafiken von mir geschrieben.

Die Berechnungen wurden auf Unix-Risc-Rechnern des Typs IBM RS-6000 des Universitätsrechenzentrum Münster durchgeführt.

Sämtliche Programme sind auf einer Diskette am Ende der Arbeit zugänglich.

Kapitel 2

Zur Theorie optimaler Steuerprozesse

In diesem Kapitel sollen in Anlehnung an [Maurer 1989] einige wichtige Sätze und Definitionen der Steuerungstheorie bereitgestellt werden, deren Bedeutung in den folgenden Kapiteln ihre Anwendung findet. Ziel ist es (analog zu den *Kuhn-Tucker*[1]*-Bedingungen* der nichtlinearen Optimierung) eine notwendige Bedingung für eine optimale Steuerung anzugeben.
Erreicht wird dies durch das *Pontrjaginsche*[2]*-Minimumsprinzip*, welches als zentrale Aussage dieser Theorie angesehen werden kann. Es geht um das Problem, ein durch gewöhnliche Differentalgleichungen gegebenes System durch eine Steuerfunktion so zu beeinflussen, daß ein Zielfunktional (oder auch eine Kostenfunktion) minimiert wird. Hierbei ist ein zeitoptimales Problem lediglich ein Sonderfall, bei dem die Endzeit als Funktional minimiert wird.
Mittlerweile findet diese Theorie in vielen Bereichen ihre Anwendung, z.B. in technischen Anwendungen wie Luft- und Raumfahrt oder Robotik, in wissenschaftlichen Gebieten wie Physik, Biologie oder Chemie. Aber auch in den Wirtschaftswissenschaften gewinnt sie immer mehr an Bedeutung.

2.1 Grundlagen

Betrachtet wird ein System von n Differentialgleichungen 1. Ordnung, dessen Zustand zum Zeitpunkt t durch den *Zustandsvektor*

$$x(t) = (x_1(t), \ldots, x_n(t))^T \in \mathbb{R}^n \tag{2.1}$$

[1]Albert William Tucker, geb. 1905 in Oshawa, Kanada
[2]Lew Semjonowitsch Potrjagin, geb. 1908 in Moskau, Rußland

beschrieben wird. Dieser Zustand kann zum Zeitpunkt t durch den *Steuervektor*

$$u(t) = (u_1(t), \ldots, u_m(t))^T \in \mathbb{R}^m \tag{2.2}$$

beeinflußt werden, der stets eine Funktion der Systemzeit t ist.

Definition 2.1 (Dynamik des Systems) *Das System von Differentialgleichungen*

$$\dot{x}(t) := \frac{dx(t)}{dt} = f(x(t), u(t), t) \tag{2.3}$$

nennt man Dynamik des Systems. *Hierbei sei*

$$f : \mathbb{R}^n \times \mathbb{R}^m \times [t_0, t_f] \longrightarrow \mathbb{R}^n \tag{2.4}$$

eine stetige und bezüglich x und u stetig partiell differenzierbare Funktion. $t_0 \in \mathbb{R}$ bezeichne den Anfangs- *und $t_f \in \mathbb{R}$ den* Endzeitpunkt $(t_f > t_0)$.

In der praktischen Anwendung kann t_f fest vorgegeben, oder eine freie Variable sein.
Der *Anfangszustand* $x(t_0)$ und der *Endzustand* $x(t_f)$ genüge den *Randbedingungen*

$$\begin{array}{rcl} x(t_0) & \in & M_{t_0} \\ x(t_f) & \in & M_{t_f} \end{array} \tag{2.5}$$

mit abgeschlossenen Mengen $M_{t_0}, M_{t_f} \in \mathbb{R}^n$.
In den nachfolgenden Untersuchungen beschränkt man sich auf den in der Praxis relevanteren Fall, bei dem M_{t_0} und M_{t_f} gleichungsdefinierte Flächen sind:

$$\begin{array}{rcl} M_0 & := & \{x \in \mathbb{R}^n \mid \varphi(x) = 0\} \\ M_{t_f} & := & \{x \in \mathbb{R}^n \mid \psi(x) = 0\} \end{array} \tag{2.6}$$

mit den stetig differenzierbaren Funktionen $\varphi : \mathbb{R}^n \longrightarrow \mathbb{R}^s$ und
$\psi : \mathbb{R}^n \longrightarrow \mathbb{R}^r$, $0 \leq s, r \leq n$. Somit gelangt man zu einer neuen Formulierung für die Randbedingungen

$$\begin{array}{rcl} \varphi(x(t_0)) & = & 0 \\ \psi(x(t_f)) & = & 0 \end{array} \tag{2.7}$$

Bemerkung 2.2 *Durch $s = 0$ bzw. $r = 0$ wird formal der Fall berücksichtigt, daß für $x(t_0)$ bzw. $x(t_f)$ keine Bedingungen vorgegeben sind.*

Für ein festes $x_0 \in \mathbb{R}^n$ gilt häufig $\varphi(x) = x - x_0$, damit gelangt man zur

Definition 2.3 (Standard-Randbedingungen) *Randbedingungen der Art*

$$\begin{array}{rcl} x(t_0) & = & x_0 \\ \psi(x(t_f)) & = & 0 \end{array} \tag{2.8}$$

werden als Standard-Randbedingungen *bezeichnet.*

Definition 2.4 (Steuerbereich) *Die nichtleere, konvexe und abgeschlossene Menge $U \subset \mathbb{R}^m$ wird als* Steuerbereich *bezeichnet und beschreibt die zulässige Wertemenge für den Steuervektor $u(t)$:*

$$u(t) \in U, \qquad \forall\, t \in [t_0, t_f] \tag{2.9}$$

Definition 2.5 (Lösung des Systems) *Ein Funktionenpaar*

$$(x, u) \quad mit \quad x : [t_0, t_f] \longrightarrow \mathbb{R}^n, \quad u : [t_0, t_f] \longrightarrow \mathbb{R}^m \tag{2.10}$$

und x stetig und stückweise stetig differenzierbar, u stückweise stetig, heißt Lösung *von (2.3), falls*

$$\dot{x}(t) = \frac{dx}{dt} = f(x(t), u(t), t) \qquad \forall\, t \in [t_0, t_f] \tag{2.11}$$

für alle Stetigkeitsstellen $t \in [t_0, t_f]$ von u.

Definition 2.6 (zulässige Lösung) *Ein Lösungspaar (x, u) von (2.3) wird* zulässig *zur Endzeit t_f genannt, falls zusätzlich (2.5) und (2.9) erfüllt sind.*

Definition 2.7 (Zielfunktional) [3] *Eine Funktion*

$$F(x, u, t) = g(x(t_f)) + \int_{t_0}^{t_f} f_0(x(t), u(t), t)\, dt \tag{2.12}$$

wird als Zielfunktional *bezeichnet und ist die bei einem Steuerprozeß zu optimierende Größe. Dabei gelte: $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ stetig differenzierbar, und $f_0 : \mathbb{R}^n \times \mathbb{R}^m \times [t_0, t_f] \longrightarrow \mathbb{R}$ stetig und bezüglich x, u stetig partiell differenzierbar. Die partiellen Ableitungen $f_{0,x} := \bigtriangledown_x f_0, f_{0,u} := \bigtriangledown_u f_0$ seien stetig.*

Zusammenfassend wird ein optimaler Steuerprozeß durch folgende Charakteristika beschrieben:

1. Dynamik des Systems
2. Zustandsrandbedingungen
3. Steuerbereich
4. Zielfunktional

Diese Eigenschaften eines optimalen Steuerprozesses lassen sich in einer mathematisch-kompakten Form durch die Vereinbarungen (2.1)-(2.12) beschreiben. Dabei wird in der Regel ohne Einschränkung $t_0 = 0$ gesetzt,welches zusammenfassend zu folgender Definition Anlaß gibt:

[3] In der Literatur auch unter der Bezeichnung *Kostenfunktion* zu finden.

Definition 2.8 (Optimaler Steuerprozeß) *Das Problem*

$$\begin{aligned} \textit{Minimiere} \quad F(x,u,t) &= g(x(t_f)) + \int_0^{t_f} f_0(x(t),u(t),t)\,dt \\ \textit{unter} \qquad\qquad & \\ \dot{x} &= f(x,u,t) \\ x(0) &\in M_{t_0} \\ x(t_f) &\in M_{t_f} \\ u(t) &\in U, \qquad 0 \le t \le t_f \end{aligned} \tag{2.13}$$

heißt optimaler Steuerprozeß.

Ein Steuerproblem dieser Art wird auch *Bolza-Problem* genannt. Man kann zeigen, daß diese Form des Zielfunktionals äquivalent zu einem *Lagrange*[4]-*Problem*, einem *Mayer-Problem* oder einem *Tschebyscheff*[5]-*Problem* ist und daß alle Formen ineinander transformierbar sind (vgl. Kapitel 2.3). Der formale Unterschied liegt lediglich in der Beschreibung des Zielfunktionals: [6]

$$\begin{aligned} \text{Lagrange-Problem:} \quad F(x,u) &= \int_0^{t_f} f_0(x(t),u(t),t)\,dt \\ \text{Mayer-Problem:} \quad F(x,u) &= g(x(t_f)) \\ \text{Tschebyscheff-Problem:} \quad F(x,u) &= \max_{t\in[0,t_f]} f_0(x(t),u(t),t) \end{aligned} \tag{2.14}$$

Die folgenden Betrachtungen dienen der Behandlung optimaler Steuerprozesse in Form von *Bolza-Problemen*, mit den aus (2.8) abgeleiteten Standard-Randbedingungen.

$$\begin{aligned} \text{Minimiere} \quad F(x,u) &= g(x(t_f)) + \int_0^{t_f} f_0(x(t),u(t),t)\,dt \\ \text{unter} \qquad\qquad & \\ \dot{x} &= f(x,u,t) \\ x(0) &= x_0, \qquad \psi(x(t_f)) = 0 \\ u(t) &\in U, \qquad 0 \le t \le t_f. \end{aligned} \tag{2.15}$$

[4] Joseph Louis Lagrange, geb. 1736 Turin, gest. 1813 Paris

[5] Pafnuti Lwowitsch Tschebyschow, geb. 1821 Okatowo, gest. 1894, Petersburg (St. Petersburg)

[6] Die Frage nach dem Vorzug einer der Problembeschreiber kann nicht ohne weiteres beantwortet werden. Vielmehr hängt eine geeignete Wahl vom weiteren Umfeld des Problems ab. Für numerische Auswertungen empfiehlt es sich sicher, ein *Mayer-Problem* zu behandeln und etwaige Integrale im Funktional über das Differentialgleichungssystem lösen zu lassen, während es sich bei analytischen Auswertungen genau umgekehrt verhalten kann.

Das resultierende mathematische Problem besteht nun darin, unter Erfüllung der vorgegebenen Bedingungen (Randwerte, Steuerbereich, Differentialgleichungssystem) eine Steuerung u zu finden, die das vorgegebene Funktional minimiert. Bei der praktischen Lösung von (2.15) unterscheidet man zwei verschiedene Fälle. Zum einen, daß die Endzeit t_f fest vorgegeben ist und zum anderen, daß die Endzeit t_f eine freie, unbestimmte Variable darstellt:

Definition 2.9 (Feste Endzeit) *Ein zulässiges Funktionenpaar* (x^*, u^*) *zur festen Endzeit* t_f *heißt* optimale Lösung *des durch (2.15) erklärten Steuerprozesses, falls*

$$F(x^*, u^*) \leq F(x, u)$$

für alle zulässigen Paare (x, u) *zur Endzeit* t_f.

Definition 2.10 (Freie Endzeit) *Ein zulässiges Funktionenpaar* (x^*, u^*) *zur freien Endzeit* t_f *heißt* optimale Lösung *des in (2.15) erklärten Steuerprozesses, falls*

$$F(x^*, u^*) \leq F(x, u)$$

für alle zulässigen Paare (x, u) *zur beliebigen Endzeit* $t_f > 0$.

In beiden Fällen nennt man x^* *optimale Trajektorie* und u^* *optimale Steuerung.*

Bemerkung 2.11 *Ein Sonderfall der optimalen Steuerprozesse mit freier Endzeit sind die* zeitoptimalen Steuerprozesse, *bei denen es ein einfaches Zielfunktional gibt:*

$$F(x, u) = t_f = \int_0^{t_f} 1 \, dt \tag{2.16}$$

Grundsätzlich ist es möglich, Steuerprobleme mit freier Endzeit auf Steuerprobleme mit fester Endzeit zurückzuführen (vgl. Kapitel 2.3).
Für spätere Anwendungen ist ferner der Begriff des *autonomen Problems* von Bedeutung. Ein Steuerproblem wird *autonom* genannt, wenn die Funktionen f_0 und f nicht explizit von t abhängen. Nichtautonome Probleme lassen sich aber immer auf autonome Systeme zurückführen (vgl. Kapitel 2.3).

2.2 Das unbeschränkte Steuerproblem

Das Pontryaginsche Minimumprinzip ist die zentrale Aussage der Theorie optimaler Steuerprozesse. Bevor jedoch dieser Satz über notwendige Bedingungen für eine optimale Lösung (x^*, u^*) angegeben werden kann, ist die folgende Definition der *Hamilton*[7]*-Funktion* von wesentlicher Bedeutung. Mit dieser lassen sich die notwendigen und (unter zusätzlichen Voraussetzungen) auch hinreichenden Optimalitätsbedingungen für den unter (2.15) formulierten Steuerprozeß angeben. Gleichzeitig stellen das Pontryaginsche Minimumprinzip und die Hamilton-Funktion eine wichtige Grundlage für die folgenden Kapitel dar, in denen steuer- und zustandsbeschränkte Probleme behandelt werden sollen.

Definition 2.12 (Hamilton-Funktion) *Sei*

$$\lambda_0 \in \mathbb{R} \qquad \textit{und} \qquad \lambda : [0, t_f] \longrightarrow \mathbb{R}^n$$

Die Funktion

$$H(t, x, \lambda, u) := \lambda_0 f_0(x, u, t) + \lambda f(x, u, t) = \sum_{k=0}^{n} \lambda_k f_k(x, u, t)$$

heißt Hamilton-Funktion *zum Steuerprozeß (2.15).*
Der Zeilenvektor $\lambda(t) \in \mathbb{R}^n$ *wird* adjungierte Variable *oder* Kozustand *zu* x *genannt.*

Bemerkung 2.13 *Ist der optimale Steuerprozeß in* Mayer-*Form gegeben, so vereinfacht sich die Hamiton-Funktion zu*

$$H(t, x, \lambda, u) := \lambda f(x, u, t) = \sum_{k=1}^{n} \lambda_k f_k(x, u, t).$$

Bemerkung 2.14 *Aufgrund der Definition der Hamilton-Funktion* H *und der adjungierten Differentialgleichungen, ist ein* Hamiltonisches System *der Form*

$$\dot{x} = \left(\frac{\partial H}{\partial \lambda}\right)^T, \quad \dot{\lambda} = -\frac{\partial H}{\partial x}$$

gegeben.

Die Bedeutung der skalarwertigen Hamiton-Funktion wird im folgenden Satz deutlich:

Satz 2.15 (Pontryaginsches Minimumprinzip) *Für eine optimale Lösung* (x^*, u^*) *von (2.15) existieren eine reelle Zahl* $\lambda_0 \geq 0$, *eine stetige und stückweise stetig differenzierbare Funktion*

$$\lambda : [0, t_f] \longrightarrow \mathbb{R}^n,$$

sowie ein Zeilenvektor $\nu \in \mathbb{R}^r$, *so daß gelten:*

[7]Sir William Rowan Hamilton, geb. 1805 Dublin, gest. 1865 Dunsik

1. $(\lambda_0, \lambda(t), \nu) \neq 0 \qquad \forall\, t \in [0, t_f]$,

2. An allen Stetigkeitsstellen t von u^* in $[0, t_f]$ gelten:

(a) Minimumbedingung:

$$H(t, x^*(t), \lambda(t), u^*(t)) = \min_{u \in U} H(t, x^*(t), \lambda(t), u),$$

(b) Adjungierte Differentialgleichungen:

$$\dot{\lambda} = -H_x(t, x^*(t), \lambda(t), u^*(t)),$$

3. Transversalitätsbedingung[8]*:*

$$\lambda(t_f) = \lambda_0 g_x(x^*(t_f)) + \nu \psi_x(x^*(t_f)).$$

In Erweiterung des Satzes 2.15 können die beiden folgenden Bemerkungen von Bedeutung sein:

Bemerkung 2.16 *Im Falle freier Endzeit gilt für die optimale Endzeit t_f^*:*

$$H(t_f^*, x^*(t_f^*), \lambda(t_f^*), u^*(t_f^*)) = 0.$$

Bemerkung 2.17 *Im Falle autonomer Probleme gilt:*

$$H(x^*(t), \lambda(t), u^*(t)) = \text{const.} \quad \forall\, t \in [0, t_f].$$

Einen Beweis des Minimumprinzips findet man in [Pontrjagin 1967] oder [Sethi 1981]. In [Knobloch 1974] werden die Bemerkungen 2.16 und 2.17 nachgewiesen.

Die Aussagen des Minimumprinzips gehen durch eine Skalierung

$$\lambda \longrightarrow \frac{\lambda}{\lambda_0}$$

nicht verloren. Desweiteren können die Bedingungen aus Satz 2.15 durch Setzen von $\lambda_0 = 1$ in der Hamilton-Funktion vereinfacht werden, sofern $\lambda_0 > 0$

[8]Im Falle von Randbedingungen der Form (2.7) verallgemeinern sich diese Bedingungen bei einem Lagrange-Problem zu

$$\lambda(0) = -\frac{\partial}{\partial x(0)} \nu_1 \varphi(x^*(0)), \qquad \lambda(t_f) = \frac{\partial}{\partial x(t_f)} \nu_2 \psi(x^*(t_f))$$

mit gewissen Zeilenvektoren $\nu_1 \in \mathbb{R}^s$, $\nu_2 \in \mathbb{R}^r$.

gilt. Der Nachweis von $\lambda_0 > 0$ erfordert jedoch in der Regel zusätzliche Regularitätseigenschaften der optimalen Lösung (x^*, u^*). Der Versuch $\lambda_0 = 0$ zum Widerspruch zu führen, kann bei konkreten Problemen weiterhelfen, wie dies zum Beispiel allgemein für Probleme mit freiem Endzustand in [Maurer 1989] gelingt.
Unter zusätzlichen Konvexitätsannahmen kann man für den Fall $\lambda_0 = 1$ zeigen, daß auch eine hinreichende Bedingung für die Existenz einer optimalen Lösung (x^*, u^*) durch das Pontryaginsche-Minimumprinzip gegeben ist. Es gilt der folgende Satz:

Satz 2.18 *Sei (x^*, u^*) ein zulässiges Paar für das Problem (2.15) und es existiere eine Funktion $\lambda : [0, t_f] \longrightarrow \mathbb{R}^n$, sowie ein Vektor $\nu \in \mathbb{R}^r$, so daß die notwendigen Optimalitätsbedingungen von Satz 2.15 mit $\lambda_0 = 1$ erfüllt sind. Ferner sei*

(a) $g(x)$ konvex,

(b) $\psi(x)$ affin-linear,

(c) $H^0(t, x, \lambda) := \min\limits_{u \in U} H(t, x, \lambda, u)$ konvex in x für jedes $(t, \lambda(t))$, $t \in [0, t_f]$.

Dann ist (x^*, u^*) eine optimale Lösung.

Zum Beweis vergleiche man [Feichtinger 1986].

2.3 Rückführung optimaler Steuerprozesse auf ein Ausgangsproblem

Nachfolgend sollen in Kurzform Möglichkeiten zur Rückführung verschiedener optimaler Steuerprobleme auf ein Ausgangsproblem aufgezeigt werden. Tabelle 2.1 gibt einen Überblick über nachfolgende Transformationen:

1.	freie Endzeit	$\longrightarrow$	feste Endzeit
2.	nicht autonom	$\longrightarrow$	autonom
3.	Bolza	$\longrightarrow$	Mayer
4.	Lagrange	$\longrightarrow$	Mayer
5.	Tschebyscheff	$\longrightarrow$	Mayer

originale Formulierung	äquivalente Formulierung
freie Endzeit (Mayer-Problem) min $g(x(t_f))$ unter $\dot{x} = f(x,u,t), \quad 0 \leq t \leq t_f,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $u \in U,\ t_f$ frei	feste Endzeit (Mayer-Problem) min $g(x(t_f))$ unter $\dot{x} = f(x,u,t)y, \quad 0 \leq t \leq 1,$ $\dot{y} = 0,$ $x(0) = x_0,\ \psi(x(1)) = 0,$ $u \in U$
nicht autonom (feste Endzeit) min $g(x(t_f))$ unter $\dot{x} = f(x,u,t), \quad 0 \leq t \leq t_f,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $u \in U$	autonom (feste Endzeit) min $g(x(t_f))$ unter $\dot{x} = f(y,x,u), \quad 0 \leq t \leq t_f,$ $\dot{y} = 1,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $y(0) = 0,\ u \in U$
Bolza-Steuerprozeß min $g(x(t_f)) + \int_0^{t_f} f_0(x,u,t)\,dt$ unter $\dot{x} = f(x,u,t), \quad 0 \leq t \leq t_f,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $u \in U$	Mayer-Steuerprozeß min $g(x(t_f)) + y(t_f)$ unter $\dot{x} = f(x,u,t),$ $\dot{y} = f_0(x,u,t), \quad 0 \leq t \leq t_f,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $y(0) = 0,\ u \in U$
Lagrange-Steuerprozeß min $\int_0^{t_f} f_0(x,u,t)\,dt$ unter $\dot{x} = f(x,u,t), \quad 0 \leq t \leq t_f,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $u \in U$	Mayer-Steuerprozeß min $y(t_f)$ unter $\dot{x} = f(x,u,t),$ $\dot{y} = f_0(x,u,t), \quad 0 \leq t \leq t_f,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $y(0) = 0,\ u \in U$
Tschebyscheff-Steuerprozeß min $(\max_{t \in [0,t_f]} f_0(x,u,t))$ unter $\dot{x} = f(x,u,t), \quad 0 \leq t \leq t_f,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $u \in U$	Mayer-Steuerprozeß min $y(t_f)$ unter $\dot{x} = f(x,u,t), \quad 0 \leq t \leq t_f,$ $\dot{y} = 0,$ $x(0) = x_0,\ \psi(x(t_f)) = 0,$ $u \in U,$ $f_0(x,u,t) - y \leq 0$

Tabelle 2.1: Rückführung optimaler Steuerprozesse auf ein Ausgangsproblem

2.4 Bang-bang und singuläre Steuerungen – Probleme mit linear eingehender Steuerung

Viele in der Praxis betrachtete Steuerprozesse sind gekennzeichnet durch Steuerungen, die sowohl in der Dynamik als auch im Zielfunktional linear auftreten. Steuerprozesse dieser Art werden *Probleme mit linear eingehender Steuerung* genannt. Die Betrachtung von Problemen mit nichtlinear eingehender Steuerung führt auf den Begriff der *regulären Hamiltonfunktion* und soll im nächsten Kapitel eingehender behandelt werden.
Die Unterscheidung von Problemen mit linear und nichtlinear auftretender Steuerung ist typisch für viele Bereiche der Optimierung, da für beide Aufgabenstellungen der Lösungsweg grundsätzlich verschieden ist.
Da die regulären Hamiltonprobleme aber den überwiegenden Teil der Anwendungsbeispiele in dieser Arbeit bilden, soll an dieser Stelle nur eine kurze Zusammenfassung der wesentlichen Ergebnisse der Probleme mit linear eingehender Steuerung gegeben werden.

Ausgehend vom Steuerprozeß (2.15) betrachtet man den Spezialfall:

$$\begin{aligned} f(x,u,t) &= a(t,x) + b(t,x)u \\ f_0(x,u,t) &= a_0(t,x) + b_0(t,x)u \end{aligned} \tag{2.17}$$

mit $f(x,u,t) \in \mathbb{R}^n,\ u(t) \subset U \subset \mathbb{R}^m$.
Dabei sei a ein $n \times 1$ Vektor, b eine $n \times m$ Matrix, a_0 ein Skalar, b_0 ein $1 \times m$ Vektor, U konvex und kompakt.
Die Hamiltonfunktion ergibt sich wegen (2.17) zu

$$H(t,x,\lambda,u) = \lambda_0 a_0(t,x) + \lambda a(t,x) + [\lambda_0 b_0(t,x) + \lambda b(t,x)]u. \tag{2.18}$$

Für die weiteren Betrachtungen ist der Begriff der *Schaltfunktion* von Bedeutung.

Definition 2.19 (Schaltfunktion)

$$\Phi(t) := H_u(t,x(t),\lambda(t),u(t)) = \lambda_0 b_0(t,x(t)) + \lambda(t) b(t,x(t)) \tag{2.19}$$

heißt Schaltfunktion.

Insbesondere gilt somit

$$H(t,x,\lambda,u) = \lambda_0 a_0(t,x) + \lambda a(t,x) + \Phi(t)u \tag{2.20}$$

Die Hamiltonfunktion wird also genau dann bezüglich u minimal, wenn $\Phi(t)u$ minimal wird. Damit hat man das äquivalente lineare Optimierungsproblem

$$\min\{\Phi(t)u | u \in U\} \qquad \forall\, t \in [0, t_f]. \tag{2.21}$$

Aus Gründen der Vereinfachung beschränkt man sich auf den Fall $m = 1$, $U = [u_{min}, u_{max}] \subset \mathbb{R}$. Die optimale Steuerung ist daher durch

$$u^*(t) = \left\{ \begin{array}{ll} u_{min}, & \Phi(t) > 0 \\ u_{max}, & \Phi(t) < 0 \\ \text{unbestimmt}, & \Phi(t) = 0 \end{array} \right\} \tag{2.22}$$

gegeben.

Definition 2.20 *Sei* $[t_1, t_2] \subset [0, t_f]$, $t_1 < t_2$

(i) $u(t)$ *heißt* bang-bang *in* $[t_1, t_2]$, *wenn* $\Phi(t)$ *in* $[t_1, t_2]$ *nur isolierte Nullstellen hat, d.h.* $u(t) \in \{u_{min}, u_{max}\}$. *Die Nullstellen von* $\Phi(t)$ *heißen* Schaltpunkte.

(ii) $u(t)$ *heißt* singulär *in* $[t_1, t_2]$, *falls* $\Phi(t) \equiv 0$ *in* $[t_1, t_2]$. *Die Zeitpunkte* t_1, t_2 *heißen Verbindungspunkte, falls* $u(t)$ *bang-bang in* $[t_1 - \varepsilon], [t_2 + \varepsilon]$ *für* $\varepsilon > 0$ *hinreichend klein.*

Geht die Zustandsvariable x nichtlinear in die Hamiltonfunktion ein, so können singuläre Steuerungen auftreten.
Nachfolgende Betrachtungen dienen zur expliziten Bestimmung von singulären Steuerungen. Dazu liege ein Intervall $[t_1, t_2] \subset [0, t_f]$ vor mit

$$\Phi(t) = 0, \qquad \forall\, t \in [t_1, t_2] \tag{2.23}$$

weiterhin sei $\Phi(t)$ hinreichend oft differenzierbar.
Damit gilt ebenfalls

$$\Phi^{(k)} := \frac{d^k}{dt^k}\Phi(t) = 0, \quad \forall\, t \in [t_1, t_2],\ k \geq 1. \tag{2.24}$$

Entweder gilt dann

$$\frac{\partial}{\partial u}\Phi^{(k)} \equiv 0 \quad \forall\, k \geq 0 \tag{2.25}$$

oder es gibt ein $\bar{k} > 0$ mit

$$\begin{aligned} \frac{\partial}{\partial u}\Phi^{(k)} &\equiv 0 \qquad \forall k = 0, \ldots, \bar{k} - 1, \\ \frac{\partial}{\partial u}\Phi^{(\bar{k})} &\not\equiv 0. \end{aligned} \tag{2.26}$$

Für $\bar{k} < \infty$ gilt

$$\Phi^{(\bar{k})} = A(t, x, \lambda) + B(t, x, \lambda)u, \tag{2.27}$$

da u linear auftritt.
Falls $B(t, x(t), \lambda(t)) \neq 0$, folgt wegen (2.24) für alle $t \in [t_1, t_2]$ die singuläre Steuerung

$$u_{sing}(t) = -\frac{A(t, x, \lambda)}{B(t, x, \lambda)} \tag{2.28}$$

Satz 2.21

(i) Ist $\bar{k} < \infty$, so ist $\bar{k} = 2q$ eine gerade Zahl. Die Zahl $q \geq 1$ heißt die Ordnung der singulären Steuerung.

(ii) Für eine optimale Lösung $x(t)$, $u_{sing}(t)$, $\lambda(t)$ gilt die verallgemeinerte Legendre-Clebsch-Bedingung

$$0 \leq (-1)^q B(t, x, \lambda) = (-1)^q \frac{\partial}{\partial u}\left[\frac{d^{2q}}{dt^{2q}} H_u(t)\right] \tag{2.29}$$

($B(t, x, \lambda)$ und $(-1)^q$ besitzen also gleiche Vorzeichen).

In [Knobloch 1974] findet man belegend hierzu einen Beweis.

2.5 Reguläre Hamiltonprobleme, –Probleme mit nichtlinear auftretender Steuerung

Bei Problemen mit linear auftretender Steuerung kommt es zu der häufigen Situation, daß die Steuerung unstetig in gewissen Punkten ist. Es soll aber im Gegensatz hierzu gezeigt werden, daß im Fall der sogenannten *regulären* Hamilton-Funktion die optimale Steuerung stetig und für $U = \mathbb{R}^m$ sogar eine C^k-Funktion ist.
Man orientiere sich wieder am Steuerprozeß (2.15). Weiterhin seien alle auftretenden Funktionen für $k \geq 1$ aus der Klasse der C^k. Für eine optimale Lösung $x(t)$, $u(t)$ sei das Pontryaginsche Minimumprinzip 2.15 mit $\lambda_0 = 1$ und der adjungierten Variablen $\lambda(t)$ erfüllt, dann definieren wir:

Definition 2.22 (reguläre Hamiltonfunktion) *Die Hamilton-Funktion H heißt* regulär *(bzgl. einer optimalen Lösung $x(t)$), wenn es ein $\varepsilon > 0$ gibt, so daß für die Menge*

$$D_\varepsilon := \{(t, x, \lambda) | t \in [0, t_f], \|x - x(t)\| < \varepsilon, \|\lambda - \lambda(t)\| < \varepsilon\} \tag{2.30}$$

gilt:
Die Funktion $H(t,x,\lambda,\cdot)$ hat eine eindeutig bestimmte Minimalstelle

$$u^*(t,x,\lambda) = \arg\min_{u\in U} H(t,x,\lambda,u), \qquad \forall\ (t,x,\lambda)\in D_\varepsilon.$$

Ist eine reguläre Hamilton-Funktion gegeben, dann ist die optimale Steuerung der eindeutig bestimmte Wert

$$u(t) = u^*(t, x(t), \lambda(t)) \tag{2.31}$$

Bei Problemen mit linear auftretender Steuerung ist die Hamilton-Funktion im allgemeinen nicht regulär, dies kann dazu führen, daß die Steuerung u an gewissen Punkten nicht eindeutig bestimmbar und dort unstetig ist. In diesem Zusammenhang seien die *bang-bang* Probleme als Beispiel genannt. Anders verhält es sich jedoch bei den Problemen mit regulärer Hamiltonfunktion, bei denen die optimale Steuerung stetig und für $U = \mathbb{R}^m$ sogar eine C^k-Funktion ist.
Man unterscheidet zwei Fälle bei der Untersuchung der Differenzierbarkeitseigenschaften der optimalen Steuerung $u(t)$:

1. $U = \mathbb{R}^m$

2. $U \subsetneqq \mathbb{R}^m$, U kompakt

1. Fall: $U = \mathbb{R}^m$

Aus der Minimumbedingung von Satz 2.15 erhält man entlang einer Lösung $x(t)$, $\lambda(t)$, $u(t)$:

$$\begin{aligned} H_u(t) &= 0, \\ H_{uu}(t) &\geq 0 \qquad \text{positiv semidefinit} \end{aligned} \tag{2.32}$$

(*Legendre*[9]*-Clebsch-Bedingung*)

Definition 2.23 (C^k-reguläre-Hamilton-Funktion) *Die Hamilton-Funktion H heißt C^k-regulär, wenn H regulär ist und die Funktion*

$$u^*(t,x,\lambda) = \arg\min_{u\in\mathbb{R}^m} H(t,x,\lambda,u)$$

auf der Menge D_ε (vgl. Definition 2.22) eine C^k-Funktion ist.

[9] Adrien Marie Legendre, geb. 1752 Paris, gest. 1833 Paris

Die Steuerung $u = u^*(t, x, \lambda)$ kann aus

$$\begin{array}{rcl} H_u(t, x, \lambda, u) & = & 0 \\ H_{uu}(t, x, \lambda, u) & > & 0 \end{array} \tag{2.33}$$

berechnet werden, wenn man die strenge Legendre-Clebsch-Bedingung $H_{uu}(t) > 0$ voraussetzt. Denn die C^k-Regularität hat die numerische Konsequenz, daß die Variable u in den Differentialgleichungen bzgl.

$$\begin{array}{rclcl} \dot{x} & = & f(x, u^*(t, x, \lambda), t) & =: & h_1(t, x, \lambda)^T \\ \dot{\lambda} & = & -H_x(t, x, \lambda, u^*(t, x, \lambda)) & =: & h_2(t, x, \lambda)^T \end{array} \tag{2.34}$$

eliminiert werden kann.
Die zusammengesetzte Funktion

$$h(t, x, \lambda) := \begin{pmatrix} h_1(t, x, \lambda) \\ h_2(t, x, \lambda) \end{pmatrix} \tag{2.35}$$

ist eine C^k-Funktion.
Gelten zusätzlich die in der Praxis häufig auftretenden vereinfachten Endbedingungen

$$x_i(t_f) = c_i, \quad i = 1, \ldots, r, \quad r \leq n, \tag{2.36}$$

so folgt daraus[10]

$$\begin{array}{rcl} \psi(x) & = & (x_1 - c_1, x_2 - c_2, \ldots, x_r - c_r)^T, \\ g(x) & = & g(x_{r+1}, x_{r+2}, \ldots, x_n) \end{array} \tag{2.37}$$

und mit der Transversalitätsbedingung aus Satz 2.15

$$\begin{array}{rclrcl} \lambda_i(t_f) & = & \nu_i, & i & = & 1, \ldots, r \\ \lambda_i(t_f) & = & g_{x_i}(x^*(t_f)), & i & = & r+1, \ldots, n. \end{array} \tag{2.38}$$

Also ist

$$\lambda_i(t_f) \quad frei, \quad i = 1, \ldots, r. \tag{2.39}$$

Insgesamt betrachtet erhält man damit die Endbedingungen

$$\begin{array}{rclrcl} x_i(t_f) & = & c_i, & i & = & 1, \ldots, r, \\ \lambda_i(t_f) & = & g_{x_i}(x^*(t_f)), & i & = & r+1, \ldots, n. \end{array} \tag{2.40}$$

Gelten also die C^k-Regularität von (2.35) und die vereinfachten Endbedingungen (2.36) so erhält man zusammen mit (2.40) das folgende Randwertproblem

$$\begin{array}{rclrcl} \begin{pmatrix} x \\ \lambda^T \end{pmatrix}^{\bullet} & = & h(t, x, \lambda), & & & \\ x(0) & = & x_0, & & & \\ x_i(t_f) & = & c_i, & i & = & 1, \ldots, r, \\ \lambda_i(t_f) & = & g_{x_i}(x(t_f)), & i & = & r+1, \ldots, n. \end{array} \tag{2.41}$$

[10]Die Funktion g wird nur im Zielfunktional an der Stelle t_f ausgewertet.

Bemerkung 2.24 *Ist die Endzeit t_f frei, so hat man zusätzlich*

$$H(t_f) = 0, \quad \dot{t_f} = 0 \tag{2.42}$$

Folgerung 2.25 *Ist eine Hamilton-Funktion C^k-regulär, dann hat das resultierende Randwertproblem eine C^k-Lösung $x(t), u(t)$ und die optimale Steuerung $u(t) = u^*(t, x(t), \lambda(t))$ ist eine C^k-Funktion.*

2. Fall: $U \subset \mathbb{R}^m$ kompakt

Ohne Einschränkung sei hier im folgenden $m = 1, \quad U = [u_{min}, u_{max}] \subset \mathbb{R}$ angenommen.
Die optimale Steuerung $u(t)$ setzt sich aus *inneren Teilstücken* $[t_1, t_2] \subset [0, t_f]$ mit $u_{min} < u(t) < u_{max}$, $t_1 \leq t \leq t_2$ und *Randstücken* $[t_1, t_2]$ mit $u(t) = u_{min}$ bzw. $u(t) = u_{max}$, $t_1 \leq t \leq t_2$ zusammen. Ein Zeitpunkt $t_1 \in (0, t_f)$ heißt *Eintrittspunkt* eines Randstückes $[t_1, t_2]$, wenn ein $\varepsilon > 0$ existiert mit $u(t) > u_{min}$ bzw. $u(t) < u_{max}$, $t_1 - \varepsilon \leq t < t_1$. Ein *Austrittspunkt* $t_2 \in (0, t_f)$ wird entsprechend definiert[11].

Für diesen Abschnitt macht man die

Voraussetzung 2.26 *Die Hamilton-Funktion H ist C^k-regulär mit $k \geq 2$, d.h. die durch*

$$u^*(t, x, \lambda) = \arg\min_{u \in U} H(t, x, \lambda, u)$$

definierte Funktion ist eine C^k-Funktion[12].

Dann besteht auf inneren Teilstücken die Beziehung

$$u(t) = u^*(t, x(t), \lambda(t)). \tag{2.43}$$

Wegen

$$u(t) = \arg\min_{u \in U} H(t, x(t), \lambda(t), u), \quad U = [u_{min}, u_{max}] \tag{2.44}$$

muß hingegen auf Randstücken gelten:

$$\begin{array}{rclcrcl} u(t) & = & u_{min}, & \Longrightarrow & H_u(t) & \geq & 0 \\ u(t) & = & u_{max}, & \Longrightarrow & H_u(t) & \leq & 0 \end{array} \tag{2.45}$$

Unter Ausnutzung der Regularität der Hamilton-Funktion zeigt man:

[11] Zur Definition der hier genannten Begriffe vergleiche auch Kapitel 2.6.
[12] Mit $k \geq 2$ ist $u^*(t, x, \lambda)$ aus $H_u(t, x, \lambda, u^*) = 0$, $H_{uu} > 0$ berechenbar.

Lemma 2.27 *Die optimale Steuerung $u(t)$ ist stetig in einem Eintritts- bzw. Austrittspunkt.*

Zusammenfassend läßt sich also aus den Ausführungen schließen, daß in beiden Fällen die Steuerung u^* für reguläre Hamiltonprobleme eindeutig bestimmbar ist und sich damit der optimale Steuerprozeß auf ein Randwertproblem zurückführen läßt, welches dann mit den Mitteln aus Kapitel 3 gelöst werden kann.

2.6 Das zustandsbeschränkte Steuerproblem

In vielen Anwendungen der optimalen Steuerprozesse treten Beschränkungen im Zustandsraum auf, welche entscheidenden Einfluß auf die notwendigen Optimalitätsbedingungen ausüben können. Es soll jedoch gezeigt werden, daß die Lösung solcher optimaler Steuerprozesse mit Zustandsbeschränkungen einer Erweiterung des Pontyaginschen Minimumprinzips genügt.
Man betrachte einen autonomen optimalen Steuerprozeß der Form:

$$\begin{array}{rcll} \min & & g(x(t_f)) + \int\limits_0^{t_f} f_0(x,u)\,dt & \\ \dot{x} & = & f(x,u), & \\ x(0) & = & x_0, & \psi(x(t_f)) = 0 \\ u(t) & \in & U, & 0 \le t \le t_f. \end{array} \tag{2.46}$$

Zusätzlich erfülle der Steuerprozeß (2.46) eine Zustandsbeschränkung der Form

$$S(x(t)) \le 0, \qquad \forall\, t \in [0, t_f] \tag{2.47}$$

wobei $S : \mathbb{R}^n \longrightarrow \mathbb{R}$ und alle anderen auftretenden Funktionen hinreichend oft stetig differenzierbar seien.

Bemerkung 2.28 *Durch $-S(x(t)) \le 0$ wird formal $S(x(t)) \ge 0$ erreicht. Setzt man sowohl $S(x(t)) \le 0$ als auch $-S(x(t)) \le 0$, so erhält man den Fall $S(x(t)) = 0$.*

Definition 2.29 (inneres Teilstück) *Ein Zeitintervall $[t_1, t_2] \subset [0, t_f]$ wird* inneres Teilstück[13]*genannt, wenn*

$$S(x(t)) < 0 \qquad \forall\, t \in [t_1, t_2] \tag{2.48}$$

Bemerkung 2.30 *Da die Zustandsbeschränkung auf solchen inneren Teilstücken inaktiv ist, gelten die notwendigen Bedingungen von Satz 2.15.*

[13] In der Literatur auch unter der Bezeichnung *freies Teilstück* zu finden.

Die Zustandsbeschränkung (2.47) kann intervallweise und punktweise aktiv werden, man unterscheidet dann in

Definition 2.31 (Randstück) *Ein Zeitintervall* $[t_1, t_2] \subset [0, t_f], \quad t_1 < t_2$ *heißt* Randstück*, wenn*

$$S(x(t)) = 0, \qquad \forall\, t \in [t_1, t_2]. \tag{2.49}$$

Hierbei heißt t_1 Eintrittspunkt[14]*, wenn*

$$S(x(t_1 - \varepsilon)) < 0 \qquad \textit{für } \varepsilon > 0 \textit{ hinreichend klein.} \tag{2.50}$$

Der Zeitpunkt t_2 *heißt* Austrittspunkt[15]*, wenn*

$$S(x(t_2 + \varepsilon)) < 0 \qquad \textit{für } \varepsilon > 0 \textit{ hinreichend klein.} \tag{2.51}$$

Definition 2.32 (Verbindungspunkt) *Ein Eintritts- bzw. Austrittspunkt heißt* Verbindungspunkt *zwischen einem inneren Teilstück und einem Randstück.*

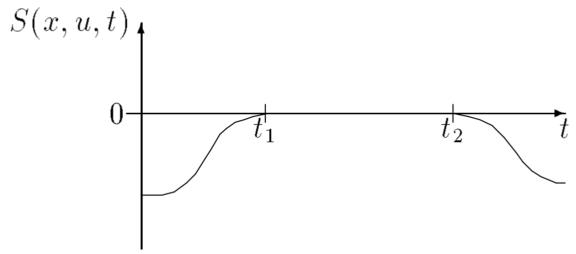


Abbildung 2.1: Beispiel eines Ein- und Austrittspunktes

Definition 2.33 (Kontaktpunkt) *Ein Zeitpunkt* $t_1 \in [0, t_f]$ *heißt* Kontaktpunkt*, wenn*

$$S(x(t_1)) = 0, \quad S(x(t_1 \pm \varepsilon)) < 0 \quad \textit{für } \varepsilon > 0 \textit{ hinreichend klein.} \tag{2.52}$$

Für $t_1 = 0$ *bzw.* $t_1 = t_f$ *gelten die Ungleichungen einseitig.*

Die zu einem Randstück $[t_1, t_2]$ gehörende Steuerung $u(t)$ heißt *Randsteuerung* und soll im folgenden genauer untersucht werden, mit der Beschränkung auf den skalaren Fall $m = 1$.

[14]In der Literatur auch unter der Bezeichnung *Aufsprungpunkt* zu finden.
[15]In der Literatur auch unter der Bezeichnung *Absprungpunkt* zu finden.

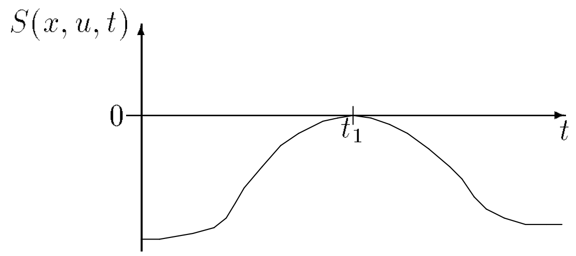


Abbildung 2.2: Beispiel eines Kontaktpunktes

Für $i \in \mathbb{N}_+$ definiert man rekursive Funktionen $S^i : \mathbb{R}^n \longrightarrow \mathbb{R}$:

$$\begin{aligned} S^0 &:= S \\ S^{i+1} &:= \left(S^i\right)_x f. \end{aligned} \tag{2.53}$$

Für die weitere Behandlung des Problems ist die Ordnung p der Zustandsbeschränkung von Bedeutung.

Definition 2.34 (Ordnung der Zustandsbeschränkung) *Sei $p \in \mathbb{N}_+$, dann hat die Zustandsbeschränkung $S(x) \leq 0$ die* Ordnung p, *wenn gilt*

$$\begin{aligned} \frac{\partial S^i}{\partial u} &= 0, \qquad i = 0, \ldots, p-1, \\ \frac{\partial S^p}{\partial u} &\neq 0. \end{aligned} \tag{2.54}$$

Nach Annahme ist S^i für $0 \leq i < p$ unabhängig von u:

$$S^i(x, u) = S^i(x), \quad \text{für } 0 \leq i < p \tag{2.55}$$

Entlang einer Trajektorie $x(t), u(t)$ mit $\dot{x} = f(x, u)$ gilt daher nach Definition von (2.53)

$$\begin{aligned} S^i(x(t)) &= \frac{d^i}{dt^i} S(x(t)), \qquad i = 0, \ldots, p-1 \\ S^p(x(t), u(t)) &= \frac{d^p}{dt^p} S(x(t)) \end{aligned} \tag{2.56}$$

p gibt also die Höhe derjenigen Ableitung von S nach t an, bei der erstmalig die Steuerung u explizit auftaucht.

Bemerkung 2.35 *Auf einem Randstück $[t_1, t_2]$ gelten daher die Beziehungen*

$$\left.\begin{aligned} S^i(x(t)) &= 0, \qquad i = 0, \ldots, p-1, \\ S^p(x(t), u(t)) &= 0 \end{aligned}\right\} \qquad \forall\, t \in [t_1, t_2] \tag{2.57}$$

Wegen des Eindeutigkeitssatzes für Differentialgleichungen sind die ersten p Beziehungen äquivalent zu den folgenden *Eintrittsbedingungen*:

$$S^i(x(t_1)) = 0 \qquad i = 0, \ldots, p-1 \tag{2.58}$$

Definition 2.36 (Berührpunkt) *Ein Punkt t_1 heißt* Berührpunkt, *wenn er Kontaktpunkt ist und darüber hinaus $S^1(x(t))$ stetig in t_1 ist. In diesem Fall ist $S^1(x(t)) = 0$, da t_1 lokales Maximum von S ist (vgl. auch Abbildung 2.2).*

Im folgenden benötigt man für die notwendigen Optimalitätsbedingungen des zustandsbeschränkten Problems die

Voraussetzung 2.37

1. *Die Gleichung $S^p(x,u) = 0$ ist eindeutig auflösbar nach $u = u(x)$ mit einer C^{p+1}-Funktion $u(x)$*
2. *Auf jedem Randstück gilt für die Randsteuerung $u(t)$:*

$$(S^p)_u (x(t), u(t)) \neq 0, \qquad t \in [t_1, t_2] \tag{2.59}$$

Aus dieser Voraussetzung folgt in Verbindung mit (2.57) die Anwendbarkeit des aus der Analysis bekannten Satzes über implizite Funktionen und damit die

Bemerkung 2.38

1. *Wegen Punkt 2 der Voraussetzung gilt auf einem Randstück $[t_1, t_2]$ die Beziehung $u(t) = u(x(t))$.*
2. *Daher genügt $x(t)$ auf $[t_1, t_2]$ der Differentialgleichung $\dot{x} = f(x, u(x))$.*
3. *Mit $u(x)$ ist dann $x(t)$ eine C^{p+1}-Funktion und damit ist auch die Randsteuerung $u(t) = u(x(t))$ eine C^{p+1}-Funktion.*

Um die notwendigen Optimalitätsbedingungen für zustandsbeschränkte Probleme formulieren zu können, bedarf es folgender Modifikation der in Definition 2.12 definierten Hamilton-Funktion.

Definition 2.39 (erweiterte Hamilton-Funktion) *Sei $\lambda_0 \in \mathbb{R}$, $\lambda : [0, t_f] \longrightarrow \mathbb{R}^n$ und $\mu : [0, t_f] \longrightarrow \mathbb{R}$.
Dann heißt*

$$\begin{aligned} \tilde{H}(x, \lambda, \mu, u) &:= \lambda_0 f_0(x, u) + \lambda f(x, u) + \mu S(x) \\ &= H(x, \lambda, u) + \mu S(x) \end{aligned} \tag{2.60}$$

die erweiterte Hamilton-Funktion *zum Steuerprozeß (2.46) und (2.47).*

Mit Hilfe dieser Definition ist man nun in der Lage, die notwendigen Optimalitätsbedingungen für einen zustandsbeschränkten optimalen Steuerprozeß zu formulieren. In Erweiterung von Satz (2.15) gilt der folgende

Satz 2.40 (erweitertes Minimumprinzip) *Es sei ein autonomer optimaler Steuerprozeß der Form (2.46) mit der Zustandsbeschränkung (2.47) gegeben. Weiter sei (x^*, u^*) eine optimale Lösung, mit welcher die Voraussetzung 2.37 erfüllt sei. Außerdem gelte für jedes Randstück $[t_1, t_2] \subset [0, t_f]$: $u^*(t) \in \mathrm{int} U$. Dann existieren*

(a) $\lambda_0 \geq 0, \quad \lambda_0 \in \mathbb{R}$ *(Konstante),*

(b) $\lambda : [0, t_f] \longrightarrow \mathbb{R}^n$, *stückweise stetig und stückweise stetig differenzierbare (adjungierte) Funktion,*

(c) $\sigma \in \mathbb{R}^r$, *(Zeilenvektor),*

(d) $\mu : [0, t_f] \longrightarrow \mathbb{R}$ *stückweise stetig (Multiplikator-Funktion),*

(e) $\nu(t_i) \geq 0, \quad \nu(t_i) \in \mathbb{R}$ *in jedem Verbindungs- oder Kontaktpunkt t_i (Multiplikatoren),*

so daß die folgenden Aussagen gelten:

1. *An allen Stetigkeitsstellen t von u^* in $[0, t_f]$ gelten:*

   (a) Minimumbedingung:

   $$\tilde{H}(x^*(t), \lambda(t), \mu(t), u^*(t)) = \min_{u \in \mathrm{U}} \tilde{H}(x^*(t), \lambda(t), \mu(t), u(t)),$$

   (b) adjungierte Differentialgleichungen und Transversalitätsbedingungen:

   $$\begin{aligned} \dot{\lambda} &= -\tilde{H}_x(x^*(t), \lambda(t), \mu(t), u^*(t)), \\ \lambda(t_f) &= \lambda_0 g_x(x^*(t_f)) + \sigma \psi_x(x^*(t_f)). \end{aligned}$$

2. *Für alle $t \in [0, t_f]$ gilt*

   $$\mu(t) \geq 0, \qquad \mu(t) S(x^*(t)) = 0.$$

3. *In jedem Verbindungs- oder Kontaktpunkt t_i gelten die* Sprungbedingungen:

   $$\lambda(t_i^+) = \lambda(t_i^-) - \nu(t_i) S_x(t_i)).$$

4. *bei freier Endzeit:*

   $$\tilde{H}(x^*(t_f), \lambda(t_f), \mu(t_f), u^*(t_f)) = 0.$$

Belegend hierzu findet man einen Beweis in [Maurer 1979].

Bemerkung 2.41 *Der Multiplikator $\mu(t)$ verschwindet auf inneren Teilstücken, wie man der zweiten Aussage entnehmen kann. Die notwendigen Optimalitätsbedingungen des unbeschränkten Problems können dadurch zurückgewonnen werden.*

Unter zusätzlichen Konvexitätsannahmen liefert auch das erweiterte Minimumprinzip 2.40 hinreichende Bedingungen für die Existenz einer optimalen Lösung. Es gilt der folgende

Satz 2.42 *Sei (x^*, u^*) ein zulässiges Paar für das Problem (2.46), (2.47), so daß die Voraussetzung 2.37 erfüllt ist. Es existiere eine Funktion $\lambda : [0, t_f] \longrightarrow \mathbb{R}^n$, $\mu : [0, t_f] \longrightarrow \mathbb{R}$ und Multiplikatoren $\nu(t_i), \sigma \in \mathbb{R}^r$, so daß die notwendigen Optimalitätsbedingungen von Satz 2.40 mit $\lambda_0 = 1$ erfüllt sind. Ferner sei*

(a) *$g(x)$ konvex,*

(b) *$\psi(x)$ affin-linear,*

(c) *$S(x)$ konvex,*

(d) *$H^0(x, \lambda) := \min\limits_{u \in U} H(x, \lambda, u)$ konvex in x für jedes $\lambda(t)$, $t \in [0, t_f]$.*

Dann ist (x^*, u^*) eine optimale Lösung.

Zum Beweis vergleiche man [Feichtinger 1986].
Für Probleme mit linear auftretender, skalarer Steuerung sei die Funktion $S^p(x, u)$ ebenfalls linear in u,

$$S^p(x, u) = A(x) + B(x)u \tag{2.61}$$

Es muß daher nach Voraussetzung 2.37 auf Randstücken $[t_1, t_2]$

$$\left(S^p\right)_u(t) = B(x(t)) \neq 0 \tag{2.62}$$

gelten, womit sich die Randsteuerung mit $S^p(x, u) = 0$ zu

$$u(x) = -\frac{A(x)}{B(x)} \tag{2.63}$$

ergibt.
Für numerische Auswertungen kann der folgende Satz aus [Maurer 1977] von Bedeutung sein.

Satz 2.43 *Sei $p = 1$ und $u(t)$ unstetig in einem Verbindungspunk t_1. Dann gilt $\nu(t_1) = 0$, also ist $\lambda(t)$ stetig in t_1 und es gilt $\Phi(t) = 0$.*

Zusätzliche Glattheitseigenschaften für die optimale Steuerung $u(t)$ können erhalten werden, wenn die Hamiltonfunktion (bzw. erweiterte Hamiltonfunktion) C^p-regulär ist, daß heißt die durch

$$u^*(x, \lambda) = \arg \min_{u \in \mathbb{R}^m} H(t, x, \lambda, u) \tag{2.64}$$

definierte Funktion ist eine eindeutig bestimmte C^p-Funktion. Zur Vereinfachung sei $m = 1, \quad U = \mathbb{R}$, es gilt der

Satz 2.44 *Sei H C^p-regulär und sei t_1 ein Verbindungs- oder Kontaktpunkt.*

1. *Für $p = 1$ ist $u(t)$ stetig in t_1 und es gilt $\nu(t_1) = 0$, d.h. $\lambda(t)$ ist stetig in t_1.*
2. *Für $p \geq 2$ sind die Ableitungen $u^{(i)}(t), \quad i = 0, \ldots, p-2$ stetig in t_1.*
3. *Ist p ungerade $p \geq 3$ und ist $u^{(p-1)}(t)$ unstetig in t_1, so ist t_1 ein Kontaktpunkt.*

Belegend hierzu vergleiche man [Jacobson 1971] und [Maurer 1976] zum Beweis.

Bemerkung 2.45 *Randstücke bzw. Berührpunkte treten nicht für alle $p \in \mathbb{N}_+$ auf. Es kann gezeigt werden, daß Randstücke und Berührpunkte in Abhängigkeit von der Ordnung von wirksamen[16] Zustandsbeschränkungen, im Falle einer regulären Hamilton-Funktion, auftreten.*
Die folgende Tabelle zeigt eine Zusammenfassung dieser Ergebnisse:

p	*Berührpunkte*	*Randstücke*
1	*nein*	*ja*
$2, 4, 6, 8, \ldots$	*ja*	*ja*
$3, 5, 7, 9, \ldots$	*ja*	*nein*

Tabelle 2.2: Das Auftreten von Randstücken und Berührpunkten in Abhängigkeit von der Ordnung p der Zustandsbeschränkung

Für Probleme mit Zustandsbeschränkungen der Ordnung $p \geq 2$ kann die folgende Bemerkung als eine Erweiterung von Satz 2.40 angesehen werden.

Bemerkung 2.46 *Es gelten die Voraussetzungen von Satz 2.40.*
Weiterhin sei $N(x(t)) := (S^{(0)}(x(t)), \ldots, S^{(p-1)}(x(t)))^T \in \mathbb{R}^p$, dann existiert für die optimale Trajektorie $x(t)$ ein $\nu = (\nu_0, \ldots, \nu_{p-1})^T \in \mathbb{R}^p$, $\nu_i \geq 0$, $i = 0, \ldots, p-1$, so daß in einem Verbindungspunkt t_1 gilt

$$\lambda^T(t_1^+) = \lambda^T(t_1^-) - \nu^T N_x(x(t_1)). \tag{2.65}$$

Für $p = 1$ spiegeln sich also die Sprungbedingungen aus Satz 2.40 wieder.

[16] D.h. die Lösungen des beschränkten und unbeschränkten Problems sind verschieden.

2.7 Weitere Beschränkungen

Als einen Spezialfall einer Zustandsbeschränkung der Ordnung 0 läßt sich die Beschränkung

$$S(x(t), u(t)) \leq 0, \qquad S_u \not\equiv 0 \tag{2.66}$$

interpretieren.
Daher gelten alle notwendigen Optimalitätsbedingungen entlang der Trajektorie mit $q = 0$ und $S^{(0)} := S$.
Im allgemeinen führen wirksame Beschränkungen der Form (2.66) nicht zu Berührpunkten. Im Vergleich zu $q \geq 1$ liegt der einzige Unterschied in der Gesetzmäßigkeit für Randstücke. Die adjungierten Differentialgleichungen sind stetig in den Verbindungspunkten t_i. Eintrittsbedingungen der Form (2.58) und damit auch der Sprungparameter ν in Satz 2.40 entfallen. Einzige Schaltbedingung im Punkte t_i ist die Stetigkeit der Hamiltonfunktion

$$H(t_i^+) = H(t_i^-) \tag{2.67}$$

Spezialfälle von (2.66) sind die Steuerbeschränkungen

$$\begin{array}{rclcl} S(x(t), u(t)) & = & u(t) - u_{max} & \leq & 0 \qquad \text{oder} \\ S(x(t), u(t)) & = & u_{min} - u(t) & \leq & 0 \end{array} \tag{2.68}$$

welche sich im Falle einer regulären Hamiltonfunktion in $U \subsetneq \mathbb{R}^m$, U kompakt wiederspiegeln.

Kapitel 3

Numerische Lösungsverfahren für Randwertprobleme

Wie schon in Kapitel 2 beschrieben, läßt sich für einen optimalen Steuerprozeß die Optimalsteuerung, unter eventueller Hinzunahme der adjungierten Differentialgleichungen, auf ein Randwertproblem zurückführen, indem die optimale Steuerung aus der Hamiltonfunktion hergeleitet und in den Steuerprozeß eingesetzt wird.
Zum numerischen Lösen von Randwertproblemen gibt es die unterschiedlichsten Verfahren. Ohne Anspruch auf Vollständigkeit seien an dieser Stelle einige genannt:
- *Einfach-Schießverfahren*[1]
- *Mehrfach-Schießverfahren*[2]
- *Kollokationsverfahren*
- *Allgemeines Newton*[3]*-Verfahren*[4]
- *Differenzenverfahren*
- *Variationsmethode*
- u.a.

In diesem Abschnitt sollen, nachdem zunächst das Randwertproblem definiert wird, die ersten drei der genannten Methoden vorgestellt werden, da diese in Spezialfällen ihre Anwendung in Kapitel 4 und Kapitel 5 finden werden.
Zunächst jedoch zur Definition des Randwertproblems.

[1] In der Literatur auch unter der Bezeichnung *Eingabelungsmethode* zu finden.
[2] In der Literatur auch unter der Bezeichnung *Mehrzielmethode* oder *multiple-shooting* zu finden.
[3] Sir Isaac Newton, geb. 1643 Whoolsthorpe (bei Grantham), gest. 1727 Kensington, England
[4] In der Literatur auch unter der Bezeichnung *Quasilinearisierung* zu finden, entsteht durch den Grenzübergang $m \to \infty$, wenn die Intervalle im Mehrfach-Schießverfahren immer feiner gewählt werden.

3.1 Das Randwertproblem

Definition 3.1 (Randwertproblem) *Sei* $t \in \mathbb{R}$, $x : \mathbb{R} \longrightarrow \mathbb{R}^n$, $f : \mathbb{R} \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$.
Wird eine Lösung des Systems

$$\dot{x} = f(t,x) = \begin{bmatrix} f_1(t, x_1, x_2, \ldots, x_n) \\ f_2(t, x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(t, x_1, x_2, \ldots, x_n) \end{bmatrix}, \qquad t_0 \leq t \leq t_f, \tag{3.1}$$

von n *gewöhnlichen Differentialgleichungen gesucht, die eine Randbedingung der Form*

$$r(x(t_0), x(t_f)) = 0 \tag{3.2}$$

mit $t_0, t_f \in \mathbb{R}$ *und* $r : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ *erfüllt, so spricht man von einem* Randwertproblem.

Anfangswertprobleme lassen sich also als spezielle Randwertprobleme auffassen. Randwertprobleme können jedoch im Gegensatz zu Anfangswertaufgaben, die in der Regel eindeutig lösbar sind (vgl. Existenz- und Eindeutigkeitssatz 7.1.1 in [Stoer 1978]), keine oder auch mehrere Lösungen besitzen. Erst unter sehr einschränkenden Bedingungen kann man auch für gewisse Randwertprobleme die eindeutige Lösbarkeit zeigen. Ein Existenz- und Eindeutigkeitssatz für solche Probleme wird zum Beispiel in [Stoer 1978], Satz 7.3.3.4 gegeben, dessen Voraussetzungen aber nur hinreichend und sehr eingeschränkt sind. Auch wenn sich einige dieser Voraussetzungen abschwächen lassen, so hat man in der Praxis in den meisten Fällen Randwertprobleme, die diese (eventuell abgeschwächten) Voraussetzungen nicht erfüllen.
Darüber hinaus sind Randwertprobleme in vielen Fällen nur numerisch lösbar.

3.2 Das Einfach-Schießverfahren

Gegeben sei das Randwertproblem

$$\begin{aligned} &\dot{x} = f(t,x) \\ &r(x(t_0), x(t_f)) = 0. \end{aligned} \tag{3.3}$$

Man bestimme einen Anfangswert $s \in \mathbb{R}^n$ für das Anfangswertproblem

$$\begin{aligned} \dot{x} &= f(t,x) \\ x(t_0) &= s, \end{aligned} \tag{3.4}$$

so daß die Lösung $x(t) = x(t;s)$ den Randbedingungen aus (3.3) genügt

$$r(x(t_0;s), x(t_f;s)) \equiv r(s, x(t_f;s)) = 0. \tag{3.5}$$

Dazu ist eine Nullstelle der Funktion

$$F(s) := r(s, x(t_f;s)) \tag{3.6}$$

zu bestimmen.
Die kann zum Beispiel mittels *Bisektionsverfahren* oder besser noch durch das *modifizierte Newtonverfahren* (vgl. [Stoer 1983]) erreicht werden, welches einen großen Konvergenzradius besitzt und daher auch noch bei schlechten Startwerten $s^{(0)}$ häufig konvergiert. Hierbei ist in jedem Iterationsschritt $x(t_f;s)$ ($s^{(i)}$ bezeichne die Näherung des Anfangswertes im i-ten Iterationsschritt, $\bar{s}$ die abschließende Lösung des Anfangswertes und $w(x;s)$ beschreibe die Trajektorie zum Anfangswert s), durch Lösen des Anfangswertproblems (3.4), zu bestimmen.
Veranschaulichend bedeutet dies im Falle bekannter Endbedingungen, daß die Differenz der durch die Anfangsschätzung bestimmten Lösung in t_f, $x(t_f;s)$ und der bekannten Endbedingungen $x(t_f)$ zu Null gerechnet wird.

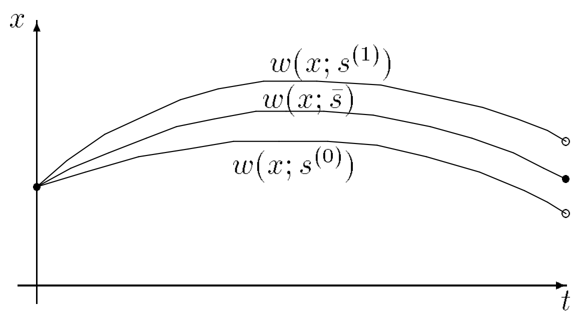


Abbildung 3.1: Graphische Übersicht eines Einfach-Schießverfahrens

Anfangswertprobleme können zum Beispiel durch

- *Einschrittverfahren* (wie das *Polygonzugverfahren von Euler*[5], *Runge-Kutta-Verfahren* oder Variationen wie das *Runge-Kutta-Fehlberg-Verfahren*),

[5]Leonhard Euler, geb. 1707 Basel, gest. 1783 Petersburg (St. Petersburg)

– *Mehrschrittverfahren* (wie das *Verfahren von Adams-Bashforth*, *Adams-Moulton-Verfahren* oder das *Nyström-Verfahren*)

– *Extrapolationsverfahren*

gelöst werden (vgl. [Stoer 1978]).
Keine der genannten Methoden weist jodoch solche Vorteile auf, daß sie den anderen Verfahren vorgezogen werden könnte.
Prinzipiell läßt sich durch das Lösen des Anfangswertproblems (3.4) zu jedem t_h aus dem Definitionsbereich ein angenäherter Wert für $x(t_h)$ bestimmen, sofern man mit dem Schießverfahren einen Lösungsvektor s ermittelt hat. In der Praxis treten jedoch beim Einfach-Schießverfahren erhebliche Ungenauigkeiten auf, wenn die Lösung $x(t) = x(t; s)$ des Anfangswertproblems sehr empfindlich von s abhängt.
Der folgende Satz zeigt, daß der Einfluß fehlerhafter Anfangsdaten exponentiell mit t wachsen kann (vgl. [Stoer 1978], Satz 7.1.4).

Satz 3.2 *Auf dem Streifen $S = \{(t,x)|\, t_0 \le t \le t_f,\ x \in \mathbb{R}^n\}$ sei die Funktion $f : S \longrightarrow \mathbb{R}^n$ stetig und genüge der* Lipschitzbedingung[6]

$$\|f(t,x_1) - f(t,x_2)\| \le L\|x_1 - x_2\| \tag{3.7}$$

für alle $(t, x_i) \in S$, $i = 1, 2$.
Sei $t_0 \le t_h \le t_f$, dann gilt für die Lösung $x(t; s)$ des Anfangswertproblems

$$\begin{aligned} \dot{x} &= f(t,x), \\ x(t_h; s) &= s \end{aligned} \tag{3.8}$$

für $t_0 \le t \le t_f$ die Abschätzung

$$\|x(t;s_1) - x(t;s_2)\| \le e^{L|t-t_h|}\|s_1 - s_2\|. \tag{3.9}$$

Der Satz zeigt aber auch, daß durch Verkleinerung des Intervalls $|t - t_0|$ der Einfluß ungenauer Anfangsdaten $s = x(t_0)$ beliebig weit zurückgeführt werden kann. Das Problem wäre beseitigt, wenn man die Werte $s_k = x(t_k)$, $k = 1, \ldots, m$ (Verwechselungen mit der Dimension m des Steuervektors $u(t)$ sind ausgeschlossen, da bei der Lösung des aus dem optimalen Steuerproblem resultierenden Randwertproblems die Steuerung eliminiert wurde), der Lösung $x(t)$ an mehreren genügend eng benachbarten Stellen t_k mit

$$t_0 = t_1 < t_2 < \ldots < t_m = t_f \tag{3.10}$$

[6] Rudolf Lipschitz, geb. 1832 Königsberg (Kaliningrad), gest. 1903 Bonn

kennen würde. Für jedes $\xi \in [t_k, t_{k+1}]$ kann man dann die Lösung $x(\xi)$ durch Lösen der Anfangswertaufgabe

$$\begin{aligned} \dot{x} &= f(t,x) \\ x(t_k) &= s_k \end{aligned} \tag{3.11}$$

auf dem kleinen Intervall $[t_k, \xi]$ mit ausreichender Genauigkeit bestimmen. Das Problem der Empfindlichkeit bezüglich der Anfangsschätzung wäre damit also beseitigt.
Diese Vorgehensweise beschreibt gleichzeitig die grundlegende Idee des multiple-shooting-Verfahrens.

3.3 Die Mehrzielmethode

Um das Problem der von s empfindlich abhängenden Lösung zu umgehen, geht man also dazu über, eine Unterteilung des Intervalls $[t_0, t_f]$ zu wählen und auf diesen Teilintervallen das Einfachschießverfahren anzuwenden.
An mehreren Stellen

$$t_0 = t_1 < t_2 \ldots < t_m = t_f \tag{3.12}$$

werden die Werte

$$\bar{s}_k = x(t_k), \quad k = 1, \ldots, m, \tag{3.13}$$

der exakten Lösung $x(t)$ des Randwertproblems (3.3) gleichzeitig iterativ berechnet. Man bezeichne dazu mit $x(t; t_k, s_k)$ die Lösung des Anfangwertproblems

$$\begin{aligned} \dot{x} &= f(t,x) \\ x(t_k) &= s_k. \end{aligned} \tag{3.14}$$

Nun sind die Vektoren s_k, $k = 1, \ldots, m$ so zu bestimmen, daß die aus den $x(t; t_k, s)$ stückweise zusammengesetzte Funktion

$$\begin{aligned} x(t) &:= x(t; t_k, s_k), \quad \text{für} \quad t \in [t_k, t_{k+1}[, \quad k = 1, \ldots, m-1, \\ x(t_f) &:= s_m \end{aligned} \tag{3.15}$$

stetig ist und und die Randbedingungen $r(x(t_0), x(t_f)) = 0$ erfüllt.

Damit erhält man die folgenden $n \times m$ Bedingungen:

$$\begin{array}{lll} x(t_{k+1}; t_k, s_k) = s_{k+1}, & k = 1, \ldots, m-1 & \text{(Stetigkeit)} \\ r(s_1, s_m) = 0 & & \text{(Randbedingungen)} \end{array} \tag{3.16}$$

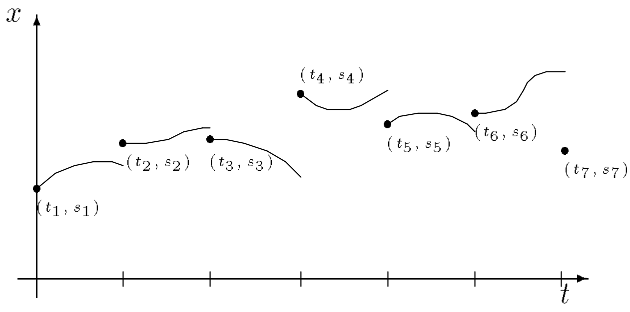


Abbildung 3.2: Graphische Übersicht eines Mehrfach-Schießverfahrens

für die $n \times m$ unbekannten Komponenten $s_k^{(j)}$, $j = 1, \ldots, n,\ k = 1, \ldots, m$ der s_k,

$$s_k = (s_k^{(1)}, s_k^{(2)}, \ldots, s_k^{(n)})^T. \tag{3.17}$$

Insgesamt erhält man somit für $s = (s_1, \ldots, s_m)^T$ aus (3.16) das Gleichungssystem

$$F(s) := \begin{bmatrix} F_1(s_1, s_2) \\ F_2(s_2, s_3) \\ \vdots \\ F_{m-1}(s_{m-1}, s_m) \\ F_m(s_1, s_m) \end{bmatrix} := \begin{bmatrix} x(t_2; t_1, s_1) - s_2 \\ x(t_3; t_2, s_2) - s_3 \\ \vdots \\ x(t_m; t_{m-1}, s_{m-1}) - s_m \\ r(s_1, s_m) \end{bmatrix} = 0, \tag{3.18}$$

welches beispielsweise wiederum mit dem modifizierten Newtonverfahren gelöst werden kann.
Dabei sind in jedem Schritt zur Berechnung von $F(s)$, $m-1$ Anfangswertprobleme zu lösen; zur Bestimmung einer Approximation $\triangle F(s)$ der Funktionalmatrix $D\,F(s)$ mittels Bildung geeigneter Differenzenquotienten müssen weitere $(m-1) \cdot n$ Anfangswertprobleme berechnet werden.
Dieser große Rechenaufwand kann jedoch erheblich reduziert werden, indem man zum Beispiel *Broyden-Approximationen* zur Bestimmung von $\triangle F(s)$ einbezieht (vgl. [Stoer 1978]). Mit der Fortran-Routine BDSCO, die in Anhang A noch näher erläutert wird, steht eine leistungsfähige multiple-shooting-Routine zur Verfügung, die diese Art der Aproximation und darüber hinaus auch andere Techniken anwendet.
Einer Lösung bedarf es somit nur noch für das Problem, eine erste Näherungslösung (Starttrajektorie) des Randwertproblems zu finden.
Da jedoch das modifizierte Newton-Verfahren in der Regel auch noch bei schlechten Startwerten konvergiert, reicht es oftmals aus, eine grobe Näherung anzugeben, die in vielen Fällen, zum Beispiel aus physikalischen Gründen, bekannt sein könnte. Sollte dies nicht ausreichend sein, so kann man oft

auf die *Homotopie-Methode*[7] zurückgreifen, bei der eine Startnäherung über ähnliche Probleme mit bekannter Lösung angestrebt wird.
Schließlich besteht noch die Möglichkeit, über die in Kapitel 4 zur Verfügung gestellten Mittel, eine Startschätzung der Trajektorie an diskreten Punkten zu erhalten.

In umfangreichen Untersuchungen hat sich die Mehrzielmethode als das praktikabelste Verfahren erwiesen, so daß diese Methode für die Behandlung von nichtlinearen Randwertproblemen für gewöhnliche Differentialgleichungen zunächst in Frage kommt.

3.4 Das Kollokationsverfahren

Einen vollkommen anderen Ansatz zur Lösung von Randwertproblemen bietet das *Kollokationsverfahren*, dessen grundlegende Idee darin besteht, ähnlich wie bei der *Spline-Interpolation*, das Intervall $[t_0, t_f]$ in Teilintervalle zu unterteilen und die Lösung des Randwertproblems auf dem Inneren jedes Teilintervalls durch Polynomfunktionen zu approximieren. Die Lösung wird also nur an diskreten Stellen, den sogenannten Gitterpunkten, genauer untersucht.

Diskretisierung
Es gibt die verschiedensten Diskretisierungsansätze für die unterschiedlichsten Problemgruppen. Das gemeinsame Ziel aller Diskretisierungsmethoden ist jedoch die Reduzierung von stetigen Systemen zu 'äquivalenten' endlichen, diskreten Systemen, welche dann einfach auf numerischen Rechnern ausgewertet werden können.
Diese Art der 'Umformulierung' von stetigen Systemen kann aber auch die verschiedensten Problematiken bezüglich der Stabilität, Konvergenz oder Genauigkeit mit sich bringen.
Ersetzt wird ein Gebiet $\mathcal{D}$ durch ein Gitter, beziehungsweise durch ein Netz von diskreten Punkten innerhalb von $\mathcal{D}$. Hierbei müssen die Gitterpunkte[8] nicht notwendig äquidistant gewählt sein, oftmals ist es sogar ratsam, die Verteilung und Dichte der Gitterpunkte den Problemanforderungen anzupassen. Statt einer Lösung, die überall in $\mathcal{D}$ definiert ist, erhält man diese nach einer Diskretisierung nur noch an den isolierten Gitterpunkten. Werte zwischen den diskreten Punkten können lediglich approximativ, wie zum Beispiel durch Interpolation, ermittelt werden.

[7]In der Literatur auch unter dem Namen *Fortsetzungsmethode* zu finden.
[8]Auch unter der Bezeichnung*Maschenpunkte* bekannt.

In den verschiedensten Bereichen der Mathematik hat die Diskretisierung

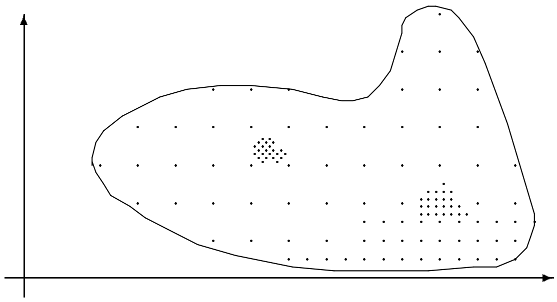


Abbildung 3.3: Beispiel eines diskreten Gebietes

und damit die Reduzierung eines unendlich dimensionalen auf ein endlich dimensionales Problem ihre Anwendung gefunden. Dieses Kapitel beschränkt sich aber auf die Diskretisierung von Randwertproblemen mittels Kollokation.

Kollokation
Gegeben sei das Randwertproblem (3.3).
Ähnlich wie beim multiple-shooting-Verfahren wird das Intervall $[t_0, t_f]$ in N Teilintervalle unterteilt:

$$[t_0, t_1[, \; [t_1, t_2[, \ldots, [t_{N-1}, t_f] \tag{3.19}$$

mit

$$t_0 < t_1 < t_2 < \ldots < t_{N-1} < t_f. \tag{3.20}$$

Wie bei der Spline-Interpolation wird nun versucht, die Lösung des Randwertproblems (3.3) auf jedem der N Teilintervalle durch Polynomfunktionen zu approximieren[9].

$$x(t) \approx x_{app}^{(i)}(t) \quad \text{für} \quad t \in [t_i, t_{i+1}[, \; i = 0, \ldots, N-1 \tag{3.21}$$

Dabei sollen die Funktionen $x_{app}^{(i)}(t)$ so gewählt werden, daß die zusammengesetzte Funktion

$$x_{app}(t) = x_{app}^{(i)}(t) \quad \text{für} \quad t \in [t_i, t_{i+1}[, \; i = 0, \ldots, N-1 \tag{3.22}$$

[9]Im folgenden soll der Index *app* für approximativ stehen.

an den Gitterpunkten t_i stetig ist.
Zusätzlich zu den existierenden Gitterpunkten werden die sogenannten *Kollokationspunkte*[10] eingeführt:

$$t_i^{(j)} := t_i + \varrho_j \cdot (t_{i+1} - t_i), \quad \varrho_j \in [0,1],\ j = 1, \ldots, l \tag{3.23}$$

Die Punkte $t_i^{(j)}$ innerhalb der Intervalle $[t_i, t_{i+1}[$ sind also durch die ϱ_j fest vorgegeben[11] und $x_{app}(t)$ erfülle an diesen Stellen die Differentialgleichung. Folgende Bedingungen müssen also demnach gelten:

$$\begin{array}{lll} \text{Stetigkeit:} & & \\ x_{app}^{(i-1)}(t_i) & = \ x_{app}^{(i)}(t_i) & \text{für } i = 1, \ldots, N-1 \\ \text{Kollokationsbedingungen:} & & \\ \dot{x}_{app}^{(i)}(t_i^{(j)}) & = \ f(t_i^{(j)}, x_{app}(t_i^{(j)})) & \text{für } i = 0, \ldots, N-1,\ j = 1, \ldots, l \end{array} \tag{3.24}$$

Außerdem muß die Bedingung

$$r(x_{app}^{(0)}(t_0), x_{app}^{(N-1)}(t_f)) = 0 \tag{3.25}$$

erfüllt sein.

Damit die Koeffizienten der Polynome $x_{app}^{(i)}$ durch (3.24) und (3.25) gerade eindeutig bestimmt sind, wählt man als Grad der Polynome die Anzahl l der Kollokationspunkte. Für die $l+1$ Koeffizienten der Polynome in jedem der N Intervalle, stehen also ebensoviele Kollokations- und Stetigkeitsbedingungen zur Verfügung.

Die Funktionen $x_{app}^{(i)}(t)$ werden bei der praktischen Durchführung als Summe ausgewählter Basispolynome $\mathcal{B}_p(t)$ dargestellt:

$$x_{app}^{(i)}(t) = \sum_p \alpha_p \cdot \mathcal{B}_p(t) \tag{3.26}$$

Dabei ist α_p ein Spaltenvektor von Gewichten.
Im einfachsten Fall könnte zum Beispiel $\mathcal{B}_p = t^p$ gewählt werden, welches jedoch nicht die günstigste Wahl darstellt.
Damit erhält man durch elementare Umformungen ein nichtlineares Gleichungssystem.
Durch geschickte Wahl der ϱ_j und α_p kann der Umfang des Problems weiter reduziert werden. Diese Eigenschaft wird in Kapitel 4 für einen Spezialfall genauer beschreiben und ausgenutzt.

[10] Kollokationspunkte und Gitterpunkte müssen nicht zwangsweise übereinstimmen.
[11] $\varrho_j = 0 \implies t_i^{(j)} = t_i, \qquad \varrho_j = 1 \implies t_i^{(j)} = t_{i+1}$

Bemerkung 3.3 *Kollokationsverfahren lassen sich in einfacher Weise mit Algorithmen zur Fehlerabschätzung verbinden, indem zu einer gegebenen Lösung eine weitere Intervallunterteilung, die den Problemanforderungen angepaßt ist, vorgenommen und die so erhaltene Lösung mit der vorherigen verglichen wird.*

Vom Prinzip her ist Kollokation das natürliche Mittel zum Lösen von Randwertproblemen, da die Randbedingungen vom Anfang der Iteration an die Parameter mitbestimmen, während beim Schießen die Randinformation erst jeweils in die nächste Iteration eingeht.
Das Kollokationsverfahren hat sich im Vergleich zum multiple-shooting-Verfahren teilweise, speziell bei langen Zeitintervallen, als robuster erwiesen, bei steifen Systemen wird Kollokation sogar als Anfangswertlöser empfohlen.

Kapitel 4

Numerische Lösung optimaler Steuerprozesse durch direkte Verfahren

Bei den direkten Verfahren verzichtet man darauf, die unter Umständen nur sehr schwer zu erhaltene Steuerung u, aus den notwendigen Optimalitätsbedingungen herzuleiten und den optimalen Steuerprozeß auf ein Randwertproblem zurückzuführen.
Bei dieser Transformation des optimalen Steuerprozesses in ein Randwertproblem werden Rand- und Optimalitätsbedingungen zur Lösungsfindung hinzugezogen. Dazu benötigt man jedoch in der Regel die adjungierten Differentialgleichungen, deren Verwendung aber zu den verschiedensten Problemen führen kann:

1. Die adjungierten Differentialgleichungen sind oft (hochgradig) nichtlinear und schwer für große Systemdynamiken zu erhalten (speziell bei Druck- und Aerodynamikproblemen).

2. Bei der Lösung des resultierenden Randwertproblems wird eine anfängliche Schätzung der adjungierten Differentialgleichungen benötigt, deren phänomenologische Bedeutung oftmals sehr schwer oder gar nicht interpretierbar ist.

3. Das multiple-shooting-Verfahren kann sehr empfindlich auf die anfängliche Schätzung reagieren, die ja, speziell nach Punkt 2, für die adjungierten Differentialgleichungen nur sehr schwer zu erhalten ist.

4. Die adjungierten Differentialgleichungen können an Ein- oder Austrittspunkten unstetig sein.

Mit der Routine BDSCO (vgl. Anhang A) steht ein leistungsstarkes multiple-shooting-Verfahren zur Verfügung, welches Punkt 4 berücksichtigt und ebenfalls die Behandlung von Schaltpunkten erlaubt[1]. Jedoch ermöglichen erst die direkten Verfahren den Verzicht auf die adjungierten Differentialgleichungen und umgehen somit die Probleme 1 - 4.

Bei den hier beschriebenen direkten Verfahren diskretisiert man den optimalen Steuerprozeß in geeigneter Weise und kombiniert ihn mit einem numerischen Algorithmus zur Lösung von nichtlinearen Optimierungsproblemen. Beschränkungen des Zustandes oder der Steuerung können ebenfalls durch einfache Diskretisierungsmechanismen mit in das nichtlineare Optimierungsprogramm aufgenommen werden.

4.1 Das Standardproblem der nichtlinearen Optimierung

Die Idee aller hier beschriebenen direkten Verfahren besteht in der Diskretisierung des optimalen Steuerprozesses und eine Rückführung auf das Standardproblem der nichtlinearen Optimierung.

Definition 4.1 (Standardproblem der nichtlinearen Optimierung)
Das nichtlineare Optimierungsproblem

$$\begin{array}{l} \textit{Minimiere} \quad f(x) \\ \textit{unter den Nebenbedingungen} \\ \qquad \begin{array}{rcll} g_i(x) & = & 0, & i = 1, \ldots, k_0 \\ g_i(x) & \leq & 0, & i = k_0 + 1, \ldots, m_0 \end{array} \end{array} \tag{4.1}$$

wird Standardproblem der nichtlinearen Optimierung genannt.

Bekanntestes Hilfsmittel zur analytischen Lösung differenzierbarer nichtlinearer Optimierungsprobleme sind die Kuhn-Tucker-Bedingungen. Jedoch können nichtlineare Optimierungsprobleme, die durch Diskretisierung eines optimalen Steuerprozesses entstehen, allein schon aufgrund ihrer Größe nur mehr numerisch gelöst werden.
Um die Vielzahl möglicher Verfahren zu verdeutlichen, seien hier ohne Anspruch auf Vollständigkeit einige Methoden aufgelistet.

1. QP-Verfahren[2],

[1] Im allgemeinen ist die Anzahl der Schaltpunkte nicht bekannt, was ebenso wie die Bestimmung einer Anfangsschätzung zu weiteren Problemen führen kann.

[2] QP steht für quadratische-Programmierung

2. SQP-Verfahren[3],

3. quadratische Approximationsverfahren,

4. Fehlerfunktionsverfahren[4],

5. Gradientenverfahren.

Die Anzahl der existierenden Optimierungsverfahren verdeutlicht gleichzeitig deren Bedeutung bezüglich möglicher Anwendungen.
Bei allen hier realisierten direkten Verfahren wurde die Routine OPRQP verwendet, die das nichtlineare Optimierungsproblem durch quadratische Approximation löst. In einer Studie ([Schittkowski 1981]) verschiedener Optimierungsverfahren, zeichnete sich die Routine OPRQP besonders durch ihre Effizienz und Zuverlässigkeit aus, obwohl sie nur lineare (in speziellen Anwendungen auch superlineare) Konvergenz besitzt. Aufgrund des Entstehungsdatums (<1975) kann jedoch vermutet werden, daß mittlerweile bessere Algorithmen (wie z.B. die SQP-Verfahren) existieren, da bereits in [Biggs 1975] die Routine OPRQP nur aus historischen Gründen aufgenommen wurde.

4.2 Direkte Verfahren durch Diskretisierung der Steuer- und Zustandsfunktionen

4.2.1 Diskretisierungsansätze nach Betts

In diesem Abschnitt sollen einige grundlegende Methoden in Kurzform beschrieben werden. Sie sollen einen ersten Eindruck zur Findung weiterer Ansätze vermitteln.
Gegeben sei das folgende optimale Steuerproblem mit fester oder freier Endzeit in *Mayer*-Form

$$\begin{array}{lrcll} \text{Minimiere} & F(x,u) & = & g(x(t_f),t_f) & \\ \text{unter} & & & & \\ & \dot{x}_i(t) & = & f_i(x,u,t) & i=1,\ldots,n \\ & R_j(x(t_0),x(t_f)) & = & 0 & j=1,\ldots,r\leq 2n \\ & S_k(x(t),u(t),t) & \leq & 0 & k=1,\ldots,\bar{s}. \end{array} \tag{4.2}$$

In Analogie zu Bemerkung 2.28 können dadurch auch Gleichheitsbeschränkungen erreicht werden.

[3]SQP steht für sequentielle-quadratische-Programmierung
[4]In der Literatur auch unter der Bezeichnung Penalty-Verfahren zu finden.

Alle Diskretisierungsansätze unterteilen das Zeitintervall $[t_0, t_f]$ in $N-1$ Segmente, mit N Gitterpunkten t_i:

$$t_0 = t_1 < t_2 < \ldots < t_{N-1} < t_N = t_f \tag{4.3}$$

Durch die Notationen $x_j := x(t_j)$, $u_j := u(t_j)$ bzw. $f_j := f(x_j, u_j, t_j)$ soll die Schreibweise im folgenden etwas vereinfacht werden. Weiterhin erfordern einige Diskretisierungsmethoden einen Wert der Steuervariablen im Intervallmittelpunkt, welcher mit $u_{j+1/2} := u(t_{j+1/2})$, $t_{j+1/2} := \frac{1}{2}(t_j + t_{j+1})$ bezeichnet werden soll. Diskretisierungsansätze werden für die drei folgenden Methoden beschrieben.

1. *Trapez-Diskretisierung*
2. *Hermite[5]-Simpson[6]-Diskretisierung*
3. *Runge-Kutta-Diskretisierung*

Jede der drei Methoden liefert einen Satz von nichtlinearen Optimierungsvariablen $\mathbf{Y}$ und Beschränkungen. Die Dynamik $\dot{x} = f(x, u, t)$ ist dabei näherungsweise erfüllt, wenn ein Fehler ζ_j, der abhängig von der Art der Diskretisierung ist und mit als Gleichheitsbeschränkung in das nichtlineare Optimierungsproblem aufgenommen wird, für $j = 1, \ldots, N-1$ und $h_j := (t_{j+1} - t_j)$ gegen 0 geht. Eine freie Endzeit kann als zusätzliche Optimierungsvariable in den Optimierungsvektor aufgenommen werden.

Trapez-Diskretisierung
Die zu optimierenden Variablen sind

$$\mathbf{Y} = [x_1, u_1, x_2, u_2, \ldots, x_N, u_N, t_N]^T. \tag{4.4}$$

Der Fehler wird festgelegt durch

$$\zeta_j = x_{j+1} - x_j - \frac{h_j}{2}(f_{j+1} + f_j). \tag{4.5}$$

Hermite-Simpson-Diskretisierung
Die Optimierungsvariablen lauten

$$\mathbf{Y} = [x_1, u_1, u_{1+1/2}, x_2, u_2, u_{2+1/2}, \ldots, x_N, u_N, t_N]^T. \tag{4.6}$$

Der Fehler wird durch

$$\zeta_j = x_{j+1} - x_j - \frac{h_j}{6}(f_{j+1} + 4f_{j+1/2} + f_j) \tag{4.7}$$

[5]Charles Hermite, geb. 1822 Dieuze, gest. 1901 Paris
[6]Thomas Simpson, geb. 1710 und gest. 1761 Market-Boshworth

definiert, wobei $f_{j+1/2} := f(x_{j+1/2}, u_{j+1/2}, t_{j+1/2})$ mit $x_{j+1/2} = \frac{1}{2}(x_{j+1} + x_j) + \frac{h_j}{8}(f_j - f_{j+1})$, $j = 1, \ldots, N-1$.

Runge-Kutta-Diskretisierung
Wie bei der Hermite-Simpson-Diskretisierung lauten die zu optimierenden Variablen

$$\mathbf{Y} = [x_1, u_1, u_{1+1/2}, x_2, u_2, u_{2+1/2}, \ldots, x_N, u_N, t_N]^T. \tag{4.8}$$

Durch

$$\zeta_j = x_{j+1} - k_j^{(4)} \tag{4.9}$$

mit

$$\begin{aligned} k_j^{(1)} &= x_j + \frac{h_j}{2} f(x_j, u_j, t_j) \\ k_j^{(2)} &= x_j + \frac{h_j}{2} f(k_j^{(1)}, u_{j+1/2}, t_{j+1/2}) \\ k_j^{(3)} &= x_j + h_j f(k_j^{(2)}, u_{j+1/2}, t_{j+1/2}) \\ k_j^{(4)} &= x_j + \frac{h_j}{6}[f(x_j, u_j, t_j) + 2f(k_j^{(1)}, u_{j+1/2}, t_{j+1/2}) \\ &\quad +2f(k_j^{(2)}, u_{j+1/2}, t_{j+1/2}) + f(k_j^{(3)}, u_{j+1}, t_{j+1})] \\ &\quad j = 1, \ldots, N-1 \end{aligned} \tag{4.10}$$

wird hier der Fehler festgelegt.

Die definierten Fehler werden, ebenso wie die Anfangs- und Endbedingungen und die allgemeinen Beschränkungen des optimalen Steuerprozesses, in das nichtlineare Optimierungsproblem aufgenommen, so daß z.B. für die Trapez-Diskretisierung das nachfolgende nichtlineare Optimierungsproblem vorliegt

$$\begin{aligned} &\min F(\mathbf{Y}), \quad F : \mathbb{R}^{N(n+m)+1} \longrightarrow \mathbb{R} \\ &\text{bezogen auf} \\ &\qquad \zeta_j = 0, \quad j = 1, \ldots, N-1 \\ &R_i(x(t_1), x(t_N), t_N) = 0 \quad i = 1, \ldots, r \\ &S_k(x(t_l), u(t_l), t_l) \leq 0 \quad k = 1, \ldots, \bar{s}, \quad l = 1, \ldots, N \end{aligned} \tag{4.11}$$

Das Differentialgleichungssystem sowie die Randbedingungen werden dadurch näherungsweise erfüllt. Die allgemeinen Beschränkungen sind an den Gitterpunkten erfüllt.
Man beachte, daß für die Hermite-Simpson- bzw. Runge-Kutta-Diskretisierung zusätzliche Beschränkungen an den Intervallmittelpunkten erfüllt sein müssen.

Mit der Routine DIROPTIM liegt ein von mir umgesetztes Fortran-Programm dieser Diskretisierungsidee mit dem Ansatz über die Trapezmethode vor.

4.2.2 Spezielle Eigenschaften der beschriebenen Diskretisierungen

Die aus den Diskretisierungen (Trapez, Hermite-Simpson und Runge-Kutta) resultierenden nichtlinearen Optimierungsprobleme sind charakterisiert durch große, aber schwach besetzte Matrizen[7]. Um eine effizientere Abarbeitung des nichtlinearen Optimierungsproblems erreichen zu können, erscheint es sinnvoll, diese speziellen Eigenschaften beim Entwurf der Optimierungsroutinen zu berücksichtigen. Motiviert wird dieser Ansatz weiterhin dadurch, daß ein Großteil der zur Optimierung benutzten Rechenzeit für die Bestimmung des Gradienten, der *Hesse*[8]- und *Jacobi*[9]-Matrix verwendet wird.
Aus Gründen der Vereinfachung sei für diesen Abschnitt die folgende Voraussetzung gültig.

Voraussetzung 4.2

1. *Die Endzeit t_f sei fest.*
2. *Die Anzahl der Nebenbedingungen sei $\bar{s} = 0$.*
3. *Es liegen r Standardrandbedingungen vor, davon r' Endbedingungen, mit $r' \leq n$.*

Demnach wird nur ein unbeschränkter Steuerprozeß untersucht. Die Ergebnisse können jedoch ohne große Probleme auf allgemeine, beschränkte optimale Steuerprozesse übertragen werden. Bei Anwendungen mit Beschränkungen erweitert sich die Jacobi-Matrix entsprechend den Nebenbedingungen, ist aber auch weiterhin durch eine gewisse Blockstruktur (s.u.) gekennzeichnet, so daß sich auch diese Charakteristika leicht in das nichtlineare Optimierungsproblem übernehmen lassen.

---

[7] Die großen und schwachbesetzten Matrizen sind eine natürliche Folge der Diskretisierung der Zustands- und Steuerfunktionen. Dadurch wird vermieden, daß bei der Ableitungsbestimmung die Optimierungsvariablen Einfluß auf Teilintervalle haben, in denen sie nicht direkt auftreten.

[8] Ludwig Otto Hesse, geb. 1811 Königsberg (Kaliningrad), gest. 1874 München

[9] Carl Gustav Jacob, geb. 1804 Potsdam, gest. 1851 Berlin

Für alle drei Diskretisierungsmethoden ergibt sich der Gradient des Zielfunktionals zu

$$\frac{\partial g}{\partial \mathbf{Y}} = \Big(\underbrace{0, \ldots, 0,}_{\substack{\text{(n+m)(N-1)} \\ \text{bzw. (n+2m)(N-1)}}} \underbrace{\frac{\partial g}{\partial x_N},}_{\text{n}} \underbrace{0}_{\substack{\text{m} \\ \text{bzw. 2m}}} \Big) \tag{4.12}$$

Die Jacobi-Matrix der dynamischen Nebenbedingungen ($\zeta_j = 0$, $j = 1, \ldots, N-1$) wird in einer schematischen Übersicht in Abbildung 4.1 wiedergegeben.

$$\begin{pmatrix} \boxed{A_1} & & & & \\ & \boxed{A_2} & & \mathbf{0} & \\ & & \boxed{A_3} & & \\ & \mathbf{0} & & \ddots & \\ & & & & \boxed{A_{N-1}} \end{pmatrix} \tag{4.13}$$

Abbildung 4.1: Spezielle Blockstruktur der Jacobi-Matrix der dynamischen Nebenbedingungen bei den beschriebenen Verfahren

Bei der Trapez-Diskretisierung haben die Teilblöcke A_i, $i = 1, \ldots, N-1$ die Dimension $n \times 2(n+m)$ und ergeben sich im einzelnen für $i = 1, \ldots, N-1$ zu

$$A_i = \begin{pmatrix} \frac{\partial \zeta_i^{(1)}}{\partial x_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(1)}}{\partial x_i^{(n)}} & \frac{\partial \zeta_i^{(1)}}{\partial u_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(1)}}{\partial u_i^{(m)}} & \frac{\partial \zeta_i^{(1)}}{\partial x_{i+1}^{(1)}} & \cdots & \frac{\partial \zeta_i^{(1)}}{\partial x_{i+1}^{(n)}} & \frac{\partial \zeta_i^{(1)}}{\partial u_{i+1}^{(1)}} & \cdots & \frac{\partial \zeta_i^{(1)}}{\partial u_{i+1}^{(m)}} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial \zeta_i^{(n)}}{\partial x_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(n)}}{\partial x_i^{(n)}} & \frac{\partial \zeta_i^{(n)}}{\partial u_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(n)}}{\partial u_i^{(m)}} & \frac{\partial \zeta_i^{(n)}}{\partial x_{i+1}^{(1)}} & \cdots & \frac{\partial \zeta_i^{(n)}}{\partial x_{i+1}^{(n)}} & \frac{\partial \zeta_i^{(n)}}{\partial u_{i+1}^{(1)}} & \cdots & \frac{\partial \zeta_i^{(n)}}{\partial u_{i+1}^{(m)}} \end{pmatrix} \tag{4.14}$$

Für die beiden anderen Diskretisierungen ergeben sich in analoger Weise die Teilblöcke jedoch in Abhängigkeit unterschiedlicher Parameter und Dimensionen für $i = 1, \ldots, N-1$:

Hermite-Simpson-Diskretisierung :

Dimension $n \times (2n + 3m)$, $A_i := A_i(x_i, x_{i+1}, u_i, u_{i+1}, u_{i+1/2})$

Runge-Kutta-Diskretisierung :

Dimension $n \times 2(n+m)$, $A_i := A_i(x_i, x_{i+1}, u_i, u_{i+1/2})$.

Werden die Werte $u_{i+1/2}$, $i = 1, \ldots, N-1$ bei den letzten beiden Diskretisierungen nicht durch zusätzliche Optimierungsvariablen, sondern durch eine Interpolation ermittelt, so ändert sich auch die Struktur der Teilblöcke.

Bei der Trapez- und Runge-Kutta-Diskretisierung lassen sich weitere Reduzierungen bezüglich Speicher- und Rechenaufwand erreichen, wenn man bei der Berechnung der Ableitungen für die dynamischen Beschränkungen ζ_i Informationen der bereits berechneten Beschränkungen ζ_{i-1} beziehungsweise die Trivialität gewisser Ableitungen ausnutzt. Für die Trapez-Diskretisierung gilt zum Beispiel :

$$\begin{aligned} &\frac{\partial \zeta_{i-1}}{\partial x_i} = 1 - \frac{h_{i-1}}{2}\frac{\partial f_i}{\partial x_i}, \quad \text{bzw.} \quad \frac{\partial \zeta_{i-1}}{\partial u_i} = -\frac{h_{i-1}}{2}\frac{\partial f_i}{\partial u_i}. \\ \text{und} \\ &\frac{\partial \zeta_i}{\partial x_i} = -1 - \frac{h_i}{2}\frac{\partial f_i}{\partial x_i}, \quad \text{bzw.} \quad \frac{\partial \zeta_i}{\partial u_i} = -\frac{h_i}{2}\frac{\partial f_i}{\partial u_i}. \end{aligned} \tag{4.15}$$

Während bei der Runge-Kutta-Diskretisierung

$$\frac{\partial \zeta_i}{\partial x_{i+1}} = 1 \tag{4.16}$$

gilt. Für die Hermite-Simpson-Diskretisierung ist eine ähnliche Reduzierung nicht möglich.
Die hier für die Jacobi-Matrix beschriebenen Verbesserungen lassen sich in analoger Weise auch auf die Hesse-Matrix übertragen.
Die Hermite-Simpson-Diskretisierung erzeugt also eine dichter besetzte Jacobi- und Hesse-Matrix.

Sollen qualitative Aussagen über die Effizienz eines Verfahrens gemacht werden, so reicht es in der Regel sicherlich nicht aus, die Besetzung der verschiedenen Ableitungs-Matrizen zu untersuchen.
Ein weiteres Maß für die Güte eines Verfahrens ist die Anzahl der Auswertungen der rechten Seite des Differentialgleichungssystems pro Iterationsschritt. Hier zeigen Trapez- und Hermite-Simpson-Diskretisierung deutliche Vorteile gegenüber der Runge-Kutta-Diskretisierung, da sich ähnlich wie bei der Auswertung der Jacobi- und Hesse-Matrix der Aufwand durch Ausnutzen bereits vorhandener Informationen weiter reduzieren läßt. Aber auch die Konvergenzordnung hat einen entscheidenden Einfluß bei der Bewertung der verschiedenen Diskretisierungsverfahren.
Eine wichtige Rolle spielt ebenfalls die Zuverlässigkeit eines Verfahrens. Gerade Verfahren größerer Konvergenzordnung weisen hier einen entscheidenden

Nachteil gegenüber Verfahren geringerer Ordnung, wie die Trapez-Methode, auf. Beide, sowohl das Hermite-Simpson-, als auch der Runge-Kutta-Ansatz, machen während ihrer Abarbeitung eine nichtlineare 'Vorhersage' für die Zustandsvariablen in den Intervallmittelpunkten, das Runge-Kutta-Verfahren sogar für die Endpunkte eines jeden Intervalls. Da bei den hier vorgestellten Verfahren diese Werte keine Optimierungsvariablen sind, unterliegen sie auch nicht den Beschränkungen des Optimierungsproblems. Im ungünstigsten Fall kann es sogar dazu kommen, daß diese 'Vorhersagen' der Zustandsvariablen nicht einmal berechenbar sind, weil die Grenzen der Berechenbarkeit des Computers überschritten werden, obwohl eine Auswertung an den Gitterpunkten durchaus möglich wäre.
Eine typische Situation wird in Abbildung 4.2 beschrieben.

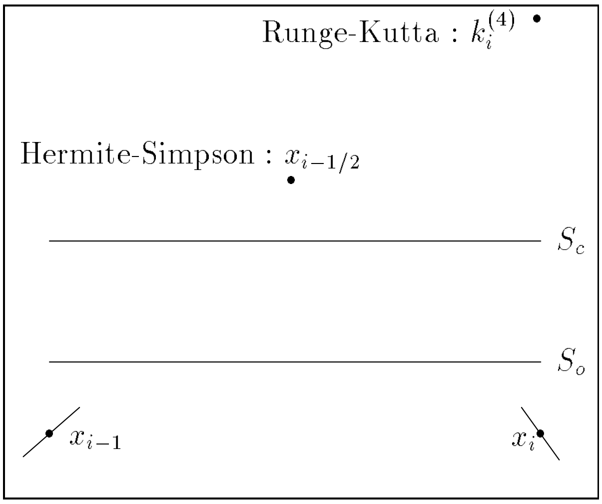


Abbildung 4.2: Reduktion der Zuverlässigkeit bei Verfahren höherer Ordnung

Dabei bedeuten :

S_c	Grenzen der Berechenbarkeit des Computers
S_o	Beschränkungen des Optimierungsproblems
x_{i-1}	Zustand am Gitterpunkt t_{i-1}
x_i	Zustand am Gitterpunkt t_i
$x_{i-1/2}$	'Vorhersage' des Zustandes im Punkt $t_{i-1/2}$
$k_i^{(4)}$	'Vorhersage' des Zustandes am Gitterpunkt t_i

Eine Zusammenfassung dieser Ergebnisse für den unbeschränkten Fall, bei voller Ausnutzung der Reduzierungsmöglichkeiten, wird in Tabelle 4.1 gegeben.
Es bedeuten :

J_0 Jacobi-Matrix Elemente $\neq 0$

	Trapez	Herm.	RK-4
J_0	$(N-1)n(n+m)$	$2(N-1)n(n+m)$	$(N-1)n(n+m)$
H_0	$N(n+m)^2/2$	$2N(n+m)^2$	$N(n+m)^2/2$
$Error$	$O(h^2)$	$O(h^4)$	$O(h^4)$
RSA	N	$2N-1$	$4N-4$
$Zuver.$	gut	schlecht	schlecht

Tabelle 4.1: Vergleich von Trapez-, Hermite-Simpson- und Runge-Kutta-Diskretisierung bezüglich ihrer Transformation in ein nichtlineares Optimierungsproblem

H_0	Hesse-Matrix Elemente $\neq 0$
$Error$	Fehler- und Konvergenzordnung
RSA	Auswertungen der rechten Seite des DGL-Systems pro Iteration
$Zuver.$	Zuverlässigkeit

4.2.3 Lösung eines optimalen Steuerprozesses durch direkte Kollokation

Es soll nun die Idee der in Kapitel 3.4 beschriebenen Kollokation aufgenommen werden, um zwei weitere Ansätze für direkte Verfahren aufzuzeigen.
Die hier beschriebenen Umsetzungen haben sich in der praktischen Anwendung als sehr zuverlässig und effizient erwiesen und wurden schon um viele praktische 'Zusätze', wie für Konvergenzuntersuchungen oder automatische Gitterpunktswahl, erweitert.

Bei beiden Ansätzen diskretisiert man, ausgehend vom optimalen Steuerprozess (4.2), das Zeitintervall $[t_0, t_f]$

$$t_0 = t_1 < t_2 < \ldots < t_N = t_f \tag{4.17}$$

Ansatz 1: Lineare Approximation der Steuerung

Die Optimierungsvariable $\mathbf{Y}$ des nichtlinearen Optimierungsproblems besteht dann aus den Werten der Steuerung u, des Zustandes x und der Endzeit t_f an den Gitterpunkten $t_i, \quad i = 1, \ldots, N$

$$\mathbf{Y} = [x(t_1), u(t_1), \ldots, x(t_N), u(t_N), t_N] \in \mathbb{R}^{N(n+m)+1}. \tag{4.18}$$

Für $t_i \leq t \leq t_{i+1}$ wird die Steuerung als stückweise linear interpolierte Funktion zwischen $u(t_i)$ und $u(t_{i+1})$ gewählt:

$$u_{app}(t) = u(t_i) + (t - t_i)\frac{u(t_{i+1}) - u(t_i)}{t_{i+1} - t_i} \tag{4.19}$$

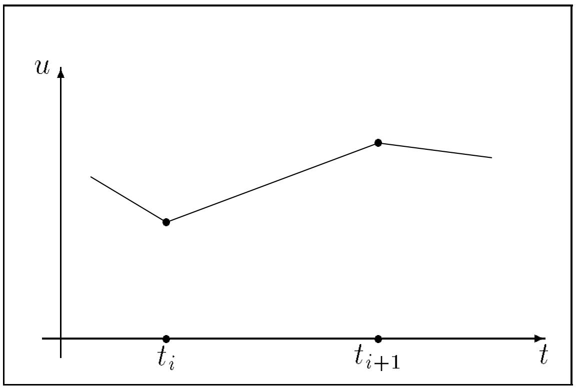


Abbildung 4.3: Lineare Interpolation der Steuerung

Der Zustand $x(t)$ wird nun zwischen den Gitterpunkten durch die stetig differenzierbare und stückweise kubische Funktion $x_{app}(t), \quad t_i \leq t \leq t_{i+1}$ zwischen $x(t_i)$ und $x(t_{i+1})$ interpoliert (Hermite-Interpolation).
Damit die Koeffizienten der Polynome eindeutig bestimmt sind, wird mit den Bezeichnungen aus Kapitel 3.4 der Grad der Polynome zu $l = 3$ und genauer die $\varrho_j \in [0, 1], \quad j = 1, \ldots, l$ zu

$$\begin{aligned} \varrho_1 &= 0 \\ \varrho_2 &= \tfrac{1}{2} \\ \varrho_3 &= 1 \end{aligned} \tag{4.20}$$

gewählt.
Durch diese Wahl von ϱ_1 bzw. ϱ_3 fallen zwei der geforderten drei Kollokationsbedingungen je Intervall heraus, da

$$\dot{x}_{app}(\hat{s}) := f(x(\hat{s}), u(\hat{s}), \hat{s}), \quad \hat{s} = t_i, t_{i+1}, \quad i = 1, \ldots, N-1 \tag{4.21}$$

Als Basispolynome wählt man

$$\mathcal{B}_q^{(i)} := \left(\frac{t - t_i}{t_{i+1} - t_i}\right)^q, \quad q = 0, \ldots, l, \quad i = 1, \ldots, N-1 \tag{4.22}$$

Zusammenfassend werden für $h_i := t_{i+1} - t_i$ und $l = 3$ die Zustände durch das zusammengesetzte Polynom

$$x_{app}(t) = \sum_{q=0}^{l} c_q^{(i)} \mathcal{B}_q^{(i)} = \sum_{q=0}^{3} c_q^{(i)} \left(\frac{t - t_i}{h_i}\right)^q, \ t \in [t_1, t_{i+1}], \ i = 1, \ldots, N-1 \tag{4.23}$$

interpoliert.
Durch Ausnutzen der Stetigkeit und stetigen Differenzierbarkeit an den Git-

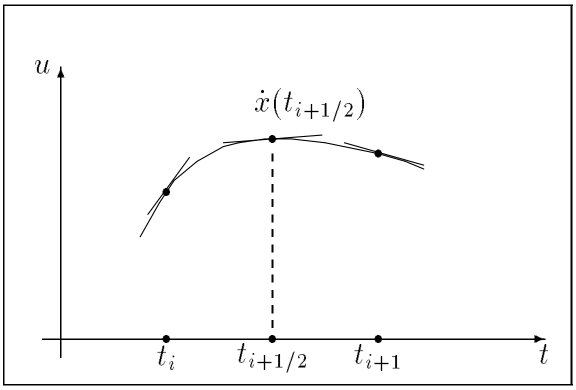


Abbildung 4.4: Kubische Interpolation des Zustandes

terpunkten $t_i, \quad i = 1, \ldots, N-1$ ergeben sich die Koeffizienten $c_q^{(i)}$ zu:

$$\begin{aligned} c_0^{(i)} &= x(t_i) \\ c_1^{(i)} &= h_i f_i \\ c_2^{(i)} &= 3[x(t_{i+1}) - x(t_i)] - h_i[f_{i+1} + 2f_i] \\ c_3^{(i)} &= -2[x(t_{i+1}) - x(t_i)] + h_i[f_{i+1} + f_i] \end{aligned} \tag{4.24}$$

In Analogie zu Kapitel 4.2.1 wird ein Fehler

$$\zeta_{i+1/2} := f(x_{app}(t_{i+1/2}), u_{app}(t_{i+1/2}), t_{i+1/2}) - \dot{x}_{app}(t_{i+1/2}), \quad i = 1, \ldots, N-1 \tag{4.25}$$

definiert.
Die Beschränkungen des nichtlinearen Optimierungsproblems ergeben sich damit zu

- Beschränkungen des optimalen Steuerprozesses an den Gitterpunkten t_i:
$$S(x(t_i), u(t_i), t_i) \leq 0, \quad i = 1, \ldots, N$$

- Randbedingungen:
$$R(x(t_0), x(t_f), t_f) = R(x(t_1), x(t_N), t_N) = 0$$

- Kollokationsbedingungen für $\varrho_2 = 1/2$, die durch den Fehler $\zeta_{i+1/2}$ beschrieben werden:
$$\zeta_{i+1/2} = 0, \quad i = 1, \ldots, N-1$$

Im Gegensatz zu ähnlichen Diskretisierungsverfahren, hat diese Art der Diskretisierung des Systemzustandes $x(t)$ zweierlei Vorteile:

1. Die Anzahl der Kollokationsbedingungen im nichtlinearen Optimierungsproblem reduzieren sich entsprechend (4.21) von drei auf eins.
2. Die Anzahl der freien Koeffizienten für die kubischen Polynome in jedem der $N-1$ Intervalle halbiert sich auf zwei.

Zusammenfassend ergibt sich damit das folgende nichtlineare Optimierungsproblem:

$$
\begin{array}{ll}
\text{min} & F(\mathbf{Y}), \quad F : \mathrm{I\!R}^{N(n+m)+1} \longrightarrow \mathrm{I\!R} \\
\text{mit} & \mathbf{Y} = (x(t_1), u(t_1), \ldots, x(t_N), u(t_N), t_N) \\
\text{bezogen auf} & \\
& \begin{array}{rcl}
f(x_{app}(t_{1+1/2}), u_{app}(t_{1+1/2}), t_{1+1/2}) - \dot{x}_{app}(t_{1+1/2}) & = & 0 \\
\vdots & \vdots & \vdots \\
f(x_{app}(t_{N-1/2}), u_{app}(t_{N-1/2}), t_{N-1/2}) - \dot{x}_{app}(t_{N-1/2}) & = & 0 \\
R(x(t_1), x(t_N), t_N) & = & 0
\end{array} \\
\text{und} & \\
& \begin{array}{rcl}
S(x(t_1), u(t_1), t_1) & \leq & 0 \\
\vdots & \vdots & \vdots \\
S(x(t_N), u(t_N), t_N) & \leq & 0
\end{array}
\end{array}
\tag{4.26}
$$

Bei erfolgreich abgeschlossener Diskretisierung kann das resultierende nichtlineare Optimierungsproblem auf numerischem Wege gelöst werden.

Mit der Fortran-Routine KOLLOKAT liegt eine von mir umgesetzte, effiziente Realisierung dieser Diskretisierungsidee vor.
Die Diskretisierung wird automatisch vorgenommen und die Optimierungsvariablen, sowie die benötigten Beschränkungen, werden an das nichtlineare Optimierungsprogramm weitergegeben.
Grundlage ist die Fortran-Routine OPRQP, die das resultierende Optimierungsproblem durch approximative quadratische Programmierung löst. Einige Modifikationen dieser Routine waren jedoch wegen der sich verändernden Problemanforderungen (erheblicher Speicherbedarf) notwendig.

Das hier beschriebene direkte Verfahren mittels Kollokation wurde schon erfolgreich auf die verschiedensten optimalen Steuerprozesse angewendet.
Der Konvergenzradius ist sehr groß, Konvergenz wurde sogar schon in Fällen, in denen keinerlei Information über die optimale Trajektorie bekannt war, erreicht.

Ansatz 2: Stückweise kubische Hermite-Approximation der Steuerung

Im Gegensatz zum 1. Ansatz soll nun die Möglichkeit einer besseren Approximation der Steuerung beschrieben werden, mit dem Ziel den in die Steuerung eingehenden Approximationsfehler konform mit dem des Zustandes zu gestalten. Dabei ist ein Ansatz über kubische Splines nicht empfehlenswert, weil dadurch die spezielle Blockstruktur der Jacobi- und Hesse-Matrix (vgl. Kapitel 4.2.4) zerstört wird.
Es wird daher die stückweise kubische Hermite-Interpolation gewählt, die allerdings den Nachteil eines größeren Optimierungsvektors, durch die benötigte Ableitung der Steuerung an den Gitterpunkten, nach sich zieht.
Die Optimierungsvariable $\mathbf{Y}$ des nichtlinearen Optimierungsproblems besteht dann aus den Werten der Steuerung u, der Ableitung der Steuerung $\dot{u}$, des Zustandes x und eventuell der Endzeit t_f an den Gitterpunkten t_i, $\quad i = 1, \ldots, N$:

$$\mathbf{Y} = [x(t_1), u(t_1), \dot{u}(t_1), \ldots, x(t_N), u(t_N), \dot{u}(t_N), t_N] \in \mathrm{I\!R}^{N(n+2m)+1} \tag{4.27}$$

Die Anzahl der aus der Steuerung resultierenden Optimierungsvariablen hat sich somit verdoppelt.

Auf jedem Teilintervall $[t_i, t_{i+1}]$ werden nun $u(t)$ und $x(t)$ durch stückweise kubische Hermite-Polynome $u_{app}(t)$ und $x_{app}(t)$ interpoliert. Dabei kann man in völliger Analogie zu Ansatz 1 vorgehen und erhält neben (4.23) und (4.24), mit (4.22) zur Beschreibung der Basispolynome, für u_{app} das zusammengesetzte Polynom

$$u_{app}(t) = \sum_{q=0}^{3} d_q^{(i)} \mathcal{B}_q^{(i)} = \sum_{q=0}^{3} d_q^{(i)} \left(\frac{t - t_i}{h_i}\right)^q, \quad i = 1, \ldots, N-1. \tag{4.28}$$

mit den eindeutig bestimmten Koeffizienten $d_q^{(i)}$:

$$\begin{aligned} d_0^{(i)} &= u(t_i) \\ d_1^{(i)} &= h_i \dot{u}(t_i) \\ d_2^{(i)} &= 3[u(t_{i+1}) - u(t_i)] - h_i[\dot{u}_{i+1} + 2\dot{u}_i] \\ d_3^{(i)} &= -2[u(t_{i+1}) - u(t_i)] + h_i[\dot{u}_{i+1} + \dot{u}_i]. \end{aligned} \tag{4.29}$$

Auf diese Weise wird eine stetig differenzierbare Approximation der Steuerung auf dem ganzen Optimierungsintervall erreicht.
Die Beschränkungen des resultierenden nichtlinearen Optimierungsproblems ergeben sich damit wie im Ansatz 1, wobei bei der Definition des Fehlers

$\zeta_{i+1/2}$ in (4.25) der nun neu ermittelte Wert für $u_{app}(t_{i+1/2})$ eingesetzt wird. Die in Ansatz 1 beschriebenen Vorteile bleiben somit erhalten und zusammenfassend ergibt sich das nichtlineare Optimierungsproblem (4.26) mit dem erweiterten Optimierungsvektor (4.27).

Bemerkung 4.3 *Ansatz 2 empfiehlt sich insbesonders, wenn explizite Beschränkungen an die Steuergeschwindigkeit $\dot{u}$ gestellt werden.*

4.2.4 Spezielle Eigenschaften der direkten Kollokation

Wie schon die in Kapitel 4.2.1 beschriebenen Diskretisierungsansätze zeichnet sich auch die direkte Kollokation durch große, aber schwach besetzte Matrizen, sowie durch die Verfügbarkeit analytischer Gradienten aus.
Einschränkend wird auch hier von Voraussetzung 4.2 ausgegangen, wobei die Erweiterung auf beschränkte Steuerprobleme mit $n > 1,\ m > 1$ ebenfalls unproblematisch ist.
Der Gradient der Zielfunktion ergibt sich offensichtlich für beide beschriebenen Verfahren zu

$$\frac{\partial g}{\partial \mathbf{Y}} = \Big(\underbrace{0, \ldots, 0,}_{\substack{\text{(n+m)(N-1)} \\ \text{bzw. (n+2m)(N-1)}}} \underbrace{\frac{\partial g}{\partial x_N},}_{\text{n}} \underbrace{0}_{\substack{\text{m} \\ \text{bzw. 2m}}} \Big) \tag{4.30}$$

Die Jacobi-Matrix der dynamischen Nebenbedingungen (Kollokationsbedingungen)

$$\zeta_{i+1/2} = 0, \qquad i = 1, \ldots, N-1 \tag{4.31}$$

hat die in Abbildung 4.5 veranschaulichte Bidiagonalblockstruktur, dessen schwachbesetzte Struktur sich in der Hesse-Matrix der Zielfunktion wiederholt.
Für die beschriebenen Ansätze zur direkten Kollokation besitzen die Teilblöcke $A_{i,j}$ die Dimension $(n \times (n+m))$ bzw. $(n \times (n+2m))$ und ergeben sich im ersten Ansatz zu

$$A_{i,i} = \begin{pmatrix} \frac{\partial \zeta^{(1)}_{i-1/2}}{\partial x^{(1)}_i} & \cdots & \frac{\partial \zeta^{(1)}_{i-1/2}}{\partial x^{(n)}_i} & \frac{\partial \zeta^{(1)}_{i-1/2}}{\partial u^{(1)}_i} & \cdots & \frac{\partial \zeta^{(1)}_{i-1/2}}{\partial u^{(m)}_i} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \zeta^{(n)}_{i-1/2}}{\partial x^{(1)}_i} & \cdots & \frac{\partial \zeta^{(n)}_{i-1/2}}{\partial x^{(n)}_i} & \frac{\partial \zeta^{(n)}_{i-1/2}}{\partial u^{(1)}_i} & \cdots & \frac{\partial \zeta^{(n)}_{i-1/2}}{\partial u^{(m)}_i} \end{pmatrix} \qquad i = 2, \ldots, N-1 \tag{4.33}$$

$$\begin{pmatrix} \boxed{A_{2,2}} & \boxed{A_{2,3}} & & & & \\ & \boxed{A_{3,3}} & \boxed{A_{3,4}} & & \mathbf{0} & \\ & & \boxed{A_{4,4}} & \boxed{A_{4,5}} & & \\ & \mathbf{0} & & \ddots & & \\ & & & & \boxed{A_{N-1,N-1}} & \boxed{A_{N-1,N}} \end{pmatrix} \tag{4.32}$$

Abbildung 4.5: Spezielle Bidiagonalblockstruktur der Jacobi-Matrix der dynamischen Nebenbedingungen bezogen auf die beschriebenen Verfahren

und

$$A_{i,i+1} = \begin{pmatrix} \frac{\partial \zeta^{(1)}_{i+1/2}}{\partial x^{(1)}_{i+1}} & \dots & \frac{\partial \zeta^{(1)}_{i+1/2}}{\partial x^{(n)}_{i+1}} & \frac{\partial \zeta^{(1)}_{i+1/2}}{\partial u^{(1)}_{i+1}} & \dots & \frac{\partial \zeta^{(1)}_{i+1/2}}{\partial u^{(m)}_{i+1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \zeta^{(n)}_{i+1/2}}{\partial x^{(1)}_{i+1}} & \dots & \frac{\partial \zeta^{(n)}_{i+1/2}}{\partial x^{(n)}_{i+1}} & \frac{\partial \zeta^{(n)}_{i+1/2}}{\partial u^{(1)}_{i+1}} & \dots & \frac{\partial \zeta^{(n)}_{i+1/2}}{\partial u^{(m)}_{i+1}} \end{pmatrix} \qquad i = 2, \dots, N-1. \tag{4.34}$$

Im zweiten Ansatz erweitern sich die Matrizen um die entsprechenden Ableitungen nach den $\dot{u}(t_i)$, $i = 1, \dots, N$.
Aufgrund des einfachen Ansatzes der stückweise kubischen Hermite-Interpolation lassen sich die Ableitungen für $j = 1, \dots, n, \quad i = 1, \dots, N-1$

$$\frac{\partial \zeta^{(j)}_{i+1/2}}{\partial x_i}, \quad \frac{\partial \zeta^{(j)}_{i+1/2}}{\partial x_{i+1}}, \quad \frac{\partial \zeta^{(j)}_{i+1/2}}{\partial u_i}, \quad \frac{\partial \zeta^{(j)}_{i+1/2}}{\partial u_{i+1}}, \quad \frac{\partial \zeta^{(j)}_{i+1/2}}{\partial \dot{u}_i}, \quad \frac{\partial \zeta^{(j)}_{i+1/2}}{\partial \dot{u}_{i+1}}, \tag{4.35}$$

analytisch noch weiter vereinfachen.
Die schwachbesetzte Struktur der Jacobi-Matrix setzt sich bei der Hesse-Matrix des Optimierungsfunktionals fort. Unter Hinzunahme von Beschränkungen erweitert sich die Jacobi-Matrix entsprechend den Nebenbedingungen, ist aber ebenfalls durch eine schwachbesetzte Blockstruktur gekennzeichnet. Auch hier empfiehlt sich die Berücksichtigung dieser speziellen Struktur im nichtlinearen Optimierungsprogramm.

Im folgenden soll auf eine erst in jüngerer Zeit von Oskar von Stryk entwickelte Idee (die sogenannte Hybride-Methode) eingegangen werden.
Es kann gezeigt werden, daß sich zumindest Teile der Optimalitätsbedingungen für Steuerprozesse in den Optimalitätsbedingungen der nichtlinearen Optimierung wiederspiegeln. Es läßt sich ein Zusammenhang zwischen den adjungierten Variablen des Steuerproblems und den Lagrange-Multiplikatoren

des Optimierungsproblems aufzeigen, welcher eine Möglichkeit zur Bestimmung einer Startschätzung der adjungierten Variablen aus den Lagrange-Multiplikatoren angibt. Motiviert wird dieser Ansatz weiterhin dadurch, daß viele der existierenden Optimierungsprogramme (so auch OPRQP) in Erweiterung der Ausgabe die Werte der Lagrange-Multiplikatoren beinhalten, so daß nach der Bestimmung einer optimalen Lösung durch die direkte Kollokation, eine Startschätzung der adjungierten Variablen durch einfachste Umrechnung erhalten werden kann.

4.2.5 Bestimmung einer Anfangsschätzung für die adjungierten Variablen

In den folgenden Abschnitten soll ein Zusammenhang zwischen den Kuhn-Tucker-Bedingungen des diskretisierten optimalen Steuerprozesses und den notwendigen Optimalitätsbedingungen des Ausgangsproblems aufgezeigt werden. Die so ermittelten Informationen können zur Bestimmung einer Schätzung der adjungierten Variablen an diskreten Punkten genutzt werden.

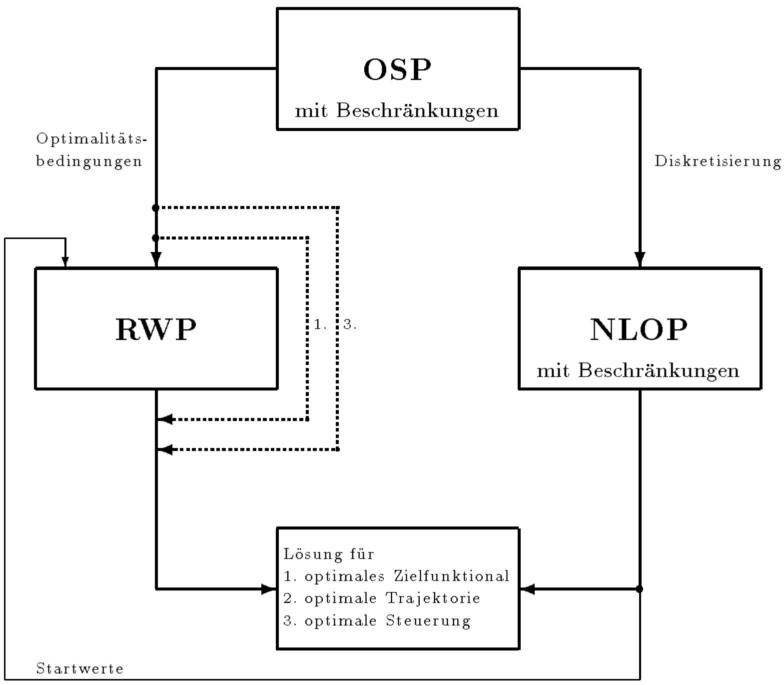


Abbildung 4.6: Schematische Übersicht zur Kombination eines direkten Verfahrens mit dem indirektenden Ansatz.

Für diesen Abschnitt seien die folgenden Voraussetzungen gültig:

Voraussetzung 4.4

1. *Die Steuerungen u_j, $j = 1, \ldots, m$ treten nichtlinear in f auf.*
2. *Die optimale Steuerung ist stetig.*
3. *Die Endzeit ist fest vorgegeben.*
4. *Die Anzahl $\bar{s}$ der unter Ungleichungen beschriebenen Nebenbedingungen ist 1.*
5. *Die Beschränkung $S = S_1$ ist in einem Intervall $[t_{entry}, t_{exit}]$ entlang der optimalen Trajektorie aktiv. $t_0 < t_{entry} < t_{exit} < t_f$*
6. *Die Steuerung wird linear interpoliert.*

Abbildung 4.6 verdeutlicht den zu erwartende Zusammenhang mit dem indirekten Ansatz.

Notwendige Optimalitätsbedingungen des optimalen Steuerprozesses

Mit den Mitteln von Kapitel 2 ist bekannt, daß es zum Steuerprozeß (4.2) einen Satz von n adjungierten Variablen

$$\lambda(t) = (\lambda_1(t), \ldots, \lambda_n(t))^T$$

und eine Multiplikatorfunktion $\mu(t)$ gibt, so daß sich die erweiterte Hamilton-Funktion zu

$$\tilde{H}(x(t), \lambda(t), \mu(t), u(t)) = \sum_{k=1}^{n} \lambda_k(t) f_k(x, u, t) + \mu(t) S(x, u, t) \tag{4.36}$$

ergibt.

Unter Anwendung des erweiterten Minimumprinzips 2.40 ergeben sich die notwendigen Optimalitätsbedingungen des Ausgangsproblems (4.2) für $i = 1, \ldots, n$ und $j = 1, \ldots, m$ zu

$$\dot{x}_i(t) = \frac{\partial \tilde{H}}{\partial \lambda_i} = f_i(x, u, t), \tag{4.37}$$

$$\dot{\lambda}_i(t) = -\frac{\partial \tilde{H}}{\partial x_i} = -\sum_{k=1}^{n} \lambda_k(t) \frac{\partial f_k(x, u, t)}{\partial x_i} - \mu(t) \frac{\partial S(x, u, t)}{\partial x_i}, \tag{4.38}$$

$$0 = \frac{\partial \tilde{H}}{\partial u_j} = \sum_{k=1}^{n} \lambda_k(t) \frac{\partial f_k(x, u, t)}{\partial u_j} + \mu(t) \frac{\partial S(x, u, t)}{\partial u_j}. \tag{4.39}$$

Auf inneren Teilstücken gilt dabei

$$S(x,u,t) < 0 \text{ und } \mu(t) = 0. \tag{4.40}$$

Auf Randstücken gilt dagegen

$$S(x,u,t) = 0 \text{ und } \mu(t) \geq 0. \tag{4.41}$$

Ergänzend müssen auch die Rand-, Transversalitäts- und Sprungbedingungen erfüllt sein, die jedoch hier für die weiteren Untersuchungen nicht von Bedeutung sind.

Notwendige Optimalitätseigenschaften des diskretisierten optimalen Steuerprozesses

Zur Reduzierung des Schreibaufwandes in den nachfolgenden Untersuchungen trifft man die Vereinbarungen

1. $n = 1$ und $m = 1$,
2. $u_i := u(t_i)$ und $x_i := x(t_i)$,
3. $f_i := f(x_i, u_i, t_i)$ und $f_{i+1/2} := f(x(t_{i+1/2}), u(t_{i+1/2}), t_{i+1/2})$.

Die Lagrange-Funktion des aus dem optimalen Steuerprozeß durch Diskretisierung mittels Kollokation entstandenen nichtlinearen Optimierungsproblems lautet für $\eta = (\eta_1, \ldots, \eta_{N-1})^T \in \mathrm{I\!R}^{N-1}$, $\sigma = (\sigma_1, \ldots, \sigma_N)^T \in \mathrm{I\!R}^N$ und $\nu = (\nu_1, \ldots, \nu_r)^T \in \mathrm{I\!R}^r$:

$$\begin{aligned} \tilde{L}(Y, \eta, \sigma, \nu) \;=\; & g(x_N, t_N) \\ & - \sum_{j=1}^{N-1} \eta_j \Big(f(x_{j+1/2}, u_{j+1/2}, t_{j+1/2}) - \dot{x}_{app}(t_{j+1/2}) \Big) \\ & - \sum_{j=1}^{N} \sigma_j S(x_j, u_j, t_j) - \sum_{j=1}^{r} \nu_j R_j(x_1, x_N, t_N) \end{aligned} \tag{4.42}$$

Eine Lösung des nichtlinearen Problems muß nach der Lagrangeschen Multiplikatorenregel (oder Kuhn-Tucker-Bedingungen) unter anderem die folgenden Optimalitätsbedingungen erfüllen:

$$\left.\begin{aligned} \frac{\partial \tilde{L}}{\partial x_i} &= 0 \\ \frac{\partial \tilde{L}}{\partial u_i} &= 0 \\ \frac{\partial \tilde{L}}{\partial \eta_j} &= 0 \end{aligned}\right\}, \qquad i = 1, \ldots, N, \quad j = 1, \ldots, N-1 \tag{4.43}$$

mit $\quad S(x_i,u_i,t_i) < 0$ und $\sigma_i = 0$
oder $\quad S(x_i,u_i,t_i) = 0$ und $\sigma_i \geq 0$

Konkret erhält man also an den Stellen t_i, $i = 2,\ldots,N$

$$\begin{aligned} 0 = \frac{\partial \tilde{L}}{\partial u_i} &= -\eta_{i-1}\left(\frac{\partial f(x_{i-1/2},u_{i-1/2},t_{i-1/2})}{\partial u_i} - \frac{\partial \dot{x}(t_{i-1/2})}{\partial u_i}\right) \\ &\quad -\eta_i\left(\frac{\partial f(x_{i+1/2},u_{i+1/2},t_{i+1/2})}{\partial u_i} - \frac{\partial \dot{x}(t_{i+1/2})}{\partial u_i}\right) \\ &\quad -\sigma_i\frac{\partial S(x(t_i),u(t_i),t_i)}{\partial u_i} \end{aligned} \tag{4.44}$$

bzw. für $i = 2,\ldots,N-1$

$$\begin{aligned} 0 = \frac{\partial \tilde{L}}{\partial x_i} &= -\eta_{i-1}\left(\frac{\partial f(x_{i-1/2},u_{i-1/2},t_{i-1/2})}{\partial x_i} - \frac{\partial \dot{x}(t_{i-1/2})}{\partial x_i}\right) \\ &\quad -\eta_i\left(\frac{\partial f(x_{i+1/2},u_{i+1/2},t_{i+1/2})}{\partial x_i} - \frac{\partial \dot{x}(t_{i+1/2})}{\partial x_i}\right) \\ &\quad -\sigma_i\frac{\partial S(x(t_i),u(t_i),t_i)}{\partial x_i}. \end{aligned} \tag{4.45}$$

Konvergenzeigenschaften

Der Einfachheit halber wählt man nun für die weiteren Untersuchungen ein äquidistantes Gitter mit

$$h := h_i = t_{i+1} - t_i = \frac{t_N - t_1}{N-1} = \frac{t_f - t_0}{N-1}, \qquad i = 1,\ldots,N-1 \tag{4.46}$$

Nach Anwendung des Korollars zur Kettenregel der Differentiation [Forster 1987, S.50] und einigen Vereinfachungen erhält man aus (4.44):

$$\begin{aligned} \frac{\partial \tilde{L}}{\partial u_i} = &-\eta_{i-1}\Bigg(\frac{\partial f_{i-1/2}}{\partial x_{i-1/2}}\frac{\partial x_{i-1/2}}{\partial u_i} + \frac{\partial f_{i-1/2}}{\partial u_{i-1/2}}\underbrace{\frac{\partial u_{i-1/2}}{\partial u_i}}_{1/2} \\ &\qquad + \underbrace{\frac{\partial f_{i-1/2}}{\partial t_{i-1/2}}\overbrace{\frac{\partial t_{i-1/2}}{\partial u_i}}^{0}}_{0} - \frac{-\frac{1}{4}\partial(f_{i-1}+f_i)}{\partial u_i}\Bigg) \\ &-\eta_i\Bigg(\frac{\partial f_{i+1/2}}{\partial x_{i+1/2}}\frac{\partial x_{i+1/2}}{\partial u_i} + \frac{\partial f_{i+1/2}}{\partial u_{i+1/2}}\underbrace{\frac{\partial u_{i+1/2}}{\partial u_i}}_{1/2} \\ &\qquad + \underbrace{\frac{\partial f_{i+1/2}}{\partial t_{i+1/2}}\overbrace{\frac{\partial t_{i+1/2}}{\partial u_i}}^{0}}_{0} - \frac{-\frac{1}{4}\partial(f_i+f_{i+1})}{\partial u_i}\Bigg) \\ &-\sigma_i\frac{\partial S(x(t_i),u(t_i),t_i)}{\partial u_i} \end{aligned} \tag{4.47}$$

Ferner ergibt sich nach einigen weiteren Umformungen und Vereinfachungen aus (4.47)

$$\begin{aligned}\frac{\partial \tilde{L}}{\partial u_i} = & -\frac{1}{2}\left(\eta_{i-1}\frac{\partial f_{i-1/2}}{\partial u_{i-1/2}} + \eta_i \frac{\partial f_{i+1/2}}{\partial u_{i+1/2}}\right) - \frac{1}{4}\frac{\partial f_i}{\partial u_i}\left(\eta_{i-1} + \eta_i\right) \\ & +\frac{1}{8}h\frac{\partial f_i}{\partial u_i}\left(\eta_{i-1}\frac{\partial f_{i-1/2}}{\partial x_{i-1/2}} - \eta_i \frac{\partial f_{i+1/2}}{\partial x_{i+1/2}}\right) \\ & -\sigma_i \frac{\partial S(x(t_i), u(t_i), t_i)}{\partial u_i}\end{aligned} \tag{4.48}$$

Untersuchung des Grenzüberganges $h \longrightarrow 0$ führt für $t = t_i$ fest auf[10]

$$\begin{aligned}\frac{\partial \tilde{L}}{\partial u_i} = & -\frac{1}{2}\left(\eta_i\frac{\partial f_i}{\partial u_i} + \eta_i \frac{\partial f_i}{\partial u_i}\right) - \frac{1}{4}\frac{\partial f_i}{\partial u_i}\left(\eta_i + \eta_i\right) \\ & -\sigma_i \frac{\partial S(x(t_i), u(t_i), t_i)}{\partial u_i}\end{aligned} \tag{4.49}$$

und schließlich zu

$$\frac{3}{2}\eta_i \frac{\partial f_i}{\partial u_i} + \sigma_i \frac{\partial S(x(t_i), u(t_i), t_i)}{\partial u_i} \quad = 0 \tag{4.50}$$

Diese Gleichung ist aber äquivalent zur Bedingung (4.39)

Analoge Berechnungen für $\frac{\partial \tilde{L}}{\partial x_i}$ führen zu

$$\begin{aligned}\frac{\partial \tilde{L}}{\partial x_i} = & -\frac{3}{2}\left(\frac{\eta_i}{h} - \frac{\eta_{i-1}}{h}\right) - \frac{1}{2}\left(\eta_{i-1/2}\frac{\partial f_{i-1/2}}{\partial x_{i-1/2}} + \eta_{i+1/2}\frac{\partial f_{i+1/2}}{\partial x_{i+1/2}}\right) \\ & -\frac{1}{4}\frac{\partial f_i}{\partial x_i}\left(\eta_{i-1} + \eta_i\right) - \sigma_i \frac{\partial S(x(t_i), u(t_i), t_i)}{\partial x_i}\end{aligned} \tag{4.51}$$

Weiterhin erhält man für $h \longrightarrow 0, \quad t = t_i$ fest:

$$\frac{3}{2}\dot{\eta}_i + \frac{3}{2}\eta_i \frac{\partial f_i}{\partial x_i} + \sigma_i \frac{\partial S(x(t_i), u(t_i), t_i)}{\partial u_i} \quad = 0 \tag{4.52}$$

Diese Gleichung ist äquivalent zu (4.39).

Bestimmung einer Anfangsschätzung der adjungierten Variablen

Es spiegelt sich zumindest ein Teil der notwendigen Optimalitätsbedingungen des optimalen Steuerprozesses in den notwendigen Bedingungen des diskretisierten Problems wieder.

[10] indem $N \longrightarrow \infty$

Mit (4.52) zeigt sich aber gleichzeitig eine Möglichkeit auf, eine Schätzung der adjungierten Variablen an diskreten Punkten über die Lagrange-Multiplikatoren zu bekommen.
Mit einem $\omega \in \mathbb{R}$ konstant gilt:

$$\lambda(t_{i+1/2}) = -\frac{3}{2}\omega\eta_i, \qquad i = 1, \ldots, N-1 \tag{4.53}$$

Der Faktor ω ist abhängig von der Diskretisierung, kann aber leicht über die Transversalitätsbedingung ermittelt werden.

Im Falle von Beschränkungen S, kann auf gleiche Weise eine Schätzung der Multiplikatoren μ erlangt werden.

Es liegt nahe, die so erhaltene (relativ schlechte) Lösung der Trajektorie und der adjungierten Differentialgleichungen als Startwerte für ein iteratives Verfahren (z.B. BDSCO) zur Lösung der resultierenden Randwertaufgabe der analytischen Lösung zu verwenden.

4.2.6 Nachteile und Schwierigkeiten bei der praktischen Realisierung direkter Verfahren durch Diskretisierung der Steuer- und Zustandsfunktionen

Bei der praktischen Realisierung von direkten Verfahren durch Diskretisierung der Steuer- und Zustandsfunktionen treten einige Probleme auf, die im folgenden kurz zusammengefaßt werden sollen:

1. Die Lösung des optimalen Zielfunktionals, der Trajektorie, der Steuerung und der adjungierten Differentialgleichungen sind nicht mit der Genauigkeit zu erreichen, wie man sie bei den indirekten Verfahren erhält. Speziell die Steuerung und die adjungierten Variablen weichen oft stark von der optimalen Lösung ab (vgl. Kapitel 5).

2. Der diskretisierte optimale Steuerprozeß kann durch mehrere Minima gekennzeichnet sein, so daß bei der Ausführung der direkten Verfahren das absolute Minimum nicht gefunden wird und das Verfahren in einem 'Pseudominima' hängen bleibt (vgl. Kapitel 5.5.1).
   Die so gefundene Lösung kann stark von der tatsächlichen optimalen Lösung abweichen.

3. Auch wenn das Verfahren einen großen Konvergenzradius besitzt, muß sowohl eine Anfangsschätzung der Steuerung, als auch der Trajektorie

angegeben werden, deren Bestimmung mit Problemen behaftet sein kann.

4. Realisiert man diese direkten Verfahren ohne Berücksichtigung der speziellen Blockstruktur in das Optimierungsprogramm, so ist ein großer Speicheraufwand (und damit verbunden auch Rechenaufwand) erforderlich. In jedem Iterationsschritt werden dabei z.B. für die direkte Kollokation bei der Routine OPRQP Matrizen $B1$, Q und A berechnet, die ein Format von $(N(n{+}m)\times N(n{+}m))$ bzw. $(N(n{+}m)\times N(n{+}s){+}r)$ besitzen.
   Speicher- und Rechenaufwand betragen also allein bzgl. dieser Matrizen $\mathbf{O}(\mathbf{N^2}(\mathbf{n}+\mathbf{m})^\mathbf{2})$ bzw. $\mathbf{O}(\mathbf{N^2}(\mathbf{n^2}+\mathbf{ns}+\mathbf{nm}+\mathbf{ms}))$.

   Rechenbeispiel : Mit $N = 100$, $n = 7$, $m = 1$, $s = 2$ *benötigt man* 640000 *bzw.* 720000 *Speicherplätze (je* 8 *Byte für Double Precision). Diese Matrizen nehmen allein* 2000000 *Speicherplätze in Anspruch, also fast* 16*MB. – Auf einem DOS-Rechner, der einen freien Arbeitsspeicher von weniger als* $0,64$*MB besitzt, wäre die Bearbeitung also unmöglich.*
   Ebenso benötigt man pro Iteration ganze Rechengänge nur für die 2000000 *Speicherplätze, was nur in etwa den riesigen Rechen- und damit Zeitaufwand verdeutlicht.*

Die erläuterten Punkte 1, 2 und 4 beschreiben Nachteile, die schon allein aufgrund des Ansatzes durch Diskretisierung auftreten und somit wohl allen direkten Verfahren mehr oder weniger anhaften. Punkt 3 ergibt sich schon aus der Idee der direkten Verfahren.

Es ist also nicht zu erwarten, daß mittels weiterer diskreter Ansätze die Probleme 1-4 vollständig beseitigt werden können. Dennoch sind es gerade die direkten Verfahren, die einen Einblick in die optimale Lösung eines Steuerprozesses erlauben, ohne daß man den komplizierten und zeitaufwendigen mathematischen Apparat einsetzen muß.

Eine Verbesserung der Güte der Lösung und eine Reduzierung des Speicher- und Rechenaufwandes (und damit auch des Zeitaufwandes) scheint hierbei die vornehmlichste Motivation zur Weiterentwicklung der direkten Verfahren zu sein.

4.3 Direkte Verfahren durch Diskretisierung der Steuerfunktion kombiniert mit dem Lösen von Anfangswertproblemen

Da der Lösungsaufwand der Optimierungsaufgabe nichtlinear (aller Erfahrung nach mindestens quadratisch) mit der Zahl der freien Parameter steigt, sollen in diesem Kapitel Wege aufgezeigt werden, wie der Optimierungsvektor möglichst klein gehalten werden kann. Der einfachste Weg scheint dabei zu sein, die Zustandsfunktion nicht mehr zu diskretisieren und in das Optimierungsproblem aufzunehmen, sondern zumindest auf einen Teil (im Idealfall auf alle) dieser Optimierungsvariablen zu verzichten und das als Nebenbedingung eingehende Differentialgleichungssystem durch Anfangswertroutinen zu lösen.

4.3.1 Die direkte Mehrzielmethode nach Bock

In Analogie zu den bereits beschriebenen Verfahren ist die grundlegende Idee der direkten Mehrzielmethode die Reduzierung eines unendlich dimensionalen Steuerproblems (4.2) auf ein endlich dimensionales Optimierungsproblem. Hier soll dazu ein multiple-shooting-Ansatz beschrieben werden, der zum Lösen des aus dem Differentialgleichungssystem und den Randbedingungen bestehenden Randwertproblems hinzugezogen wird.

Für ein beliebig gewähltes Gitter

$$t_0 = t_1 < t_2 < \ldots < t_n = t_f \tag{4.54}$$

soll zu gegebenen Basisfunktionen $\mathcal{B}_i^{(k)}$ die Steuerfunktion auf jedem der Teilintervalle durch m_i Parameter approximiert werden (z.B. stückweise durch Polynome).

$$u(t) \approx u_{app,i}(t, \alpha_i),\ \alpha_i \in \mathrm{I\!R}^{m_i},\ t \in [t_1, t_{i+1}],\ i = 1, \ldots, N-1. \tag{4.55}$$

Die Zustandsvariable wird auf jedem der Teilintervalle durch die berechnete Lösung $x(t; s_i, \alpha_i)$ des Anfangswertproblems

$$\begin{array}{rcl} \dot{x}(t) & = & f(x(t), u_{app,i}(t, \alpha_i), t) \\ x(t_i) & = & s_i, \qquad t \in [t_i, t_{i+1}] \end{array} \tag{4.56}$$

als Funktionen von s_i und α_i ersetzt. Mit den Optimierungsvariablen

$$\mathbf{Y} = [s_1, \alpha_1, \ldots, s_{N-1}, \alpha_{N-1}, s_N] \in \mathrm{I\!R}^{\sum_{i=1}^{N-1} m_i + Nn} \tag{4.57}$$

ergibt sich das resultierende nichtlineare Optimierungsproblem zusammen mit einem Fehler

$$\zeta_i := x(t_{i+1}, s_i, \alpha_i) - s_{i+1}, \quad i = 1, \ldots, N-1 \tag{4.58}$$

zu

$$\begin{array}{lll} \min & F(\mathbf{Y}), & F : \mathbb{R}^{\sum_{i=1}^{N-1} m_i + Nn} \longrightarrow \mathbb{R} \\ \text{unter} & & \end{array}$$

$$\begin{array}{rcl} \zeta_1 & = & 0 \\ \vdots & \vdots & \vdots \\ \zeta_{N-1} & = & 0 \\ R(s_1, s_N, t_N) & = & 0 \end{array} \tag{4.59}$$

und

$$\begin{array}{rcl} S(s_1, u(t_1), t_1) & \leq & 0 \\ \vdots & \vdots & \vdots \\ S(s_N, u(t_N), t_N) & \leq & 0 \end{array}$$

Durch die Bedingungen $\zeta_i = 0$, $i = 1, \ldots, N-1$ wird wie bei einem multiple-shooting Verfahren die Stetigkeit des Zustandes erreicht.
Mit der von mir umgesetzten Routine DIRMESCH liegt eine Realisierung dieser Diskretisierungsidee mit stückweise linear interpolierter Steuerung, unter Benutzung eines Runge-Kutta-Fehlberg-Verfahrens mit automatischer Schrittweitensteuerung der Ordnung sieben und der nichtlinearen Optimierungsroutine OPRQP, vor.

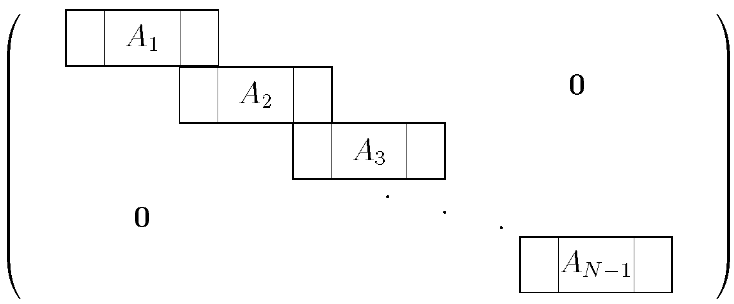
4.3.2 Spezielle Eigenschaften der direkten Mehrzielmethode

Die Aufnahme der zusätzlichen Optimierungsvariablen s_i, $i = 1, \ldots, N$ hat verschiedene Vorteile:

1. Vorteile die sich schon theoretisch aus dem Ansatz des multiple-shooting ergeben wie
   (a) Beschleunigte Konvergenz.
   (b) Vergrößerter Konvergenzradius (numerische Stabilität).
   (c) Niedriger Diskretisierungs- und Rundungsfehlereinfluß (im Gegensatz zum einfachen Schießverfahren).
   (d) Bewältigung von Instabilitäten in der Systemdynamik.

2. Weitere Vorteile, die sich aus der Untersuchung des resultierenden Optimierungsproblems ergeben.

Weiterhin kann auch die direkte Mehrzielmethode durch große, schwachbesetzte, sowie durch eine gewisse Blockstruktur der beteiligten Matrizen charakterisiert werden. Im Gegensatz zu den bisher beschriebenen Verfahren weisen jedoch die hier eingehenden Blöcke eine in Abhängigkeit von den Parametern $\alpha_i \in \mathbb{R}^{m_i}$, $i = 1, \ldots, N-1$ gekennzeichnete dynamische Dimension aus, sofern die m_i von unterschiedlicher Dimension gewählt werden. Die Jacobi-Matrix der dynamischen Nebenbedingungen hat dann die in Abbildung 4.7 verdeutlichte Struktur.



(4.60)

Abbildung 4.7: Spezielle Blockstruktur der Jacobi-Matrix der dynamischen Nebenbedingungen bei der direkten Mehrzielmethode

Die Blöcke $A_1, \ldots, A_{N-1}$ besitzen die Dimensionen $(n \times 2n + m_i)$ und ergeben sich für $i = 1, \ldots, N-1$ im einzelnen zu

$$A_i = \begin{pmatrix} \frac{\partial \zeta_i^{(1)}}{\partial s_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(1)}}{\partial s_i^{(n)}} & \frac{\partial \zeta_i^{(1)}}{\partial \alpha_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(1)}}{\partial \alpha_i^{(m_i)}} & \frac{\partial \zeta_i^{(1)}}{\partial s_{i+1}^{(1)}} & \cdots & \frac{\partial \zeta_i^{(1)}}{\partial s_{i+1}^{(n)}} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial \zeta_i^{(n)}}{\partial s_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(n)}}{\partial s_i^{(n)}} & \frac{\partial \zeta_i^{(n)}}{\partial \alpha_i^{(1)}} & \cdots & \frac{\partial \zeta_i^{(n)}}{\partial \alpha_i^{(m_i)}} & \frac{\partial \zeta_i^{(n)}}{\partial s_{i+1}^{(1)}} & \cdots & \frac{\partial \zeta_i^{(n)}}{\partial s_{i+1}^{(n)}} \end{pmatrix} \tag{4.61}$$

Offensichtlich ist eine weitere Reduzierung des Speicher- und Rechenaufwandes wie in Kapitel (4.2.1) beschrieben möglich, denn es gilt :

$$\frac{\partial \zeta_i^{(j)}}{\partial s_{i+1}^{(j)}} = 1, \quad i = 1, \ldots, N-1, \quad j = 1, \ldots, n \tag{4.62}$$

Die so gefundene Struktur setzt sich in der Hesse-Matrix des Zielfunktionals, sowie bei einer Erweiterung des Problems um allgemeine Beschränkungen in der Jacobi-Matrix, fort.

Ein weiterer Vorteil dieses Verfahrens gegenüber allen anderen in dieser Arbeit beschriebenen, liegt in der Verfügbarkeit einer nicht diskreten Trajektorie, da zwischen zwei Mehrzielknoten im Rahmen der vorgegebenen Genauigkeitsschranken des Integerators (z.B. Runge-Kutta-Fehlberg) die Trajektorie 'exakt' bezüglich der interpolierten Steuerung berechnet und angegeben wird. Es ist allerdings nicht zu erwarten, daß der optimale Steuerprozeß dadurch genauer als bei den anderen[11] gelöst wird. Die gefundene Lösung ist nur von niedrigerer Ordnung, jedoch hat die berechnete Trajektorie zur interpolierten Steuerung einen sehr viel geringeren Fehler (abhängig von der Ordnung der Integrationsroutine). Andererseits ergibt sich bei dieser Methode ein Nachteil durch den hohen Rechenaufwand, der durch die aufwendigen Integrationen entsteht.

4.3.3 Lösung eines optimalen Steuerprozesses durch Problemsharing

Schon rein äußerlich betrachtet unterteilt sich ein optimaler Steuerprozeß in drei Hauptkomponenten:

1. Zielfunktional
2. Systemdynamik mit Randbedingungen
3. Steuerung und weitere Beschränkungen

Wäre die optimale Steuerung u bekannt, so läge mit Punkt 2 ein in sich abgeschlossenes System (Randwertproblem) zugrunde, welches mit den in Kapitel 3 beschriebenen Techniken gelöst werden könnte[12].
Andererseits könnte bei bekannter optimaler Trajektorie die Steuerung durch einfache Berechnungen ermittelt werden. Aufwendige Berechnungen für Optimierungsvariablen und Nebenbedingungen entfielen.
Es soll nun diese Idee des Problemsharing aufgenommen werden, um ein direktes Verfahren aufzuzeigen, welches durch gleichzeitige iterative Verbesserung der Trajektorie und der Steuerung unter Berücksichtigung von Beschränkungen die optimale Lösung zu einem gegebenen Zielfunktional liefert. Motiviert wird dieser Ansatz durch eine Verringerung der Optimierungsvariablen und Nebenbedingungen, sowie durch die Verbesserung der Lösungsergebnisse bei gleichzeitiger Reduzierung des Rechen- und Zeitaufwandes.

[11] Eher sogar ungenauer, weil die obere Fehlerschranke für die Interpolation der Steuerung auf jeden Fall erreicht wird (vgl. Sätze 4.5, 4.6 und 4.7).

[12] Dies ist die Grundidee der Ausnutzung der Optimalitätsbedingungen von Pontryagin.

Abbildung 4.8 verdeutlicht das Zusammenspiel der unterschiedlichen Komponenten dieses Ansatzes.

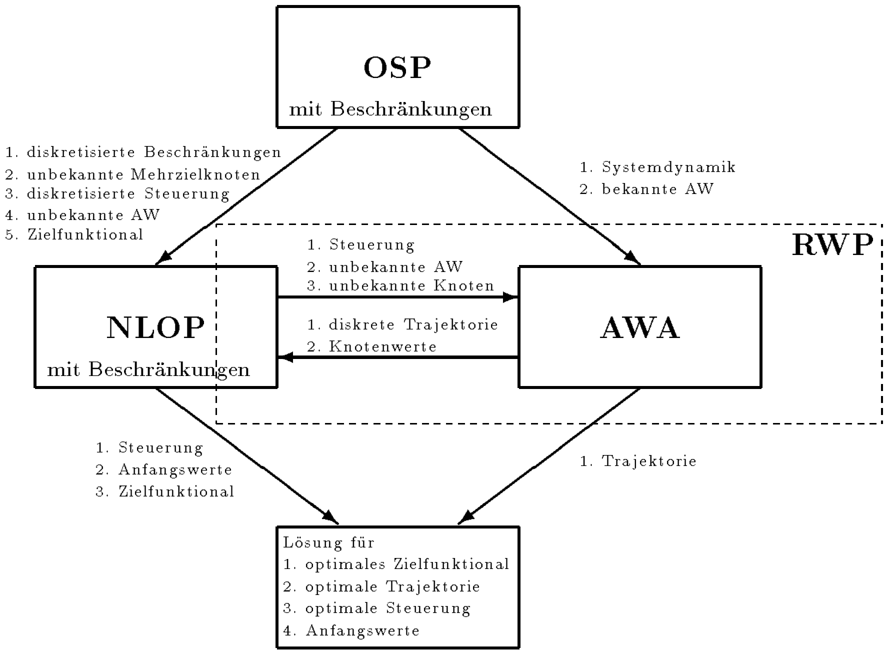


Abbildung 4.8: Schematische Übersicht zur direkten Lösung eines optimalen Steuerprozesses durch Problemsharing

Es ist offensichtlich, daß die zu erwartende Konvergenzordnung eines direkten Verfahrens abhängig von den Fehlern der Integrationsroutine, der Interpolation der Steuerung, der geforderten Genauigkeit der Lösungsroutine für das nichtlineare Optimierungsprogramm und der numerischen Differentiation ist. Vernachlässigt man die letzteren beiden eingehenden Fehler, so ist zu erwarten, daß in der Regel durch das Minimum der Ordnungen der approximierten Steuerung und Integration eine obere Schranke für die Konvergenzordnung des gesamten Verfahrens gegeben ist.
Dies bedeutet aber, daß die hohe Integrationsordnung bei der direkten Mehrzielmethode bei gleichzeitiger schlechter Approximation der Steuerung aus Sicht des Rechenaufwandes nicht vertretbar ist.
Bei dem hier beschriebenen Verfahren soll daher sowohl die Ordnung als auch die Schrittweite der Integrationsroutine der Ordnung der Steuerinterpolation und der diskreten Zeitunterteilung angepaßt werden.
Im Gegensatz zum indirekten Verfahren, bei dem das Optimierungsintervall $[t_0, t_f]$ wegen der eventuellen Instabilität der adjungierten Differentialglei-

chungen in Vorwärtsintegrationsrichtung in mehrere Schießintervalle unterteilt werden muß, kann man, wie schon theoretisch beim direkten Mehrzielverfahren mit nur einem Knoten beschrieben, über das ganze Intervall zielen, da nur die Zustandsdifferentialgleichungen integriert werden.
Die Reduzierung des (aus Sicht der nichtlinearen Optimierung) unendlich dimensionalen Steuerproblems (4.2) mit den Standardrandbedingungen (2.8) auf ein endlich dimensionales Optimierungsproblem ist das Ziel des hier beschriebenen Verfahrens. Dazu diskretisiert man das Intervall $[t_0, t_f]$ in ein (nicht zwangsweise äquidistantes) Gitter.

$$t_0 = t_1 < t_2 < \ldots < t_N = t_f \tag{4.63}$$

In einem ersten Anzatz soll auf die Einführung von Mehrzielknoten als zusätzliche Optimierungsvariablen verzichtet werden. Der Optimierungsvektor $\mathbf{Y}$ besteht dann aus den Werten der Steuerungen an den diskreten Punkten und einer Ergänzung des Anfangswertes $x(t_0) \in \mathbb{R}^{n-l}$ um l Optimierungsvariablen $s \in \mathbb{R}^l$. Eine eventuell freie Endzeit geht damit ebenfalls nach einer geeigneten Transformation in einen Steuerprozeß mit fester Endzeit als unbekannter Anfangswert in den Optimierungsvektor ein.

$$\mathbf{Y} = [u(t_1), u(t_2), \ldots, u(t_N), s] \in \mathbb{R}^{Nm+l} \tag{4.64}$$

Da bei der Bestimmung der Trajektorie einige der Integrationsroutinen eine Schätzung der Steuerung an Punkten im Inneren der Teilintervalle benötigen, muß die Steuerfunktion, wenn keine zusätzlichen Optimierungsvariablen eingeführt werden sollen, an diesen Punkten in geeigneter Weise approximiert werden.
Für $h_i := t_{i+1} - t_i$, $i = 1, \ldots, N-1$ stehen die folgenden Möglichkeiten zur Verfügung.

Approximation der Steuerfunktionen

Stückweise konstante Approximation der Steuerung

$$u_{app}(t) = u(t_i), \quad t \in [t_i, t_{i+1}[, \; i = 1, \ldots, N-1 \tag{4.65}$$

Für diese Art der Approximation gilt der Satz

Satz 4.5 *Sei $u(t) \in C^1([t_0, t_f])$ und u_{app} nach der oben genannten Vorschrift konstruiert, dann wird durch*

$$|u(t) - u_{app}(t)| \leq \max_{\tau \in [t_0, t_f]} |u^{(1)}(\tau)| (\max_k \; h_k) \quad \forall \, t \in [t_0, t_f] \tag{4.66}$$

eine obere Fehlerschranke angegeben.

Stückweise lineare Interpolation der Steuerung

$$u_{app}(t) = u(t_i)+(t-t_i)\frac{u(t_{i+1}) - u(t_i)}{h_i}, \quad t \in [t_i, t_{i+1}[, \ i = 1, \dots, N-1 \quad (4.67)$$

Als Abschätzung für die obere Fehlerschranke bei der linearen Interpolation der Steuerung hilft der Satz

Satz 4.6 *Sei $u(t) \in C^2([t_0, t_f])$ und u_{app} nach der oben genannten Vorschrift konstruiert, dann wird durch*

$$|u(t) - u_{app}(t)| \leq \frac{1}{8} \max_{\tau \in [t_0, t_f]} |u^{(2)}(\tau)| (\max_k \ h_k)^2 \quad \forall \, t \in [t_0, t_f] \qquad (4.68)$$

eine obere Fehlerschranke bei der stückweise linearen Interpolation angegeben.

Kubische Interpolation der Steuerung

Zu einer auf $[t_0, t_f]$ gegebenen reellen Funktion soll eine reelle Funktion $u_{app}(t)$ angegeben werden, die $u(t)$ durch kubische Splines interpoliert. Dazu wird u_{app} auf jedem der Teilintervalle $[t_i, t_{i+1}]$, $i = 1, \dots, N-1$ durch ein Polynom vom Grade ≤ 3 definiert.

$$u_{app}^{(i)}(t) = \alpha_i + \beta_i(t - t_1) + \gamma_i(t - t_i)^2 + \delta_i(t - t_i)^3, \quad t \in [t_i, t_{i+1}] \qquad (4.69)$$

Unterschiedliche Approximationsvorschriften führen zu unterschiedlichen Koeffizientensätzen $\alpha_i, \ \beta_i, \ \gamma_i, \ \delta_i \in \mathbb{R}^m$. Stellvertretend für andere Verfahren wird hier die Version beschrieben, bei der sich die $4m(N-1)$ Koeffizienten in eindeutiger Weise aus den $2m(N-1) + 2m(N-2) + 2m = 4m(N-1)$ Bedingungen

$$\begin{array}{ll} \left.\begin{aligned} u_{app}^{(i)}(t_i) &= u(t_i) \\ u_{app}^{(i)}(t_{i+1}) &= u(t_{i+1}) \end{aligned}\right\} & i = 1, \dots, N-1 \\ \left.\begin{aligned} \dot{u}_{app}^{(i)}(t_{i+1}) &= \dot{u}_{app}^{(i+1)}(t_i) \\ \ddot{u}_{app}^{(i)}(t_{i+1}) &= \ddot{u}_{app}^{(i+1)}(t_i) \end{aligned}\right\} & i = 1, \dots, N-2 \\ \dot{u}_{app}^{(1)}(t_0) \quad = \dot{u_0} & \\ \dot{u}_{app}^{(N-1)}(t_f) = \dot{u_f} & \end{array} \qquad (4.70)$$

ergeben[13]. Die Unbekannten $\dot{u_0}$ und $\dot{u_f}$ gehen als weitere Optimierungsvariablen in das nichtlineare Optimierungsproblem ein, so daß sich der Optimie-

[13]Direkte Konsequenz der stetigen Differenzierbarkeit der approximierten Steuerung kann die Verschlechterung der Ergebnisse bei einer nicht stetig differenzierbaren Steuerung sein. Durch Einführung zusätzlicher Optimierungsvariablen für den Wert der Steuerung an Unstetigkeitsstellen und für die entsprechenden Ableitungen an diesen Punkten kann dieses Problem aber umgangen werden, wenn man die Interpolation nicht über das gesamte Optimierungsintervall, sondern jeweils nur über die entsprechenden Teilintervalle durchführt.

rungsvektor dann zu

$$\mathbf{Y} = [u(t_1), u(t_2), \ldots, u(t_N), s, \dot{u}_0, \dot{u}_f] \in \mathbb{R}^{(N+2)m+l} \tag{4.71}$$

ergibt.
Damit lassen sich für $i = 1, \ldots, N-1$ nach einigen Berechnungen die Koeffizienten zu

$$\begin{aligned}
\alpha_i &= u(t_i) \\
\beta_i &= \frac{u(t_{i+1}) - u(t_i)}{h_i} - h_i \frac{2M_i + M_{i+1}}{6} \\
\gamma_i &= \frac{u_i}{2} \\
\delta_i &= \frac{M_{i+1} - M_i}{6h_i}
\end{aligned} \tag{4.72}$$

für gewisse Momente $M_i := \ddot{u}_{app}(t_i)$, $i = 1, \ldots, N$ bestimmen, da sich $\ddot{u}_{app}(t)$ als lineare Funktion unter Hinzunahme der M_i beschreiben läßt:

$$\ddot{u}_{app}(t) = M_i \frac{t_{i+1} - t}{h_i} + M_{i+1} \frac{t - t_i}{h_i}, \quad t \in [t_i, t_{i+1}] \tag{4.73}$$

Definiert man aus Gründen der Übersichtlichkeit

$$\left.\begin{aligned}
\lambda_i &:= \frac{h_i}{h_{i-1} + h_i} \\
\mu_i &:= 1 - \lambda_i = \frac{h_{i-1}}{h_{i-1} + h_i} \\
d_i &:= \frac{6}{h_{i-1} + h_i} \left(\frac{u(t_{i+1}) - u(t_i)}{h_i} - \frac{u(t_i) - u(t_{i-1})}{h_{i-1}} \right)
\end{aligned}\right\} \quad i = 2, \ldots, N-1 \tag{4.74}$$

und

$$\begin{aligned}
\lambda_1 &:= 1, \qquad \mu_N := 1, \\
d_1 &:= \frac{6}{h_1} \left(\frac{u(t_2) - u(t_1)}{h_1} - \dot{u}_0 \right), \\
d_N &:= \frac{6}{h_{N-1}} \left(\dot{u}_f - \frac{u(t_N) - u(t_{N-1})}{h_{N-1}} \right)
\end{aligned} \tag{4.75}$$

so ergeben sich die M_i aus dem Gleichungssystem in Matrixschreibweise

$$\begin{bmatrix}
2 & \lambda_1 & & & & \\
\mu_2 & 2 & \lambda_2 & & & 0 \\
& \mu_3 & 2 & \lambda_3 & & \\
& & \ddots & \ddots & \ddots & \\
0 & & & \mu_{N-1} & 2 & \lambda_{N-1} \\
& & & & \mu_N & 2
\end{bmatrix}
\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{N-1} \\ M_N \end{bmatrix}
\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix} \tag{4.76}$$

Eine Schranke für den Approximationsfehler liefert der

Satz 4.7 *Sei $u(t) \in C^4([t_0, t_f])$ und u_{app} nach der oben genannten Vorschrift konstruiert, dann wird durch*

$$|u(t) - u_{app}(t)| \leq \frac{5}{384} \max_{\tau \in [t_0, t_f]} |u^{(4)}(\tau)| (\max_k \; h_k)^4 \quad \forall\, t \in [t_0, t_f] \qquad (4.77)$$

eine obere Fehlerschranke bei der kubischen Interpolation angegeben.

Die hier benannten Sätze zur Fehlerschranke der verschiedenen Approximationsvorschriften werden zum Beispiel in [Conte 1980] bewiesen.

Der Wert der Trajektorie an jedem der diskreten Punkte wird dann durch die Lösung der Anfangswertaufgabe

$$\dot{x}(t) = f(x, u_{app}, t), \quad x(t_0) = \begin{pmatrix} x_0 \\ s \end{pmatrix} \qquad (4.78)$$

bestimmt, wobei ohne Einschränkung die fehlenden Anfangswerte s am Ende von $x(t_0)$ stehen.
Um die Fehlerordnung der Integration in etwa der der beschriebenen Interpolationen der Steuerungen anzugleichen, stehen die folgenden Einschrittmethoden zur Verfügung. Die Schrittweiten h_i werden dabei den gewählten Diskretisierungen angepaßt.

Approximation der Trajektorie durch Einschrittverfahren

Zu einer approximativ gegebenen Steuerung sind Einschrittverfahren hier gekennzeichnet durch folgende Bildungsvorschrift

$$x_{k+1} := x_k + h_k \eta(x_k, u_{app}, t_k), \quad k = 1, \ldots, N-1. \qquad (4.79)$$

Der Startwert x_1 ergibt sich aus den bekannten Anfangswerten $x(t_0) \in \mathbb{R}^{n-l}$ und einer Ergänzung von Optimierungsvariablen $s \in \mathbb{R}^l$ zu einem vollständigen Anfangswert

$$x_1 := \begin{pmatrix} x(t_0) \\ s \end{pmatrix} \in \mathbb{R}^n. \qquad (4.80)$$

Die Funktion $\eta(x_k, u_{app}, t_k)$ beschreibt die zugrundeliegende Methode, wie aus der bekannten Information (x_k, u_{app}, t_k) und der Schrittweite h_k der neue Näherungswert x_{k+1} zu berechnen ist.

Eulersche Polygonzugmethode
Im Fall der Methode von Euler ist

$$\begin{array}{rrcl} & \eta(x_k, u_{app}, t_k) & = & z_1 \\ \text{mit} & z_1 & := & f(x_k, u_{app}(t_k), t_k) \end{array} \qquad (4.81)$$

wobei ein Verfahren der Fehlerordnung $O(h)$ vorliegt.
Durch Anwendung einer Extrapolation gelangt man auf die

Verbesserte Polygonzugmethode von Euler[14]
Die Bildungsvorschrift für η lautet hier

$$\begin{array}{lrcl} & \eta(x_k, u_{app}, t_k) & = & z_2 \\ \text{mit} & z_2 & := & f(x_k + \frac{h_k}{2} z_1, u_{app}(t_k + \frac{h_k}{2}), t_k + \frac{h_k}{2}) \\ \text{und} & & & z_1 \quad \text{wie in (4.81)} \end{array} \qquad (4.82)$$

und besitzt die Fehlerordnung $O(h^2)$.

Das so beschriebene zweistufige Verfahren kann in einfacher Weise zu einem Runge-Kutta-Verfahren der Fehlerordnung $O(h^4)$ ausgebaut werden.
Das klassische Runge-Kutta-Verfahren ist jedoch nur bedingt geeignet, wenn ohne viel größeren Mehraufwand auch Verfahren noch höherer Fehlerordnung hieraus konstruiert werden sollen. Ein ausbaufähigeres Runge-Kutta-Verfahren fand *England*:

Runge-Kutta-Verfahren der Ordnung 4 nach England
Durch

$$\begin{array}{lrcl} & \eta(x_k, u_{app}, t_k) & = & \frac{1}{6}(z_1 + 4z_3 + z_4) \\ \text{mit} & z_3 & := & f(x_k + \frac{h_k}{4}(z_1 + z_2), u_{app}(t_k + \frac{h_k}{2}), t_k + \frac{h_k}{2}) \\ & z_4 & := & f(x_k - h_k(z_2 + 2z_3), u_{app}(t_{k+1}), t_{k+1}) \\ \text{und} & & & z_1,\ z_2 \quad \text{wie in (4.81), (4.82)} \end{array} \qquad (4.83)$$

wird ein vierstufiges Runge-Kutta-Verfahren der Fehlerordnung $O(h^4)$ beschrieben, zu dem eine Schätzung des lokalen Diskretisierungsfehlers mit einem Verfahren der Fehlerordnung $O(h^5)$ erfolgen kann. Dazu ist allerdings ein sechsstufiges Verfahren notwendig, denn *Butcher* hat gezeigt, daß die erreichbare Fehlerordnung p eines expliziten m-stufigen Runge-Kutta-Verfahrens gemäß Tabelle 4.2 gegeben ist.

m	1	2	3	4	5	6	7	8	9
p	1	2	3	4	4	5	6	6	7

Tabelle 4.2: Maximale Fehlerordnung von expliziten Runge-Kutta-Verfahren

[14]Hierbei handelt es sich um ein explizites zweistufiges Runge-Kutta-Verfahren der Fehlerordnung $O(h^2)$.

Runge-Kutta-Verfahren der Ordnung 5 nach England
Die Runge-Kutta-Methode vierter Ordnung von England kann zu einem sechsstufigen Runge-Kutta-Verfahren der Fehlerordnung $O(h^5)$ durch

$$\begin{array}{rrcl} & \eta(x_k, u_{app}, t_k) & = & \frac{1}{336}(14z_1 + 35z_4 + 162z_5 + 125z_6) \\ \text{mit} & z_5 & := & f(\hat{z}_5, u_{app}(t_k + \frac{2h_k}{3}), t_k + \frac{2h_k}{3}) \\ & z_6 & := & f(\hat{z}_6, u_{app}(t_{k+1} + \frac{h_k}{5}), t_k + \frac{h_k}{5}) \\ & \hat{z}_5 & := & x_k + \frac{h_k}{27}(7z_1 + 10z_2 + 4z_4) \\ & \hat{z}_6 & := & x_k + \frac{h_k}{625}(28z_1 - 125z_2 + 546z_3 + 54z_4 - 378z_5), \\ \text{und} & & & z_1,\ z_2,\ z_3,\ z_4 \quad \text{wie in (4.81), (4.82) bzw. (4.83)} \end{array} \tag{4.84}$$

erweitert werden.

Bis auf die Polygonzugmethode nach Euler, die aber leider nur die Fehlerordnung $O(h)$ besitzt, benötigen alle beschriebenen Integrationsroutinen eine Schätzung der Steuerung an Punkten im Inneren der Teilintervalle $[t_i, t_{i+1}]$. Daher soll noch ergänzend ein Verfahren der Fehlerordnung $O(h^2)$ vorgestellt werden, welches ebenfalls auf eine Schätzung der Steuerung an Punkten im Inneren der Teilintervalle verzichtet.

Methode von Heun[15]
Bei der Methode von Heun handelt es sich ebenfalls um einen Repräsentanten eines expliziten zweistufigen Runge-Kutta-Verfahrens der Fehlerordnung $O(h^2)$. Hier ist

$$\begin{array}{rrcl} & \eta(x_k, u_{app}, t_k) & = & \frac{h_k}{2}(z_1 + z_7) \\ \text{mit} & z_7 & := & f(x_k + h_i k_1, u_{app}(t_{k+1}), t_{k+1}) \\ \text{und} & & & z_1 \quad \text{wie in (4.81)} \end{array} \tag{4.85}$$

Das Eulersche Polygonzugverfahren und das Verfahren nach Heun nehmen somit eine Sonderstellung ein, da zu einer gegebenen Schätzung der Steuerung an diskreten Punkten keine Approximation an weiteren Punkten benötigt wird. Interpolationsfehler der Steuerung treten daher nicht auf.
Desweiteren ist es sinnvoll, die anderen Einschrittverfahren mit Interpolationsverfahren gleicher Fehlerordnung zu koppeln. Da die obere Schranke des Interpolationsfehlers abhängig von dem Maximum gewisser Ableitungen auf dem gesamten Optimierungsintervall ist, andererseits eine Approximation der Steuerfunktion aber nur an gewissen Punkten im Inneren der Teilintervalle

[15] Die Methode von Heun wird auch als Prädiktor-Korrektor-Methode bezeichnet.

benötigt wird, besteht die berechtigte Hoffnung, daß die Kombination mit einem Integrationsverfahren höherer Ordnung zu besseren Ergebnissen führt. Mit der Routine NUDOCCCS liegt eine von mir umgesetzte effiziente Realisierung dieses Ansatzes vor, die überdies die Benutzung von Mehrzielknoten gestattet, um das Verfahren stabiler zu gestalten. Grundlage des resultierenden nichtlinearen Optimierungsproblems ist wiederum die Fortran-Routine OPRQP.
Tabelle B.1 gibt die in NUDOCCCS realisierten Kombinationen von Interpolations- und Integrationsverfahren, sowie einige weitere Informationen wider. Dabei bedeuten

ξ_1 Fehlerordnung des Integrationsverfahrens,
ξ_2 Fehlerordnung des Interpolationsverfahrens,
RSA Auswertungen der rechten Seite des Differentialgleichungssystems.

Nummer	Integr.Verf.	ξ_1	Intpol.Verf.	ξ_2	$\min(\xi_1, \xi_2)$	RSA
0	Poly.	$O(h)$	—	—	$O(h)$	1
1	Heun	$O(h^2)$	—	—	$O(h^2)$	2
2	verb.Poly.	$O(h^2)$	konstant	$O(h)$	$O(h)$	2
3	verb.Poly.	$O(h^2)$	linear	$O(h^2)$	$O(h^2)$	2
4	RK 4	$O(h^4)$	linear	$O(h^2)$	$O(h^2)$	4
5	RK 4	$O(h^4)$	kubisch	$O(h^4)$	$O(h^4)$	4
6	RK 5	$O(h^5)$	kubisch	$O(h^4)$	$O(h^4)$	6

Tabelle 4.3: Realisierte Kombinationen der Routine NUDOCCCS

Zusammenfassend ergibt sich also zum Beispiel im Falle der linearen Interpolation das nichtlineare Optimierungsproblem

$$\begin{array}{ll} \min F(\mathbf{Y}), & F : \mathbb{R}^{Nm+l} \longrightarrow \mathbb{R} \\ \text{bezogen auf} & \psi(x(t_f)) = 0 \\ & S(x(t), u_{app}(t), t) \leq 0 \end{array} \tag{4.86}$$

wobei $x(t)$ in geeigneter Weise durch die beschriebenen Integrationsverfahren berechnet wird.
Diese Art der Berechnung hat verschiedene Vorteile:

- Die Anzahl der Optimierungsvariablen bleibt relativ klein und ist unabhängig von der Dimension des Differentialgleichungssystems.
- Der Rechenaufwand für die numerische Integration wird gering gehalten.
- Die Anzahl der Nebenbedingungen bleibt relativ klein, da es keine dynamischen Nebenbedingungen des Zustandes gibt und die bekannten Anfangswerte direkt in die Integrationsroutine eingehen und nicht in das Optimierungsproblem.
- einfache Parallelisierbarkeit

Überlegungen wie in Kapitel 4.2.2 führen zu dem Ergebnis, daß Integrationsverfahren niedriger Ordnung stabiler sind, als die höherer Ordnung, so daß es oft zweckmäßig ist, erst eine Näherung der Trajektorie mit einer Methode niedriger Fehlerordnung zu berechnen, um diese dann als Startschätzung für Verfahren höherer Ordnung zu benutzen.

4.3.4 Spezielle Eigenschaften des direkten Verfahrens durch Problemsharing

Durch Hinzunahme von Mehrzielknoten und damit zusätzlichen Optimierungsvariablen und Nebenbedingungen beim direkten Verfahren durch Problemsharing, ergibt sich eine ähnliche Blockdiagonalstruktur der Jacobi- und der Hesse-Matrix, wie bei der direkten Mehrzielmethode.
Auf eine diesbezüglich eingehende Untersuchung wird aber an dieser Stelle zugunsten einer weiteren speziellen Eigenschaft verzichtet, die sich aus der Integration mit relativ großer Schrittweite ergibt, wenn die Ausgangslösung (ohne Mehrzielknoten) zu gegebenen Anfangswerten untersucht wird.
Für die folgenden Untersuchungen wird nur der Spezialfall eines optimalen Steuerprozesses mit $m = 1$, $n = 1$, $\bar{s} = 1$ und bekannten Anfangswerten betrachtet, der durch ein Einschrittverfahren mit linearer Interpolation der Steuerung gelöst wird.
Grundsätzlich lassen sich die so gefundenen Ergebnisse auf größere Steuerprozesse mit unbekannten Anfangswerten und anderen Interpolationsvorschriften erweitern.
Ein Großteil des Rechenaufwandes der numerischen Ermittlung der Lösung wir für die Berechnung des Gradienten, der Jacobi- und Hesse-Matrix benötigt. Bei genauerer Betrachtung dieser Ableitungen stellt man aber fest, daß sie abhängig von den Ableitungen der Trajektorie nach den diskreten Optimierungsvariablen sind. So gilt zum Beispiel für die Jacobi-Matrix

$$\frac{dS(x,u,t)}{du} = \frac{\partial S}{\partial x}\frac{dx}{du} + \frac{\partial S}{\partial u} \tag{4.87}$$

Durch den rekursiven Aufbau der Trajektorie $x_k := x(x_{k-1}, \ldots, x_1)$ bedingt, liegt die Hauptschwierigkeit bei der Berechnung dieser Ableitung in der Bestimmung der Ableitung für $\frac{dx}{du}$. Der folgende Satz nutzt aber gerade die rekursive Definition der Trajektorie für eine schnellere Bestimmung der Ableitung aus.

Satz 4.8 *Gegeben sei ein allgemeines Einschrittverfahren der Form* $x_{k+1} := x_k + h\eta_k$, $\eta_k := \eta(x_k, u_k, u_{k+1}, t_k, t_{k+1})$, *(lineare Interpolation)* $k = 1, \ldots, N-1$ *mit den bekannten Anfangswerten* $x_1 = x_0$ *und* $u_k := u(t_k)$ *(Optimierungsvektor der diskreten Steuerung) wie in Kapitel 4.3.3 beschrieben, mit einer äquidistanten Schrittweite* $h := t_{i+1} - t_i$. *Alle auftretenden Funktionen seien*

hinreichend oft differenzierbar.
Dann gilt:

$$
\begin{array}{lrcll}
a) & \dfrac{dx_j}{du_i} & = & 0 & \textit{für } j < i \\
b) & \dfrac{dx_j}{du_i} & = & h\dfrac{\partial \eta_{i-1}}{\partial u_i} & \textit{für } j = i \\
c) & \dfrac{dx_j}{du_i} & = & h\left[\dfrac{\partial \eta_i}{\partial u_i} + \dfrac{\partial \eta_{i-1}}{\partial u_i}\left(1 + h\dfrac{\partial \eta_i}{\partial x_i}\right)\right] & \textit{für } j = i+1 \\
d) & \dfrac{dx_j}{du_i} & = & h\prod_{k=i+1}^{j-1}\left(1 + h\dfrac{\partial \eta_k}{\partial x_k}\right)\left[\dfrac{\partial \eta_i}{\partial u_i} + \dfrac{\partial \eta_{i-1}}{\partial u_i}\left(1 + h\dfrac{\partial \eta_i}{\partial x_i}\right)\right] & \textit{für } j > i+1
\end{array}
\tag{4.88}
$$

Beweis: Durch Anwendung der Kettenregel der Differentiation zeigt man

$a)$ offensichtlich aufgrund der Definition von x_k

$b)$ $\dfrac{dx_j}{du_i} \overset{\text{j=i}}{=} \dfrac{dx_i}{du_i} = \dfrac{\partial x_{i-1}}{\partial u_i} + h\dfrac{\partial \eta_{i-1}}{\partial u_i} \overset{\text{a)}}{=} h\dfrac{\partial \eta_{i-1}}{\partial u_i}$

$c)$
$$
\begin{aligned}
\frac{dx_j}{du_i} &\overset{\text{j=i+1}}{=} \frac{dx_{i+1}}{du_i} = \frac{dx_i}{du_i} + h\left(\frac{\partial \eta_i}{\partial x_i}\frac{dx_i}{du_i} + \frac{\partial \eta_i}{\partial u_i}\right) \\
&= \frac{dx_i}{du_i}\left(1 + h\frac{\partial \eta_i}{\partial x_i}\right) + h\frac{\partial \eta_i}{\partial u_i} \overset{\text{b)}}{=} h\left[\frac{\partial \eta_i}{\partial u_i} + \frac{\partial \eta_{i-1}}{\partial u_i}\left(1 + h\frac{\partial \eta_i}{\partial x_i}\right)\right]
\end{aligned}
$$

$d)$ Induktion über $n := j - i$

I.A.: $n = 2$

$$
\begin{aligned}
\frac{dx_j}{du_i} &= \frac{dx_{i+2}}{du_i} = \frac{dx_{i+1}}{du_i} + h\frac{d\eta_{i+1}}{du_i} \\
&= \frac{dx_{i+1}}{du_i} + h\frac{\partial \eta_{i+1}}{\partial x_{i+1}}\frac{dx_{i+1}}{du_i} = \frac{dx_{i+1}}{du_i}\left(1 + h\frac{\partial \eta_{i+1}}{\partial x_{i+1}}\right) \\
&\overset{\text{c)}}{=} h\prod_{k=i+1}^{j-1}\left(1 + h\frac{\partial \eta_k}{\partial x_k}\right)\left[\frac{\partial \eta_i}{\partial u_i} + \frac{\partial \eta_{i-1}}{\partial u_i}\left(1 + h\frac{\partial \eta_i}{\partial x_i}\right)\right]
\end{aligned}
$$

I.V.: Sei die Behauptung für ein beliebiges $n \geq 2$ bewiesen

$$
\frac{dx_j}{du_i} = \frac{dx_{i+n}}{du_i} = h\prod_{k=i+1}^{i+n-1}\left(1 + h\frac{\partial \eta_k}{\partial x_k}\right)\left[\frac{\partial \eta_i}{\partial u_i} + \frac{\partial \eta_{i-1}}{\partial u_i}\left(1 + h\frac{\partial \eta_i}{\partial x_i}\right)\right]
$$

I.S.: $n \longmapsto n+1$

$$
\begin{aligned}
\frac{dx_j}{du_i} &= \frac{dx_{i+n+1}}{du_i} = \frac{dx_{i+n}}{du_i} + h\frac{d\eta_{i+n}}{du_i} \\
&= \frac{dx_{i+n}}{du_i} + h\frac{\partial \eta_{i+n}}{\partial x_{i+n}}\frac{dx_{i+n}}{du_i} = \frac{dx_{i+n}}{du_i}\left(1 + h\frac{\partial \eta_{i+n}}{\partial x_{i+n}}\right) \\
&\overset{\text{I.V.}}{=} h\prod_{k=i+1}^{i+n-1}\left(1 + h\frac{\partial \eta_k}{\partial x_k}\right)\left[\frac{\partial \eta_i}{\partial u_i} + \frac{\partial \eta_{i-1}}{\partial u_i}\left(1 + h\frac{\partial \eta_i}{\partial x_i}\right)\right]\left(1 + h\frac{\partial \eta_{i+n}}{\partial x_{i+n}}\right) \\
&= h\prod_{k=i+1}^{i+n}\left(1 + h\frac{\partial \eta_k}{\partial x_k}\right)\left[\frac{\partial \eta_i}{\partial u_i} + \frac{\partial \eta_{i-1}}{\partial u_i}\left(1 + h\frac{\partial \eta_i}{\partial x_i}\right)\right] \qquad \square
\end{aligned}
$$

In ähnlicher Weise lassen sich auch Aussagen über andere Interpolationsarten finden.
Durch Anwendung der Kettenregel der Differentiation und Satz 4.8 a) erhält man unmittelbar die

Folgerung 4.9 *Die Jacobimatrix der Nebenbedingungen* $S_i := S(x_i, u_i, t_i)$ *hat eine untere Dreiecksstruktur*

$$\begin{pmatrix} \frac{dS_1}{du_1} & & & \\ \frac{dS_2}{du_1} & \frac{dS_2}{du_2} & & 0 \\ \vdots & \vdots & \ddots & \\ \frac{dS_N}{du_1} & \frac{dS_N}{du_2} & \cdots & \frac{dS_N}{du_N} \end{pmatrix} \tag{4.89}$$

Aus Satz 4.8 folgt ebenfalls unmittelbar nach d)

Folgerung 4.10 $\frac{dx_j}{du_i}$, $j > i+1$ *ist unabhängig von* x_k, $k < i-1$, *d.h. bei der Berechnung der Ableitungen kann man von der bekannten Lösung* x_k *starten.*

Bei der praktischen Realisierung der beiden Folgerungen halbiert sich in etwa der jeweilige Rechenaufwand. Trotzdem bleibt die Anzahl der Auswertungen der rechten Seite des Differentialgleichungssystems in der Größenordnung $O(N^2)$ bei der Bestimmung der numerischen Ableitungen.
Durch weitere Ausnutzung der rekursiven Eigenschaften und von Satz 4.8 kann der Rechenaufwand aber weiter verringert werden, wenn die Werte für

$$\left(1 + h\frac{\partial \eta_i}{\partial x_i}\right), \quad \frac{\partial \eta_i}{\partial u_i} \quad \text{und} \quad \frac{\partial \eta_{i-1}}{\partial u_i} \tag{4.90}$$

nur einmalig berechnet und immer wieder ausgenutzt werden.
Ebenso läßt sich der Aufwand der Multiplikationen (in der Produktformel) und Additionen weiter verringern.
Insgesamt führt dies auf einen Aufwand von $O(N)$ Aufrufen der rechten Seite des Differentialgleichungssystems bei der Bestimmung der numerischen Ableitungen, also gerade bei hohen Diskretisierungsraten eine enorme Verbesserung.
Die Ergebnisse lassen sich weiter verbessern, da man teilweise die Ableitungen analytisch bestimmen kann.

4.4 Allgemeine Modifikationstechniken der beschriebenen Diskretisierungsmethoden

In diesem Abschnitt sollen einige Modifikationstechniken zur Verbesserung der beschrieben direkten Verfahren aufgezeigt werden.

4.4.1 Variationsmöglichkeiten der Zustands- und Steuerbeschränkungen

Beschränkungen in Zustands- oder Steuerungsabhängigkeit spielen gerade bei den praktischen Anwendungen eine entscheidende Rolle und haben somit, nicht zuletzt wegen der oftmals nur umständlichen Realisierung auf indirektem Wege, einen entscheidenden Einfluß bei der praktischen Umsetzung der direkten Verfahren.
In diesem Abschnitt sollen vier Varianten zur Aufnahme von Beschränkungen in das resultierende nichtlineare Optimierungsproblem kurz vorgestellt und miteinander verglichen werden. Dabei kommen ihnen durch ihre unterschiedlichen Eigenschaften auch verschiedene Bedeutungen bei der praktischen Realisierung zu.

Variante 1:

Bei der ersten Variante werden die Beschränkungen $S(x, u, t) \leq 0$ durch die Einführung zusätzlicher Differentialgleichungen in Randwerte transformiert:

$$\left.\begin{array}{rcl} \dot{x}_{n+i} & = & \max(0, S(x, u, t)) \\ x_{n+i}(t_0) & = & 0 \\ x_{n+i}(t_f) & = & 0 \end{array}\right\} i = 1, \ldots, \bar{s} \qquad (4.91)$$

Dadurch weist diese Variate verschiedene Vorteile auf:

1. Die Beschränkungen werden entlang der Trajektorie nicht verletzt.
2. Die Anzahl aller Nebenbedingungen im nichtlinearen Optimierungsproblem bleibt klein.

Nachteile sind:

1. Die Integranden von (4.91) sind in der Regel an Ein- und Austrittspunkten nicht differenzierbar, d.h. wenn ein solcher Punkt nicht direkt angesteuert wird, sinkt dort die Approximationsordnung des Verfahrens, was eine ungenauere Lösung zur Folge hätte.
2. Diese Variante eignet sich nur bei Verfahren mit hoher Fehlerordnung der Integration, wie zum Beispiel bei der direkten Mehrzielmethode, weil bei den anderen Verfahren die Lösung nur an diskreten Punkten vorliegt. Folge: Vorteil 1 wird aufgehoben und bei Verfahren, die durch Diskretisierung der Steuer- und Zustandsfunktionen charakterisiert sind, ist dieser Ansatz sogar 'unsinnig', da durch die zusätzlichen Differentialgleichungen die Anzahl der Optimierungsvariablen und der dynamischen Nebenbedingungen wachsen. Vorteil 2 entfällt also auch, vielmehr steigt sogar der Rechenaufwand durch die hinzukommenden Differentialgleichungen.

Variante 2:

Eine zweite Variante ergibt sich durch die direkte Auswertung der Nebenbedingungen an den diskreten Zeitpunkten t_i, $i = 1, \ldots, N$:

$$S_j(x(t_i), u(t_i), t_i) \leq 0, \quad i = 1, \ldots, N, \ j = 1, \ldots, \bar{s} \tag{4.92}$$

Vorteile dieser Variante sind:

1. Es werden keine zusätzlichen dynamischen Variablen eingeführt.
2. Das Problem der Nichtdifferenzierbarkeit entfällt durch Punkt 1.
3. Diese Art der Behandlung der Nebenbedingungen ist für alle beschriebenen direkten Verfahren realisierbar.

Die aus dieser Variante resultierenden Nachteile sind:

1. Innerhalb der Teilintervalle $[t_i, t_{i+1}]$, $i = 1, \ldots, N-1$ muß die Beschränkung nicht zwangsweise erfüllt sein[16].
2. Die Anzahl der 'algebraischen' Nebenbedingungen des nichtlinearen Optimierungsproblems wächst.

Eine Abmilderung des ersten Nachteils kann erreicht werden, wenn man die Intervallgrenzen gezielt an die Stelle der Ein- bzw. Austrittspunkte setzt. Möglichkeiten zur Bestimmung unbekannter Ein- bzw- Austrittspunkte und somit auch einer Verbesserung der Approximationsgüte werden in den Abschnitten 4.4.2 und 4.4.3 erläutert.
Da bei der Integration der Trajektorie die Bedingungen (4.92) beiläufig anfallen, wird Nachteil 2 durch Vorteil 1 bei weitem aufgehoben. Desweiteren sind aktive Beschränkungen im Optimierungsproblem durchaus von Vorteil, da durch sie die Freiheitsgrade des Problems weiter eingeschränkt werden.

Variante 3:

Eine dritte Variante ergibt sich durch die Zusammenfassung der diskreten Nebenbedingungen (4.92).
Man wähle für $w \leq N$, $w \in \mathbb{N}$

$$1 = M_1 < M_2 < \ldots < M_w = N, \quad M_j \in \{1, \ldots, N\}, \ j = 1, \ldots, w \tag{4.93}$$

eine grobere Unterteilung der diskreten Punkte

$$t_0 = t_1 = t_{M_1} < \ldots < t_{M_j-1} < t_{M_j} < t_{M_j+1} < \ldots < t_{M_w} = t_N = t_f, \tag{4.94}$$

[16]Liegen die Ein- bzw. Austrittspunkte innerhalb der Teilintervalle, so führt dies insbesondere zum Verlust der Approximationsgüte.

für $j = 2, \ldots, M_w - 1$. Die allgemeinen Beschränkungen werden dann zum Beispiel durch die Aufnahme der Nebenbedingungen

$$\sum_{j=M_k}^{M_{k+1}-1} |\max(0, S_i(x(t_j), u(t_j), t_j))| = 0, \quad i = 1, \ldots, \bar{s},\ k = 1, \ldots, w-2$$

und

$$\sum_{j=M_{w-1}}^{M_w-1} |\max(0, S_i(x(t_j), u(t_j), t_j))| = 0, \quad i = 1, \ldots, \bar{s} \tag{4.95}$$

oder

$$\sum_{j=M_k}^{M_{k+1}-1} (\max(0, S_i(x(t_j), u(t_j), t_j)))^2 = 0, \quad i = 1, \ldots, \bar{s},\ k = 1, \ldots, w-2$$

und

$$\sum_{j=M_{w-1}}^{M_w-1} (\max(0, S_i(x(t_j), u(t_j), t_j)))^2 = 0, \quad i = 1, \ldots, \bar{s} \tag{4.96}$$

eingehalten.
Im einfachsten Fall für $w = 2$ und $M_w = N$ erhielte man somit

$$\sum_{j=1}^{N} |\max(0, S_i(x(t_j), u(t_j), t_j))| = 0, \quad i = 1, \ldots, \bar{s} \tag{4.97}$$

oder

$$\sum_{j=1}^{N} (\max(0, S_i(x(t_j), u(t_j), t_j)))^2 = 0, \quad i = 1, \ldots, \bar{s} \tag{4.98}$$

Vorteile dieser Variante sind

1. Es werden keine zusätzlichen dynamischen Variablen eingeführt.
2. Die Anzahl aller Nebenbedingungen bleibt klein.
3. Diese Variante ist für alle direkten Verfahren realisierbar.

Die sich ergebenden Nachteile sind

1. Innerhalb der Teilintervalle $[t_i, t_{i+1}]$, $i = 1, \ldots, N-1$ muß die Beschränkung nicht zwangsweise erfüllt sein.
2. Die Nebenbedingungen könnten bzgl. u nicht differenzierbar sein[17].

Eine Abmilderung von Nachteil 1 kann analog zur zweiten Variante erreicht werden. Ebenfalls fällt bei der Bestimmung der Trajektorie bereits ein Großteil der benötigten Informationen an.

[17] Die nichtlinearen Optimierungsprogramme gehen in der Regel von der Differenzierbarkeit der auftretenden Beschränkungen aus.

Variante 4:

Die Anzahl der Nebenbedingungen läßt sich bekanntermaßen stark reduzieren. Es soll nun eine Möglichkeit aufgezeigt werden, welche schließlich völlig auf Nebenbedingungen des nichtlinearen Optimierungsproblems verzicht. Sie ergibt sich aus der Kombination der beschriebenen Verfahren mit einer sogenannten 'Penalty-Funktion', bei der die Beschränkungen in geeigneter Weise mit der Zielfunktion verknüpft werden.
Nach einer Diskretisierung des optimalen Steuerprozesses mit Beschränkungen liege das folgende allgemeine nichtlineare Optimierungsproblem mit Gleichungs[18]- und Ungleichungsnebenbedingungen[19] vor:

$$\begin{array}{rlcll} \min & f(x,u,t) & & & \\ \text{unter} & S_i(x,u,t) & = & 0 & i=1,\ldots,k \\ & S_i(x,u,t) & \leq & 0 & i=k+1,\ldots,q \end{array} \tag{4.99}$$

Durch die Transformation dieses beschränkten Problems in das unbeschränkte Problem

$$\min f(x,u,t) + \frac{1}{R}\sum_{i=1}^{k}(S_i(x,u,t))^2 + \frac{1}{R}\sum_{i=k+1}^{q}(\max(0,S_i(x,u,t))^2 \tag{4.100}$$

mit einem Fehlerparameter $R \in \mathbb{R}$, der eine Art 'Bewertungsfunktion' der Beschränkungen darstellt, werden alle Beschränkungen von (4.99) vermieden.

Je nach Kombination mit den anderen Varianten übertragen sich die Vor- und Nachteile der einzelnen Methoden. Ein zusätzlicher Vorteil liegt in der Möglichkeit, auch Routinen zur nichtlinearen Optimierung benutzen zu können, die keine Nebenbedingungen zulassen.
Bei sämtlichen hier realisierten direkten Verfahren wurde Variante 2 gewählt, da durch die zusätzlichen aktiven Beschränkungen die Freiheitsgrade des Systems reduziert werden und in der Regel eine optimale Lösung schneller gefunden wird. Einige spezielle Techniken, wie z.B. die in Abschnitt 4.4.4 beschriebene Schlauchtechnik profitieren sogar von der Hinzunahme zusätzlicher Beschränkungen.

4.4.2 Optimale Diskretisierungsschrittweite

Alle bisher dargelegten direkten Verfahren gingen von einem nicht zwangsweise äquidistanten aber *festen* Gitter aus, dessen einziger Optimierungsfreiheitsgrad eine möglicherweise freie Endzeit ist.

[18]z.B. Randbedingungen oder aus den Varianten 1-3 entstanden
[19]z.B. diskretisierte Nebenbedingungen

Will man aber qualitative Aussagen über die Güte eines Verfahrens machen, so können durch etwaige auftretende Schaltpunkte (z.B. durch Beschränkungen) die Ergebnisse negativ verfälscht werden. Speziell bei der Analyse der Schaltstruktur ist es häufig hilfreich, durch variable Diskretisierung[20] zusätzliche Freiheitsgrade zu erlauben.
Unumgänglich ist es dann, zusätzliche Optimierungsvariablen für die dann variablen Zeitpunkte zuzulassen. Desweitern empfiehlt sich die Einführung einer eigenen Gitterstruktur für jede Komponente der Steuerung, um den individuellen Anforderungen der einzelnen Steuerungen gerecht zu werden:

$$\begin{array}{ccccccc} t_0 & = & t_1^{(1)} & < \ldots < & t_N^{(1)} & = & t_f \\ \vdots & & \vdots & & \vdots & & \\ t_0 & = & t_1^{(m)} & < \ldots < & t_N^{(m)} & = & t_f \end{array} \tag{4.101}$$

Diese beziehen sich jeweils auf die Steuerungen $u_i,\ i = 1, \ldots, m$.
Der nichtlineare Optimierungsvektor ergibt sich damit zu:

$$\mathbf{Y} = (u_1(t_1^{(1)}), \ldots, u_m(t_N^{(m)}), t_2^{(1)}, \ldots, t_{N-1}^{(m)}, t_N) \tag{4.102}$$

Die Anzahl der Optimierungsvariablen hat sich somit fast verdoppelt, was unweigerlich einen höheren Rechenaufwand nach sich zieht. Um diesen aber dennoch gering zu halten, wird bei vorgegebenen Schranken $\varepsilon^{(i)} \in \mathbb{R}$ und $\delta^{(i)} \in \mathbb{R}$ in [Kraft 1985] folgende Strategie zur Lösungsfindung vorgeschlagen:

1. Löse das Optimierungsproblem mit den bisher bekannten Mitteln auf einem festen Gitter und dem entsprechend reduzierten Optimierungsvektor.

2. Benutze die so gefundene Lösung als Abschätzung für das Optimierungsproblem mit variablen Gitter (4.101) und Optimierungsvektor (4.102).

3. Füge zusätzliche Knoten $t_k^{(i)},\ t_j^{(i)} < t_k^{(i)} < t_{j+1}^{(i)}$ ein, falls

$$\begin{array}{l} t_{j+1}^{(i)} - t_j^{(i)} < \varepsilon^{(i)} \\ \text{und} \\ \left\| u_i(t_{j+1}^{(i)}) - u_i(t_j^{(i)}) \right\| > \delta^{(i)} \end{array} \tag{4.103}$$

4. Eliminiere Knoten $t_{j+1}^{(i)}$, falls

$$\begin{array}{l} t_{j+1}^{(i)} - t_j^{(i)} < \varepsilon^{(i)} \\ \text{und} \\ \left\| u_i(t_{j+1}^{(i)}) - u_i(t_j^{(i)}) \right\| \leq \delta^{(i)} \end{array} \tag{4.104}$$

[20] durch Variation der Lage der diskreten Zeitpunkte im zulässigen Intervall

Bemerkung 4.11 *Die so beschriebene Lösungsstrategie erinnert stark an die optimale Schrittweitensteuerung bei Runge-Kutta-Fehlberg-Verfahren.*

[Horn 1989] empfiehlt, alle oder auch nur einige der Gitterpunkte variabel zu gestalten, ohne die Hinzunahme weiterer diskreter Zeitpunkte und damit weiterer Optimierungsvariablen zuzulassen. Vorzugsweise werden die Minima und Maxima der Steuerfunktionen variabel gehalten.

In [Betts 1990] wird eine Möglichkeit vorgeschlagen, das Gitter neu zu definieren, indem der Fehler $\delta^{(i)}$ nicht wie hier mit einer Funktion der Steuerung verglichen wird, sondern mit dem Fehler bei der Integration durch Routinen unterschiedlicher Fehlerordnungen.

Je nach Schwerpunkt der Untersuchungen stellten die Autoren verschiedene Vorteile der variablen Gitterstruktur fest:

- Minima und Maxima der Steuerfunktionen werden besser eingehalten.
- Die berechnete Lösung der Trajektorie, Steuerung und des Zielfunktionals sind genauer.
- Schaltpunkte, durch Sprünge in den Steuerungen charakterisiert, können genauer angegeben werden.
- Ein- bzw. Austrittspunkte werden genauer erreicht.
- Die Anzahl der diskreten Punkte zu einem vorgegebenen Fehler des Zielfunktionals wird optimiert.
- Aussagen über die Approximationsgüte können leichter gemacht werden.

Schwerwiegende Nachteile gegenüber den Methoden mit festem Gitter konnten, außer durch einen Mehraufwand an Rechenzeit, nicht festgestellt werden. Zusammenfassend läßt sich also sagen, daß die Einführung von variablen Gitterpunkten eine durchaus sinnvolle Erweiterung der Ausgangslösungen darstellt, so daß zu erwarten ist, daß die Verwendung von höher entwickelten Strategien (als die hier vorgestellten) zur variablen Gitterpunktsgestaltung und/oder Extrapolationsverfahren zu noch besseren Ergebnissen führt.

4.4.3 Eine 'sichere' Methode zur genaueren Bestimmung von speziellen Schaltpunkten

In diesem Abschnitt soll eine neue Methode zur genaueren Bestimmung von Schaltpunkten, die durch Ein- bzw. Austrittspunkte definiert sind, also an

Punkten, an denen eine aktive Nebenbedingung in eine inaktive (oder umgekehrt) übergeht, vorgeschlagen werden, ohne daß zusätzliche Informationen als die vorhandenen benutzt werden.
Eine häufig auftretende Situation, in der die in Abschnitt 4.4.2 beschriebenen Verfahren versagen[21], ergibt sich aus der Tatsache, daß eine Verschiebung der Gitterpunkte nur durch eine Reduzierung des Minimierungsfunktionals gerechtfertigt ist. Das Zielfunktional ist also einziges 'Bewertungskriterium'. Oft ist aufgrund der Diskretisierung des optimalen Steuerprozesses und der verwendeten Integrations- und Interpolationsverfahren das berechnete Minimum des resultierenden nichtlinearen Optimierungsproblems kleiner, als die tatsächliche optimale Lösung. Diskretisierungsverfahren mit variablen Gitterpunkten kommen also im Laufe ihrer Berechnungen ohne Hinzunahme weiterer Gitterpunkte mindestens auf die ursprünglich berechnete Lösung zurück. Insbesondere kann die Festlegung der definierten Schranken $\varepsilon^{(i)}$ und $\delta^{(i)}$ zu weiteren Problemen (z.B. Rechnerkapazität) führen.
Der im folgenden beschriebene Ansatz erlaubt mit einfachen Mitteln eine genauere Bestimmung von Schaltpunkten, die durch Ein- bzw. Austrittspunkte zu gegebenen Steuer- oder Zustandsbeschränkungen definiert sind, ohne Erweiterung des Optimierungsvektors um variable Gitterpunkte oder unter Hinzunahme zusätzlicher Informationen.
Durch folgende Strategie läßt sich der Ansatz beschreiben:

1. Löse das optimale Steuerproblem mit den gegebenen Verfahren.

2. Bestimme die diskreten Punkte t_i, t_{i+1}, an denen die Nebenbedingungen in t_i aktiv und in t_{i+1} inaktiv (oder umgekehrt) sind und setze $\hat{t}_k := t_i$ und $\tilde{t}_k := t_{i+1}$.

3. Führe neue Gitterpunkte $t_k := (\hat{t}_k + \tilde{t}_k)/2$ und neue Optimierungsvariablen u_k ein.

4. Löse das neue Optimierungsproblem einschließlich dem Gitterpunkt t_k.

5. Wenn $t_k - \hat{t}_k \geq \varepsilon \in \mathbb{R}$ und die Beschränkung in t_k aktiv ist, so setze $\hat{t}_k := t_k$, $t_k := (\hat{t}_k + \tilde{t}_k)/2$, ansonsten setze $\tilde{t}_k := t_k$, $t_k := (\hat{t}_k + \tilde{t}_k)/2$ (bei einem Übergang von einer inaktiven zu einer aktiven Nebenbedingung verfahre man analog) und fahre mit 4. fort.

Die so gefundenen Werte t_k beschreiben eine Näherung für eventuelle Schaltpunkte in Abhängigkeit von ε und dem resultierenden Optimierungsproblem. Gilt für zwei Werte t_k und t_{k+1} die Bedingung $t_{k+1} - t_k \leq 2\varepsilon$, so handelt es sich vermutlich um einen Berührpunkt.

[21] Versagen in dem Sinne, daß eine Verbesserung der Lösung durch ein variables Gitter nicht erreicht werden kann.

In jedem Schritt wird die Genauigkeit der t_k bzgl. des Optimierungsproblems verdoppelt, aber auch Realisierungen mit einer schnelleren Annäherung sind denkbar.
Vorteile dieses Ansatzes sind:

1. Schaltpunkte können genauer angegeben werden.

2. Die berechneten Lösungen für die Trajektorie, die Steuerung und das Zielfunktional sind genauer.

3. Aussagen über die Approximationsgüte des Verfahrens können leichter gemacht werden.

4. Es werden nur wenige zusätzliche Optimierungsvariablen benötigt.

5. Verbesserung des Zielfunktionals durch Vergrößerung des Wertes ist möglich.

6. Ein Großteil der alten Informationen (z.B. Hesse-Matrix, Jacobi-Matrix, Gradient) kann unverändert als Startschätzung übernommen werden.

Motiviert wird dieser Ansatz weiterhin dadurch, daß die bei allen beschriebenen direkten Verfahren benutzte Routine OPRQP einen Vektor mit sich führt, der angibt, welche Nebenbedingungen aktiv bzw. inaktiv sind.
Will man zusätzlich auch Sprünge in der Steuerung zulassen, so muß eine weitere Steuervariable pro Schaltpunkt zugelassen werden. Außerdem müssen die Routinen zur Interpolation der Steuerung bzw. der Integration der Trajektorie dann den neuen Problemanforderungen angepaßt werden.
Eine praktische Realisierung dieses Ansatzes liegt für die Routine NUDOCCCS vor.

4.4.4 Eine spezielle 'Schlauchtechnik'

Um eine qualitativ hochwertige Lösung des optimalen Steuerprozesses zu bekommen, wird es notwendig sein, eine hohe Diskretisierungsrate zu verwenden.
Da der Lösungsaufwand der Optimierungsaufgabe aber nichtlinear (aller Erfahrung nach mindestens quadratisch) mit der Zahl der freien Parameter steigt, wird man die folgende Strategie anwenden wollen:

1. Löse den optimalen Steuerprozeß mit einer niedrigen Diskretisierungsrate.

2. Benutze die so gefundene Lösung als Startschätzung für den optimalen Steuerprozeß mit einer höheren Diskretisierungsrate.

Dieser Vorgang kann bei Bedarf mehrfach wiederholt werden. Dabei kann z.B. die Schätzung der neuen Steuervariablen bei einer Halbierung der Gitterintervalle durch die Linearinterpolierende im Intervallmittelpunkt genommen werden.
Außerdem empfiehlt sich eine Einschränkung der Freiheitsgrade des Problems durch Hinzunahme künstlicher Steuer- oder Zustandsbeschränkungen. Eine Möglichkeit soll nachstehend für direkte Verfahren, die nur die Steuerfunktionen diskretisieren, kurz beschrieben werden. Bei Verfahren, die auch den Zustand diskretisieren oder bei der Idee der optimalen Diskretisierungsschrittweite, läßt sich dieser Ansatz in analoger Weise formulieren.
Strategie:

1. Finde eine Lösung bei niedriger Diskretisierungsrate für $u(t_i),\ i = 1, \ldots, N$ und speichere sie in $\hat{u}(t_i)$.

2. Verdopple die Anzahl der Optimierungintervalle und interpoliere die neuen Steuerpunkte in den Intervallmittelpunkten in geeigneter Weise.

3. Führe zusätzliche Beschränkungen $(\hat{u} + \varepsilon) - u \geq 0$ und $u - (\hat{u} - \delta) \geq 0$ ein, wobei die Intervallmittelpunkte auch hier in geeigneter Weise durch einen interpolierten Wert ersetzt werden. (ε und δ beschreiben Genauigkeitsvektoren).

4. Finde eine Lösung der $u(t_i),\ i = 1, \ldots, 2N - 1$ (Halbierung der Intervallänge) des erweiterten Optimierungsproblems mit den zusätzlichen Beschränkungen.

5. Vergrößere die Werte der ε_i bzw. δ_i an den diskreten Gitterpunkten, wenn die zusätzlichen Beschränkungen dort aktiv sind, reduziere ε_i bzw. δ_i an den anderen diskreten Punkten entsprechend.

Abbildung 4.9 verdeutlicht noch einmal die Idee des Verfahrens und erklärt auch seine Namensgebung.

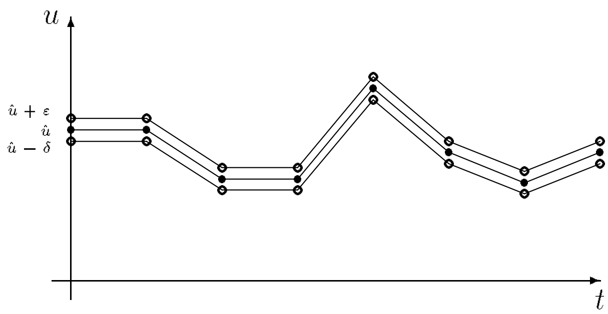


Abbildung 4.9: Graphische Übersicht einer speziellen Schlauchtechnik

Kapitel 5

Anwendungsbeispiele und deren Auswertung

5.1 Grundlagen/Voraussetzungen

In diesem Kapitel sollen verschiedene optimale Steuerprozesse ausgewertet und die direkten Verfahren bzgl. ihrer numerischen Ergebnisse miteinander verglichen werden. Dazu sind jedoch einige Voraussetzungen und Definitionen notwendig.

Voraussetzung 5.1

1. *Alle Eingangsparameter der in allen direkten Verfahren benutzten Routine OPRQP werden identisch gewählt.*
2. *Die Startschätzungen der direkten Verfahren werden identisch gewählt.*
3. *Alle direkten Verfahren lösen auf dem gleichen äquidistanten Gitter.*
4. *Alle optimalen Steuerprozesse werden in ein Mayer-Problem mit fester Endzeit transformiert.*

Wegen Punkt 3 definiert man durch äquidistante Zerlegung des Intervalls $[t_0, t_f]$ mit Schrittweite $h_{\hat{n}}$ folgende Gitter:

Definition 5.2 (Gitter der Stufe $\hat{n}$) *Sei $\mathcal{D} = [t_0, t_f] \in \mathbb{R}$ mit $-\infty < t_0 < t_f < \infty$, $n_0 \in \mathbb{N}$ fest gewählt, $l_{\hat{n}} := n_0 \cdot 2^{\hat{n}}$ und $h_{\hat{n}} := \frac{t_f - t_0}{l_{\hat{n}}}$, $\hat{n} \in \mathbb{N}_0$; dann wird die Menge*

$$\begin{aligned} \mathcal{D}_{\hat{n}} &:= \{t_0, t_0 + h_{\hat{n}}, \ldots, t_f - h_{\hat{n}}, t_f\} \\ &= \{y_j \in \mathbb{R} \mid y_j = t_0 + j \cdot h_{\hat{n}},\ j = 0, 1, \ldots, l_{\hat{n}}\} \end{aligned}$$

als Gitter der Stufe $\hat{n}$ auf $\mathcal{D}$ bezeichnet.

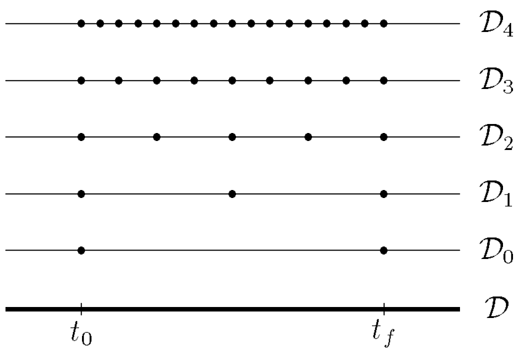


Abbildung 5.1: Gittersystem $\mathcal{D}_k$

Um die numerischen Ergebnisse der verschiedenen direkten Verfahren miteinander vergleichen zu können benötigt man

Definition 5.3 (globaler Diskretisationsfehler) *Als globaler Diskretisationsfehler (Diskretisierungsfehler) G_k des Zielfunktionals an der Stelle $t_k \in [t_0, t_f]$ bezeichnet man die Differenz*

$$G_k := g(x(t_k)) - g(x_k),$$

wobei $x(t_k)$ die exakte Lösung und x_k die Lösung der direkten Verfahren an den Punkten t_k beschreibt.

Mit dem globalen Diskretisationsfehler definiert man nun die folgenden normierten Fehler:

ζ_0 : Fehler der Routine NUDOCCCS mit `nart=0`,
ζ_1 : Fehler der Routine NUDOCCCS mit `nart=1`,
ζ_2 : Fehler der Routine NUDOCCCS mit `nart=2`,
ζ_3 : Fehler der Routine NUDOCCCS mit `nart=3`,
ζ_4 : Fehler der Routine NUDOCCCS mit `nart=4`,
ζ_5 : Fehler der Routine NUDOCCCS mit `nart=5`,
ζ_6 : Fehler der Routine NUDOCCCS mit `nart=6`,
ζ_7 : Fehler der Routine KOLLOKAT,
ζ_8 : Fehler der Routine DIROPTIM,
ζ_9 : Fehler der Routine DIRMESCH,

wobei

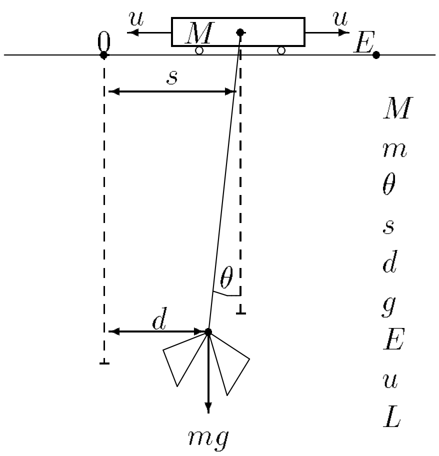
$$\zeta_i := \frac{|G_N|}{||g(x(t_0), \ldots, g(t_N))||_\infty}, \quad i = 0, \ldots, 9. \tag{5.1}$$

Alle direkten Verfahren wurden so eingestellt, daß die höchste zu erreichende Genauigkeit bei $1 \cdot 10^{-8}$ liegt. Es ist also nicht zu erwarten, daß die ζ_i, $i = 0, \ldots, 9$ diesen Wert unterschreiten.
Auf eine grafische Darstellung der adjungierten Variablen wurde verzichtet, da die hier untersuchten direkten Verfahren im Vordergrund dieser Arbeit stehen und die adjungierten Variablen zur Lösungsfindung nicht benötigen.

5.2 Anwendungsbeispiel 1: Ein zeitoptimales Problem mit Steuer- und Zustandsbeschränkungen und linear eingehender Steuerung

5.2.1 Zeitoptimale Steuerung einer Verladebrücke

Als eine erste Anwendung soll ein optimaler Steuerprozeß mit linear eingehender Steuerung und Beschränkungen im Kontroll- und Zustandsbereich behandelt werden. Probleme können hier durch die eventuelle Unstetigkeit der Steuerung erwartet werden, wenn bei den direkten Verfahren die Steuerung durch eine stetige Funktion approximiert wird.
Betrachtet wird das Problem der zeitoptimalen Überführung einer Verladebrücke punktueller Masse von einer ruhenden[1] Anfangsposition in eine ruhende Endposition.



M	Masse der Laufkatze
m	Masse des Greifers
θ	Greiferwinkel gegen die Senkrechte
s	Horizontaler Abstand der Laufkatze
d	Horizontaler Abstand des Greifers
g	Erdanziehungskraft
E	Endpunkt
u	Steuerung (äußere Kraft)
L	Länge des Pendels

Abbildung 5.2: Verladebrücke (Erzentlader)

Die Bewegungsgleichungen der Verladebrücke lassen sich mit den Bezeichnungen aus Abbildung 5.2 zu

$$\begin{aligned} \ddot{s} &= \frac{u}{M} \\ \ddot{d} &= g\sin(\theta) \end{aligned} \tag{5.2}$$

formulieren.
Da nur kleine Winkel θ zu erwarten sind, gilt

$$\theta \approx \sin(\theta) = \frac{s-d}{L}. \tag{5.3}$$

[1] d.h. die Momentangeschwindigkeit der Laufkatze und des Greifers bezüglich der Vertikalachse verschwindet

Nach einer Substitution $z = s - d$, einer Normierung der Ursprungsgrößen $\frac{g}{L} = 1$, $E = 1$ und $M = 1$, sowie einer Rückführung auf ein Differentialgleichungssystem vom Grade 1, führt dies für $0 \le t \le t_f$ auf den zeitoptimalen Steuerprozeß

$$\begin{array}{lrcl} \min & F(x,u) & = & \int_0^{t_f} 1\, dt \\ \text{unter} & \dot{x}_1(t) & = & x_2(t) \\ & \dot{x}_2(t) & = & u(t) \\ & \dot{x}_3(t) & = & x_4(t) \\ & \dot{x}_4(t) & = & u(t) - x_3(t) \\ & & & -1 \le u(t) \le 1 \end{array} \qquad x(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad x(t_f) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{5.4}$$

In der Praxis ist die Laufkatze einer Maximalgeschwindigkeit unterworfen, daher wird eine Zustandsbeschränkung

$$S(x) = x_2(t) - \alpha \le 0 \tag{5.5}$$

der Ordnung $p = 1$ (da $S^{(1)}(x,u) = \dot{x}_2 = u$) eingeführt, aus der für eine Randsteuerung $u = 0$ folgt.
Nach der Betrachtung der Ergebnisse der direkten Verfahren, liegt die Vermutung nahe, daß im unbeschränkten Fall die Steuerung bang-bang ist und folgende Struktur besitzt:

$$u(t) = \begin{cases} 1, & 0 \le t \le t_1 \\ -1, & t_1 \le t \le t_2 \\ 1, & t_2 \le t \le t_3 \\ -1, & t_3 \le t \le t_f \end{cases} \tag{5.6}$$

Für die 4 Differentialgleichungen und die 4 unbekannten Zeitpunkte t_1, t_2, t_3 und t_f stehen dann die 8 Randbedingungen zur Verfügung, so daß diese Informationen (ohne adjungierte Differentialgleichungen) ausreichen, eine Lösung mit der Routine BDSCO zu finden.
In genaueren Untersuchungen ergibt sich

$$\alpha_0 := \max_{0 \le t \le t_f} x_2(t) = t_1 = 0.802357, \tag{5.7}$$

so daß die Zustandsbeschränkung nur für $\alpha \le \alpha_0$ aktiv wird.
In analoger Weise erhält man auch hier aus den direkten Verfahren die Struktur der Steuerung

$$u(t) = \begin{cases} 1, & 0 \le t \le t_1 \\ 0, & t_1 \le t \le t_2 \\ -1, & t_2 \le t \le t_3 \\ 1, & t_3 \le t \le t_4 \\ 0, & t_4 \le t \le t_5 \\ -1, & t_5 \le t \le t_f, \end{cases} \tag{5.8}$$

jedoch reichen aufgrund der zwei neuen Unbekannten die vorhandenen Informationen nicht aus, so daß zusätzliche Bedingungen gefunden werden müssen.
Über die erweiterte Hamilton-Funktion

$$\tilde{H} = 1 + \lambda_1 x_2 + \lambda_2 u + \lambda_3 x_4 + \lambda_4 (u - x_3) + \mu(x_2 - \alpha) \tag{5.9}$$

ergeben sich die adjungierten Differentialgleichungen zu

$$\begin{aligned} \dot{\lambda}_1(t) &= 0 \\ \dot{\lambda}_2(t) &= -\lambda_1(t) - \mu(t) \\ \dot{\lambda}_3(t) &= \lambda_4(t) \\ \dot{\lambda}_4(t) &= -\lambda_3(t). \end{aligned} \tag{5.10}$$

Die Schaltfunktion

$$\Phi(t) = \tilde{H}_u = \lambda_2 + \lambda_4 \tag{5.11}$$

führt auf die entsprechenden Schaltbedingungen, die zum Beispiel in t_2, t_3 und t_5 ausgenutzt werden können. Weiterhin ergibt sich aus $\dot{\Phi}(t) = 0$ auf einem Randstück $[t_1, t_2]$

$$\mu = -(\lambda_1 + \lambda_3). \tag{5.12}$$

Nach Satz 2.43 sind die adjungierten Variablen stetig in den Ein- bzw. Austrittspunkten, es werden also keine Sprungbedingungen benötigt.
Zusätzliche Bedingungen erhält man aus der freien Endzeit:

$$\tilde{H}(t_f) = 1 + \lambda_2(t_f)u(t_f) + \lambda_4(t_f)u(t_f) + \alpha(\lambda_1(t_f) + \lambda_3(t_f)) = 0 \tag{5.13}$$

und den Eintrittsbedingungen bei t_1 und t_4

$$\begin{aligned} x_2(t_1) &= \alpha \\ x_2(t_4) &= \alpha. \end{aligned} \tag{5.14}$$

5.2.2 Numerische Auswertungen

Durch die Vorüberlegungen im vorangegangenen Abschnitt ist man nun in der Lage mit der Routine BDSCO eine hinreichend genaue Lösung der optimalen Endzeit für die unbeschränkte Verladebrücke zu erhalten. Diese kann dann mit der Lösung der verschiedenen direkten Verfahren auf unterschiedlichen Gitterstufen verglichen werden.
BDSCO liefert als optimale Zielfunktion den Wert

$$t_f = 4.2819120957$$

mit den drei Schaltpunkten

$$\begin{aligned} t_1 &= 0.802358 \\ t_2 &= 2.140956 \\ t_3 &= 3.479555 \end{aligned}$$

Tabelle 5.1 zeigt eine Zusammenfassung dieser Ergebnisse, mit dem normierten globalen Diskretisationsfehler. Die erforderliche exakte Lösung wurde dazu über die Routine BDSCO ermittelt.

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
2	4	$4.18 \cdot 10^{-1}$	$6.63 \cdot 10^{-2}$	$5.10 \cdot 10^{-2}$	$6.63 \cdot 10^{-2}$	$1.70 \cdot 10^{-1}$
3	8	$6.67 \cdot 10^{-2}$	$3.74 \cdot 10^{-2}$	$2.59 \cdot 10^{-3}$	$3.74 \cdot 10^{-2}$	$2.11 \cdot 10^{-2}$
4	16	$1.61 \cdot 10^{-2}$	$5.78 \cdot 10^{-3}$	$5.82 \cdot 10^{-3}$	$5.79 \cdot 10^{-3}$	$1.02 \cdot 10^{-2}$
5	32	$3.99 \cdot 10^{-3}$	$1.47 \cdot 10^{-3}$	$1.54 \cdot 10^{-3}$	$1.54 \cdot 10^{-3}$	$2.49 \cdot 10^{-3}$
6	64	$4.26 \cdot 10^{-3}$	$3.91 \cdot 10^{-4}$	$3.83 \cdot 10^{-4}$	$3.90 \cdot 10^{-4}$	$6.18 \cdot 10^{-4}$
7	128	$6.48 \cdot 10^{-4}$	$3.26 \cdot 10^{-4}$	$8.68 \cdot 10^{-4}$	$1.50 \cdot 10^{-4}$	$2.80 \cdot 10^{-4}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
2	4	$3.64 \cdot 10^{-2}$	$1.94 \cdot 10^{-3}$	$1.61 \cdot 10^{-1}$	$5.71 \cdot 10^{-1}$	$1.57 \cdot 10^{-1}$
3	8	$1.02 \cdot 10^{-1}$	$5.00 \cdot 10^{-3}$	$2.05 \cdot 10^{-2}$	$5.90 \cdot 10^{-2}$	$2.04 \cdot 10^{-2}$
4	16	$1.46 \cdot 10^{-1}$	$1.50 \cdot 10^{-1}$	$1.01 \cdot 10^{-2}$	$1.87 \cdot 10^{-2}$	$1.10 \cdot 10^{-2}$
5	32	$1.70 \cdot 10^{-1}$	$1.72 \cdot 10^{-1}$	$3.84 \cdot 10^{-2}$	$4.53 \cdot 10^{-3}$	$5.87 \cdot 10^{-3}$
6	64	$1.14 \cdot 10^{-1}$	$5.18 \cdot 10^{-2}$	—	—	—
7	128	$4.12 \cdot 10^{-4}$	$7.57 \cdot 10^{-4}$	—	—	—

Tabelle 5.1: Lösungsdaten der unbeschränkten Verladebrücke

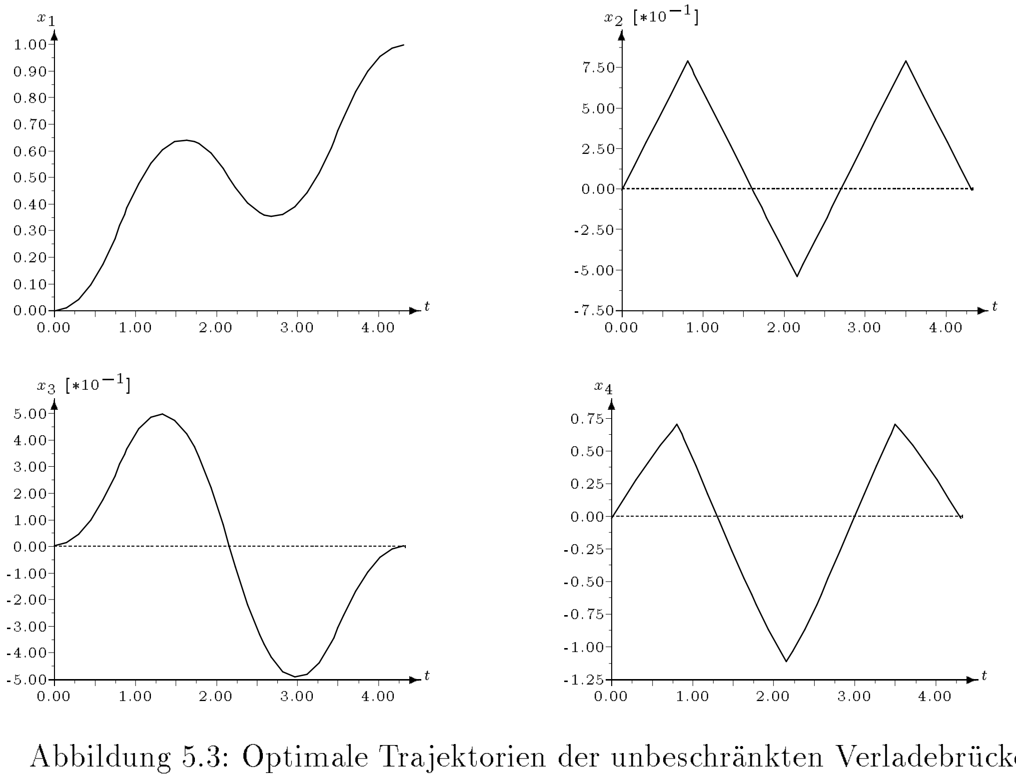


Abbildung 5.3: Optimale Trajektorien der unbeschränkten Verladebrücke

Als Startschätzung wurden, wie auch bei der beschränkten Verladebrücke, alle Optimierungsvariablen auf 0 gesetzt. Einzige Ausnahme war die mit dem Wert 5 versehene Optimierungsvariable, die sich auf die unbekannte Endzeit bezieht. Die Fehler ζ_7, ζ_8 und ζ_9 konnten aus rechentechnischen Gründen (zu großer Rechen- und Speicheraufwand) teilweise nicht ermittelt werden. Abbildung 5.3 zeigt den Verlauf der optimalen Trajektorien. Auf eine Darstellung der einfachen optimalen Steuerung wurde verzichtet.
Sämtliche Fehler der Routine NUDOCCCS bewegen sich im Bereich der Fehlergrößenordnung $O(10^{-4})$, die durch eine Erhöhung der Gitterstufen weiter verkleinert werden könnte. Es erscheint allerdings aufgrund der einfachen Struktur der Steuerung sinnvoller, die Lösung durch die in Abschnitt 4.4.2 und 4.4.3 vorgestellten Verfahren zu verbessern, da hierdurch die Anzahl der Optimierungsvariablen geringer gehalten werden kann. Erfreulich ist weiterhin, daß obwohl die Steuerung Sprünge besitzt, die Verfahren mit linearer oder kubischer Interpolation der Steuerung eine sehr gute Lösung liefern, obwohl dies für ζ_5 und ζ_6 erst für Gitter höherer Stufe gelingt.
Vergleicht man die Größenordnung der auftretenden Fehler mit den Fehlern der anderen Beispiele, so liegt die Vermutung nahe, daß die spezielle Struktur der Steuerung bessere Ergebnisse verhindert, wenn auf äquidistantem Gitter gelöst wird.
Gleiches gilt für die in Tabelle 5.2 zusammengefaßten Ergebnisse der durch $x_2 \leq 0.4$ beschränkten Verladebrücke, deren optimale Trajektorien in Abbildung 5.4 dargestellt werden. Auf eine Darstellung der Steuerung wurde ebenfalls verzichtet.

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
2	4	$1.41 \cdot 10^{+1}$	$1.45 \cdot 10^{-1}$	$3.49 \cdot 10^{-1}$	$1.45 \cdot 10^{-1}$	$7.08 \cdot 10^{-2}$
3	8	$1.44 \cdot 10^{-1}$	$6.45 \cdot 10^{-2}$	$1.99 \cdot 10^{-2}$	$6.45 \cdot 10^{-2}$	$2.12 \cdot 10^{-2}$
4	16	$2.87 \cdot 10^{-2}$	$1.65 \cdot 10^{-2}$	$2.29 \cdot 10^{-3}$	$1.66 \cdot 10^{-2}$	$7.22 \cdot 10^{-3}$
5	32	$7.08 \cdot 10^{-3}$	$5.56 \cdot 10^{-3}$	$2.97 \cdot 10^{-4}$	$5.56 \cdot 10^{-3}$	$7.06 \cdot 10^{-4}$
6	64	$1.90 \cdot 10^{-3}$	$1.04 \cdot 10^{-3}$	$2.29 \cdot 10^{-4}$	$1.23 \cdot 10^{-3}$	$5.68 \cdot 10^{-4}$
7	128	$4.86 \cdot 10^{-4}$	$4.99 \cdot 10^{-5}$	$2.14 \cdot 10^{-4}$	$6.53 \cdot 10^{-5}$	$3.15 \cdot 10^{-4}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
2	4	$8.15 \cdot 10^{-2}$	$1.10 \cdot 10^{-2}$	$6.26 \cdot 10^{-2}$	$4.38 \cdot 10^{-1}$	$5.91 \cdot 10^{-2}$
3	8	$1.58 \cdot 10^{-2}$	$8.81 \cdot 10^{-3}$	$2.03 \cdot 10^{-2}$	$8.58 \cdot 10^{-2}$	$2.01 \cdot 10^{-2}$
4	16	$3.13 \cdot 10^{-2}$	$3.66 \cdot 10^{-2}$	$1.34 \cdot 10^{-2}$	$2.04 \cdot 10^{-2}$	$1.28 \cdot 10^{-2}$
5	32	$5.58 \cdot 10^{-2}$	$5.73 \cdot 10^{-2}$	$1.38 \cdot 10^{-2}$	$6.23 \cdot 10^{-3}$	$6.33 \cdot 10^{-3}$
6	64	$2.54 \cdot 10^{-2}$	$8.26 \cdot 10^{-5}$	—	—	—
7	128	$2.18 \cdot 10^{-4}$	$2.92 \cdot 10^{-4}$	—	—	—

Tabelle 5.2: Lösungsdaten der durch $x_2 \leq 0.4$ beschränkten Verladebrücke

BDSCO liefert als optimale Endzeit den Wert

$$t_f = 4.6775996981$$

mit den Schaltpunkten

t_1	=	0.400000	Aufsprungpunkt
t_2	=	1.495568	Absprungpunkt
t_3	=	2.338800	Schaltpunkt
t_4	=	3.182032	Aufsprungpunkt
t_5	=	4.277600	Absprungpunkt

Aufgrund der veränderten Struktur der Steuerung und der Trajektorien konnte die Güte der Lösung teilweise verbessert werden.

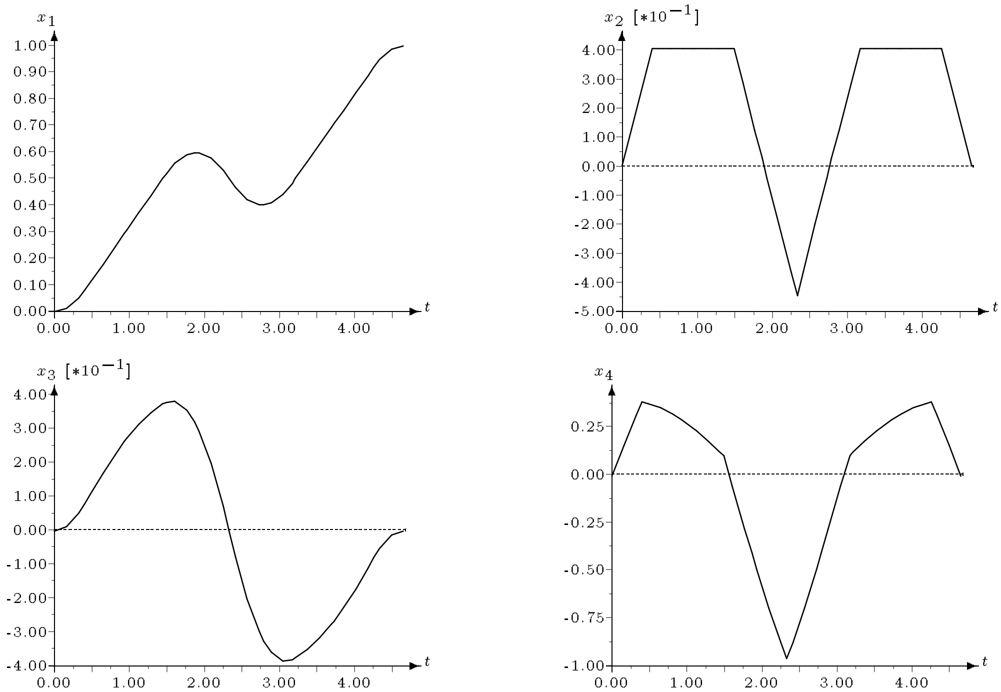


Abbildung 5.4: Optimale Trajektorien der durch $x_2 \leq 0.4$ beschränkten Verladebrücke

Abbildung 5.5 zeigt eine übliche Situation ('chattering-control') einer linear interpolierten Steuerung bei einem singulären Teilstück oder wie hier bei einem bang-bang Problem (Verladebrücke) mit aktiver Zustandsbeschränkung auf einem Gitter der Stufe 5.

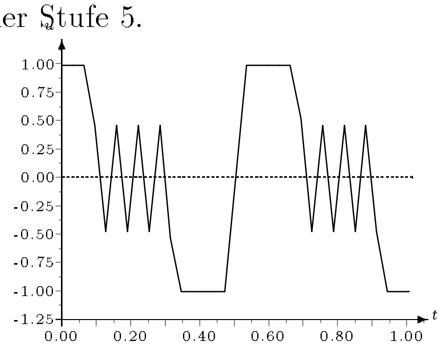


Abbildung 5.5: 'Chattering-control'

Trotz der von der tatsächlichen Steuerung stark abweichenden Struktur der so gefundenen Steuerung sind die Ergebnisse (z.B. ζ_4) genauer als bei konstanter Interpolation (ζ_2), die der Steuerstruktur eigentlich näher kommen sollte. Durch die Aufnahme der Bedingung $u = 0$ in den Aufsprungpunkten kann hier eine solche Situation aber weitestgehend vermieden werden.

5.3 Anwendungsbeispiel 2: Ein Problem mit regulärer Hamilton-Funktion und kompaktem Steuerbereich

5.3.1 Optimale Temperaturverteilung in einem chemischen Reaktor

Im folgenden sollen als weitere Anwendungsbeispiele Probleme mit regulärer Hamilton-Funktion behandelt werden, wobei zu Beginn ein steuerbeschränkter Steuerprozeß untersucht wird.
Nach der Reaktionsgleichung $A_1 \longrightarrow A_2 \longrightarrow A_3$ produziere eine chemische Fabrik in einem heizbaren Reaktor die Substanz A_2 aus A_1 mit dem Abfallprodukt A_3. Ziel ist es, die Konzentration der Substanz A_2 im Endpunkt zu maximieren.

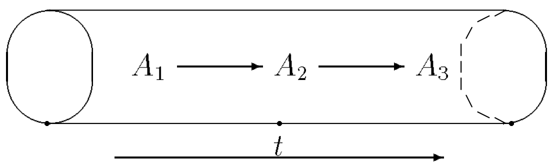


Abbildung 5.6: Chemischer Reaktor

Diese Anwendung kann beschrieben werden durch den Steuerprozeß

$$
\begin{array}{ll}
\min & -x_2(1) \\
\text{unter} & \begin{array}{rcll} \dot{x}_1(t) & = & -u(t)x_1(t) + u(t)^2 x_2(t) & x_1(0) = 1 \\ \dot{x}_2(t) & = & u(t)x_1(t) - 3u(t)^2 x_2(t) & x_2(0) = 0 \end{array} \\
 & \quad u_{min} \le u(t) \le u_{max}, \quad 0 \le t \le 1.
\end{array}
\tag{5.15}
$$

Aus der Hamilton-Funktion

$$H = \lambda_1(u^2 x_2 - u x_1) + \lambda_2(u x_1 - 3u^2 x_2) \tag{5.16}$$

und der Auswertung der Transversalitätsbedingungen ergeben sich die adjungierten Differentialgleichungen zu

$$
\begin{array}{rcll}
\dot{\lambda}_1(t) & = & (\lambda_1(t) - \lambda_2(t))u(t) & \lambda_1(1) = 0 \\
\dot{\lambda}_2(t) & = & (3\lambda_2(t) - \lambda_1(t))u(t)^2 & \lambda_2(1) = -1.
\end{array}
\tag{5.17}
$$

Mit der Hamilton-Funktion ergibt sich weiterhin aus

$$H_u = (\lambda_2 - \lambda_1)x_1 + 2u(\lambda_1 - 3\lambda_2)x_2 = 0 \tag{5.18}$$

die optimale Steuerung zu

$$u^*(x, \lambda) = -\frac{(\lambda_2 - \lambda_1)x_1}{2x_2(\lambda_1 - 3\lambda_2)}. \tag{5.19}$$

Nach Auswertung der Lösung durch die direkten Verfahren, sowie aus der Tatsache, daß $u^*(x, \lambda)$ zum Zeitpunkt $t = 0$ den Wert $+\infty$ annimmt, schließt man leicht auf die Beschränkungsverletzung $u(t) \leq u_{max}$ und damit auf ein Randstück

$$u(t) = u_{max}, \qquad 0 \leq t \leq t_1. \tag{5.20}$$

Nach Lemma 2.27 gilt die Stetigkeit der optimalen Steuerung in t_1:

$$u^*(x(t_1), \lambda(t_1)) = u_{max} \tag{5.21}$$

Mit der Routine BDSCO kann das Randwertproblem (5.15), (5.17), mit der Steuerung (5.19) bzw. (5.20) und der Bedingung (5.21) gelöst werden.

5.3.2 Numerische Auswertungen

Die Auswertung der Ergebnisse im vorangegangenen Abschnitt führt die Routine BDSCO auf den Wert

$$x_2(t_f) = -0.28810478668$$

des optimalen Zielfunktionals und den Wert

$$t_1 = 0.10585067$$

für einen Absprungpunkt.

Tabelle 5.3 zeigt die Zusammenfassung der Ergebnisse des normierten globalen Diskretisationsfehlers bei einer Rechenzeitbeschränkung von 10 Minuten und einer Belegung der Startschätzung aller Optimierunsvariablen mit dem Wert 0. Die Abbildungen 5.7 und 5.8 zeigen grafisch den optimalen Verlauf der Trajektorien und der Steuerung.

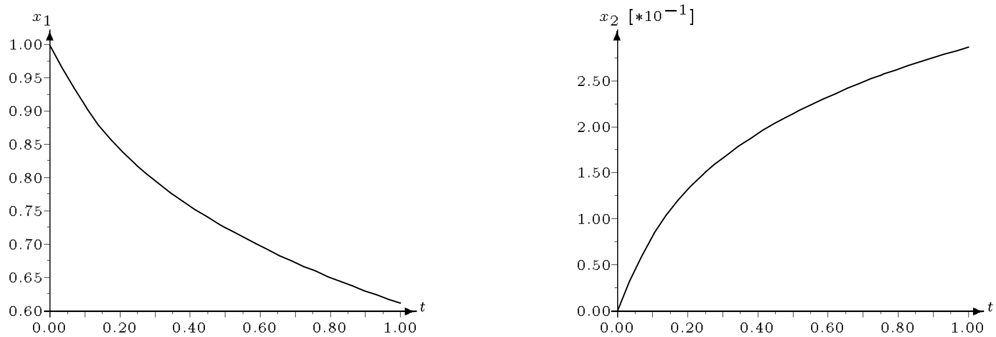


Abbildung 5.7: Optimale Trajektorien des chemischen Reaktors für $u_{max} = 1$

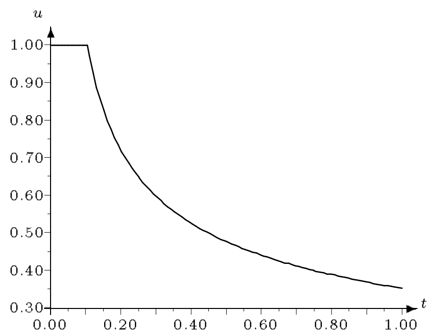


Abbildung 5.8: Optimale Steuerung des chemischen Reaktors für $u_{max} = 1$

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
1	2	$2.81 \cdot 10^{+1}$	$1.68 \cdot 10^{-2}$	$9.04 \cdot 10^{-2}$	$1.05 \cdot 10^{-1}$	$3.77 \cdot 10^{-3}$
2	4	$2.27 \cdot 10^{-1}$	$1.12 \cdot 10^{-2}$	$3.30 \cdot 10^{-2}$	$1.67 \cdot 10^{-2}$	$1.59 \cdot 10^{-3}$
3	8	$8.14 \cdot 10^{-2}$	$4.24 \cdot 10^{-3}$	$9.57 \cdot 10^{-3}$	$3.42 \cdot 10^{-3}$	$4.28 \cdot 10^{-4}$
4	16	$3.76 \cdot 10^{-2}$	$1.22 \cdot 10^{-3}$	$2.19 \cdot 10^{-3}$	$3.20 \cdot 10^{-3}$	$7.84 \cdot 10^{-5}$
5	32	$1.80 \cdot 10^{-2}$	$3.03 \cdot 10^{-4}$	$5.26 \cdot 10^{-4}$	$7.19 \cdot 10^{-4}$	$6.76 \cdot 10^{-6}$
6	64	$8.89 \cdot 10^{-3}$	$7.78 \cdot 10^{-5}$	$1.26 \cdot 10^{-4}$	$1.84 \cdot 10^{-4}$	$2.23 \cdot 10^{-6}$
7	128	$4.40 \cdot 10^{-3}$	$1.10 \cdot 10^{-4}$	$3.20 \cdot 10^{-5}$	$8.37 \cdot 10^{-5}$	$9.57 \cdot 10^{-5}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
1	2	$3.32 \cdot 10^{-5}$	$4.01 \cdot 10^{-3}$	$4.82 \cdot 10^{-3}$	$1.88 \cdot 10^{-1}$	$1.99 \cdot 10^{-3}$
2	4	$2.35 \cdot 10^{-4}$	$2.73 \cdot 10^{-3}$	$1.86 \cdot 10^{-3}$	$3.90 \cdot 10^{-2}$	$1.61 \cdot 10^{-3}$
3	8	$2.19 \cdot 10^{-3}$	$3.75 \cdot 10^{-3}$	$4.16 \cdot 10^{-4}$	$8.39 \cdot 10^{-3}$	$3.90 \cdot 10^{-4}$
4	16	$3.88 \cdot 10^{-3}$	$4.83 \cdot 10^{-3}$	$1.09 \cdot 10^{-4}$	$2.13 \cdot 10^{-3}$	$1.82 \cdot 10^{-4}$
5	32	$3.73 \cdot 10^{-3}$	$4.41 \cdot 10^{-3}$	—	$1.41 \cdot 10^{-3}$	—
6	64	$3.35 \cdot 10^{-3}$	$3.80 \cdot 10^{-3}$	—	—	—
7	128	$2.65 \cdot 10^{-5}$	$7.43 \cdot 10^{-5}$	—	—	—

Tabelle 5.3: Lösungsdaten des chemischen Reaktors für $u_{max} = 1$

Die Verschlechterung der Ergebnisse für ζ_1 und ζ_4 auf einem Gitter der Stufe sieben soll anhand des nächsten Beispieles genauer untersucht werden. Tabelle 5.4 zeigt die benötigten Rechenzeiten der verschiedenen Verfahren für $u_{min} = 0$. Diese Werte sind jedoch absolut gesehen weniger aussagekräftig, da man alle Verfahren aufgrund ihrer speziellen Eigenschaften weiter beschleunigen kann. Lediglich Tendenzen können dieser Tabelle entnommen werden.

$\hat{n}$	$l_{\hat{n}}$	0	1	2	3	4	5	6	7	8	9
1	2	0.26	0.14	0.06	0.18	0.19	1.35	0.22	0.60	0.22	5.28
2	4	0.28	0.39	0.11	0.44	0.35	0.43	0.56	4.49	1.45	13.38
3	8	0.56	0.87	0.46	3.07	0.93	1.46	2.13	18.30	5.00	72.65
4	16	2.19	4.27	2.99	4.41	5.77	7.87	10.73	336.59	69.93	291.73
5	32	14.51	19.30	12.80	31.62	39.37	48.69	68.46	>600	296.98	>600
6	64	91.85	157.39	137.50	116.29	192.76	398.27	539.14	—	>600	—
7	128	155.70	272.31	272.78	317.19	378.29	444.16	539.72	—	—	—

Tabelle 5.4: Rechenzeiten der direkten Verfahren in Sekunden

Die Rechenzeiten sind stark abhängig von der Startschätzung und den angegebenen Beschränkungen. Ein Weglassen der Beschränkung für u_{min}, die für den hier angegebenen Wert nicht aktiv wird, hat eine starke Vergrößerung des Rechenaufwandes zur Folge. Die in Abschnitt 4.4.4 beschriebene Schlauchtechnik stellt also eine sinnvolle Erweiterung dar.

5.4 Anwendungsbeispiel 3: Ein Problem mit regulärer Hamilton-Funktion und einer Beschränkung im Zustandsraum

5.4.1 Ein Spline-Problem

Das aus der numerischen Mathematik wohlbekannte Problem der Interpolation einer Funktion durch Splines, soll in einem Spezialfall für kubische Splines genauer untersucht werden.
Gesucht sei eine Funktion s: $[0, t_f] \to \mathbb{R}$, $t_f > 0$ fest, mit den Eigenschaften

(i) $\dot{s}(t)$ ist stetig,
(ii) $\ddot{s}(t)$ ist stückweise stetig
(iii) s minimiert das Integral $\int_0^{t_f} \ddot{s}(t)^2 \, dt$,

mit $s(0) = a_1$, $s(t_f) = b_1$ und $\dot{s}(0) = a_2$, $\dot{s}(t_f) = b_2$, welche die bekannte Lösung eines kubischen Polynoms für $s(t)$ besitzt. Durch Definition von

$$
\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
x_1(t) & := & s(t), & x_1(0), & := & a_1, & x_1(t_f) & := & b_1, \\
x_2(t) & := & \dot{s}(t), & x_2(0), & := & a_2, & x_2(t_f) & := & b_2, \\
u(t) & := & \ddot{s}(t), & & & & & &
\end{array}
\tag{5.22}
$$

ergibt sich für $x_0 = (a_1, a_2)^T$ und $x_f = (b_1, b_2)^T$ der äquivalente Steuerprozeß

$$
\begin{array}{lrcl}
\min & \multicolumn{3}{c}{\displaystyle\int_0^{t_f} u(t)^2 \, dt} \\
\text{unter} & \dot{x}_1(t) & = & x_2(t) \\
 & \dot{x}_2(t) & = & u(t) \\
\multicolumn{4}{l}{x(0) = x_0, \; x(t_f) = x_f, \; 0 \le t \le t_f,}
\end{array}
\tag{5.23}
$$

für eine stückweise stetige Funktion u: $[0, t_f] \to \mathbb{R}$.
Konkretisierend soll nun, nach Einführung einer Zustandsbeschränkung $S(x(t)) := x_1(t) - \alpha \le 0$ der Steuerprozeß in Mayer-Form

$$
\begin{array}{llcll}
\min & & \frac{1}{2}x_3(1) & & \\
\text{unter} & \dot{x}_1(t) & = & x_2(t), \quad x_1(0) = 0, & x_1(1) = 0 \\
& \dot{x}_2(t) & = & u(t), \quad x_2(0) = 1, & x_2(1) = -1 \\
& \dot{x}_3(t) & = & u(t)^2, \quad x_3(0) = 0, & \\
& & & x_1(t) - \alpha \le 0, \quad 0 \le t \le 1, &
\end{array}
\tag{5.24}
$$

untersucht werden.
In einem allgemeinen Ansatz ergeben sich aus der erweiterten Hamilton-Funktion

$$\tilde{H}(x, \lambda, u) = \lambda_1(t)x_2(t) + \lambda_2(t)u(t) + \lambda_3(t)u(t)^2 + \mu(t)(x_1(t) - \alpha) \tag{5.25}$$

und der Transversalitätsbedingung die adjungierten Differentialgleichungen

$$
\begin{array}{rcll}
\dot{\lambda}_1(t) & = & -\mu(t) & \\
\dot{\lambda}_2(t) & = & -\lambda_1(t) & \\
\dot{\lambda}_3(t) & = & 0, & \lambda_3(t_f) = \frac{1}{2}.
\end{array}
\tag{5.26}
$$

Die Zustandsbeschränkung hat die Ordnung 2, da

$$
\begin{array}{rcl}
S(x(t)) & = & x_1(t) - \alpha \\
S^{(1)}(x(t)) & = & x_2(t) \\
S^{(2)}(x(t)) & = & u(t).
\end{array}
\tag{5.27}
$$

Auf inneren Teilstücken gilt $\mu = 0$ und aus $\tilde{H}_u = 0$ und (5.26) ergibt sich die optimale Steuerung

$$u^*(t) = -\lambda_2(t). \tag{5.28}$$

Auf einem Randstück folgt aus $S^{(i)}(x(t)) = 0$, $i = 1, 2$

$$
\begin{array}{rcl}
x_2(t) & = & 0 \\
u(t) & = & 0
\end{array}
\tag{5.29}
$$

und nach (5.26) und (5.28) ebenfalls $\mu(t) = 0$, so daß auf dem ganzen Optimierungsintervall $\mu(t) = 0$ gilt. Spezialisierend untergliedert sich das Splineproblem in die in Tabelle 5.5 beschriebenen drei Lösungsformen.

	Bereich	Beschreibung
1	$1/4 \le \alpha < \infty$	unbeschränkte Lösung
2	$1/6 \le \alpha \le 1/4$	eine Berührpunkt
3	$0 < \alpha \le 1/6$	ein Randstück

Tabelle 5.5: Lösungsstrukturen des Spline-Problems

Problem 1 läßt sich in einfacher Weise aus den hergeleiteten Bedingungen mit der Routine BDSCO lösen. Für Probleme der Art 2 und 3 gelten zusätzlich die Sprungbedingungen in einem Eintrittspunkt

$$\lambda_1(t_1^-) = \lambda_1(t_1^+) - \nu_1(t) \tag{5.30}$$

mit den Eintrittsbedingungen (5.29). Desweiteren gilt für Probleme der Art 3 nach Bemerkung 2.46 die zusätzliche Sprungbedingung

$$\lambda_2(t_1^-) = \lambda_2(t_1^+) - \nu_2(t) \tag{5.31}$$

mit den aus (5.28) und (5.29) folgenden zusätzlichen Randbedingungen

$$\begin{aligned} \lambda_2(t_1) &= 0 \\ \lambda_2(t_2) &= 0. \end{aligned} \tag{5.32}$$

Es stehen also ausreichend viele Bedingungen zur Lösung mit der Routine BDSCO zur Verfügung.

5.4.2 Numerische Auswertungen

Für die drei Anwendungsbeispiele des Spline-Problems erhält man nach den Vorüberlegungen des vorangegangenen Abschnitts die nachfolgenden Lösungen:

Im unbeschränkten Fall erhält man mit der Routine BDSCO den optimalen Zielfunktionalwert

$$\frac{1}{2}x_3(t_f) = 2$$

ohne Schalt-, Berühr-, Aufsprung- oder Absprungpunkte.
Abbildung 5.9 spiegelt den optimalen Verlauf der Trajektorie wieder, Abbildung 5.10 zeigt die optimale Steuerung.

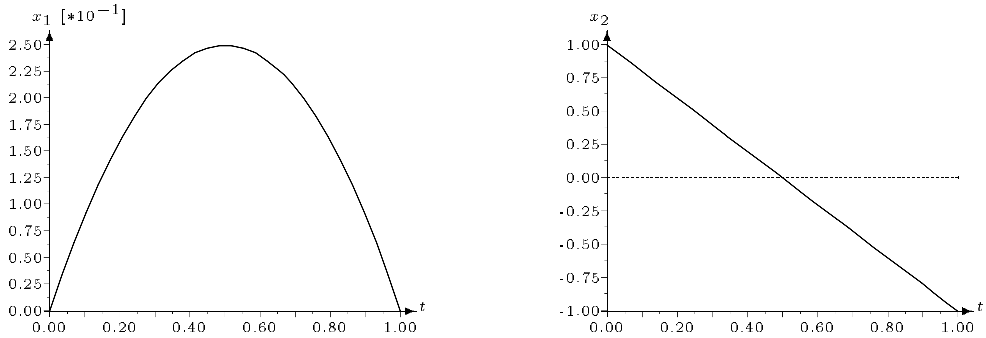


Abbildung 5.9: Optimale Trajektorien des unbeschränkten Spline-Problems

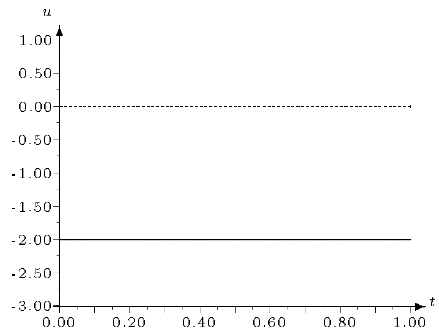


Abbildung 5.10: Optimale Steuerung des unbeschränkten Spline-Problems

Da aufgrund der einfachen Struktur der Steuerung die Lösungen der direkten Verfahren schon auf Gittern unterster Stufe im Rahmen der vorgegebenen Genauigkeitsschranken lagen, wurde auf die Darstellung der normierten globalen Diskretisationsfehler in Tabellenform verzichtet.

Für das durch $x_1 \leq 0.2$ beschränkte Spline-Problem liefert die Routine BDSCO als optimalen Wert des Zielfunktionals

$$\frac{1}{2}x_2(t_f) = 2.2399999999,$$

so daß die Vermutung nahe liegt, daß die optimale Lösung bei 2.24 liegt.
Der Berührpunkt wird bei

$$t_1 = 0.5$$

gefunden.
Ebenfalls mit einer Belegung der Startschätzung des Optimierungsvektors mit dem Wert 0, zeigt Tabelle 5.6 den normierten globalen Diskretisationsfehler der realisierten direkten Verfahren für Gitter der Stufe $1, \ldots, 9$.
Die Möglichkeit der dichten Gitter der Stufen 7, 8 und 9 (oder noch höher) verdeutlicht die Überlegenheit direkter Verfahren wie die Routine NUDOCCCS, die auf eine Diskretisierung des Zustandes verzichten. Dabei stellt die

Reduzierung des Konvergenzradiusses keine wirkliche Einschränkung dar, weil die Routine NUDOCCCS ergänzend eine Diskretisierung des Zustandes durch Mehrzielknoten (die gegebenenfalls in jeden diskreten Punkt gelegt werden) erlaubt. Es kann damit auf einem Gitter niedriger Stufe eine Startschätzung für ein Gitter höherer Stufe ermittelt werden, welches dann auf eine Aufnahme von Optimierungsvariablen, die sich auf den Zustand beziehen, verzichten kann. Dies zieht eine Reduzierung der Optimierungsvariablen, der Nebenbedingungen und deren Folgewirkungen nach sich. Abbildung 5.11 und Abbildung 5.12 zeigen den optimalen Verlauf der Trajektorien und der Steuerung.

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
1	2	$7.85 \cdot 10^{-1}$	$7.14 \cdot 10^{-2}$	$1.10 \cdot 10^{-1}$	$1.07 \cdot 10^{-1}$	$5.58 \cdot 10^{-8}$
2	4	$5.47 \cdot 10^{-1}$	$3.57 \cdot 10^{-2}$	$3.57 \cdot 10^{-2}$	$7.14 \cdot 10^{-2}$	$5.71 \cdot 10^{-8}$
3	8	$1.24 \cdot 10^{-1}$	$1.19 \cdot 10^{-2}$	$7.14 \cdot 10^{-3}$	$3.57 \cdot 10^{-2}$	$5.80 \cdot 10^{-9}$
4	16	$2.68 \cdot 10^{-2}$	$3.25 \cdot 10^{-3}$	$1.70 \cdot 10^{-3}$	$1.19 \cdot 10^{-2}$	$2.75 \cdot 10^{-14}$
5	32	$6.47 \cdot 10^{-3}$	$8.35 \cdot 10^{-4}$	$4.20 \cdot 10^{-4}$	$3.25 \cdot 10^{-3}$	$1.32 \cdot 10^{-14}$
6	64	$1.61 \cdot 10^{-3}$	$2.09 \cdot 10^{-4}$	$1.05 \cdot 10^{-4}$	$8.30 \cdot 10^{-4}$	$5.25 \cdot 10^{-15}$
7	128	$4.01 \cdot 10^{-4}$	$5.23 \cdot 10^{-5}$	$2.62 \cdot 10^{-5}$	$1.89 \cdot 10^{-4}$	$5.01 \cdot 10^{-15}$
8	256	$1.01 \cdot 10^{-4}$	$1.31 \cdot 10^{-5}$	$6.52 \cdot 10^{-6}$	$4.65 \cdot 10^{-5}$	$4.75 \cdot 10^{-15}$
9	512	$2.51 \cdot 10^{-5}$	$3.27 \cdot 10^{-6}$	$1.63 \cdot 10^{-6}$	$1.15 \cdot 10^{-5}$	$8.71 \cdot 10^{-15}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
1	2	$3.48 \cdot 10^{-8}$	$1.81 \cdot 10^{-3}$	$8.93 \cdot 10^{-10}$	$1.13 \cdot 10^{-1}$	$4.46 \cdot 10^{-10}$
2	4	$7.01 \cdot 10^{-4}$	$3.79 \cdot 10^{-4}$	$9.79 \cdot 10^{-5}$	$1.79 \cdot 10^{-1}$	$7.08 \cdot 10^{-6}$
3	8	$1.31 \cdot 10^{-4}$	$1.45 \cdot 10^{-4}$	$9.54 \cdot 10^{-4}$	$2.73 \cdot 10^{-2}$	$4.19 \cdot 10^{-6}$
4	16	$1.69 \cdot 10^{-5}$	$8.28 \cdot 10^{-6}$	$8.80 \cdot 10^{-3}$	$6.47 \cdot 10^{-3}$	$3.69 \cdot 10^{-6}$
5	32	$2.13 \cdot 10^{-6}$	$1.08 \cdot 10^{-6}$	$7.94 \cdot 10^{-3}$	$1.66 \cdot 10^{-3}$	$2.78 \cdot 10^{-6}$
6	64	$2.66 \cdot 10^{-7}$	$1.35 \cdot 10^{-7}$	$1.00 \cdot 10^{\pm 0}$	$6.37 \cdot 10^{-3}$	$5.89 \cdot 10^{-6}$
7	128	$6.88 \cdot 10^{-8}$	$3.26 \cdot 10^{-8}$	—	—	—
8	256	$8.48 \cdot 10^{-9}$	$4.02 \cdot 10^{-9}$	—	—	—
9	512	$5.43 \cdot 10^{-9}$	$1.94 \cdot 10^{-9}$	—	—	—

Tabelle 5.6: Lösungsdaten des durch $x_1 \leq 0.2$ beschränkten Spline-Problems

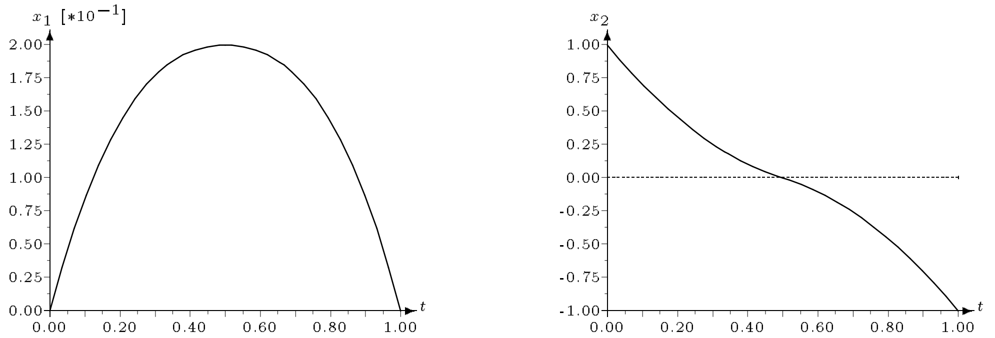


Abbildung 5.11: Optimale Trajektorien des durch $x_1 \leq 0.2$ beschränkten Spline-Problems

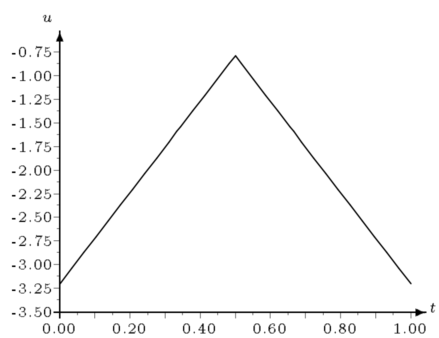


Abbildung 5.12: Optimale Steuerung des durch $x_1 \leq 0.2$ beschränkten Spline-Problems

Die sehr kleinen Fehler, bei Gittern niedriger Stufe lassen sich leicht durch die besondere Struktur der Steuerung erklären. Denn bereits durch 3 diskrete Punkte (Gitter der Stufe 1) kann die optimale Steuerung durch lineare Interpolation exakt beschrieben werden. Dies zeigen insbesondere die Ergebnisse der Routine DIRMESCH.

Im Falle des durch $x_1 \leq 0.1$ beschränkten Spline-Problems findet die Routine BDSCO im Rahmen der vorgegebenen Genauigkeitsschranken den Zielfunktionalwert

$$\frac{1}{2} x_3(t_f) = 4.4444444444,$$

so daß hier die Vermutung nahe liegt, daß die optimale Lösung bei $4.\bar{4}$ liegt. Das Randstück wird durch

$$\begin{array}{rcll} t_1 & = & 0.3 & \text{Aufsprungpunkt} \\ t_2 & = & 0.7 & \text{Absprungpunkt} \end{array}$$

eingeschränkt.
Der normierte globale Diskretisationsfehler wird auf Gittern, wie bei dem durch $x_1 \leq 0.2$ beschränkten Spline-Problem mit gleicher Startschätzung, in Tabelle 5.7 beschrieben. Die Abbildungen 5.13 und 5.14 verdeutlichen den Verlauf der optimalen Trajektorien und der optimalen Steuerung.

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
1	2	$1.01 \cdot 10^{-1}$	$3.88 \cdot 10^{-1}$	$5.49 \cdot 10^{-1}$	$5.50 \cdot 10^{-1}$	$6.40 \cdot 10^{-2}$
2	4	$5.47 \cdot 10^{-1}$	$2.08 \cdot 10^{-1}$	$3.32 \cdot 10^{-1}$	$3.88 \cdot 10^{-1}$	$1.00 \cdot 10^{-2}$
3	8	$6.43 \cdot 10^{-1}$	$7.43 \cdot 10^{-2}$	$4.16 \cdot 10^{-2}$	$2.08 \cdot 10^{-1}$	$5.83 \cdot 10^{-3}$
4	16	$1.27 \cdot 10^{-1}$	$2.08 \cdot 10^{-2}$	$9.78 \cdot 10^{-3}$	$7.43 \cdot 10^{-2}$	$8.30 \cdot 10^{-4}$
5	32	$2.82 \cdot 10^{-2}$	$5.40 \cdot 10^{-3}$	$2.51 \cdot 10^{-3}$	$2.08 \cdot 10^{-2}$	$1.57 \cdot 10^{-4}$
6	64	$6.85 \cdot 10^{-3}$	$1.35 \cdot 10^{-3}$	$6.71 \cdot 10^{-4}$	$5.40 \cdot 10^{-3}$	$4.31 \cdot 10^{-6}$
7	128	$1.70 \cdot 10^{-3}$	$3.74 \cdot 10^{-4}$	$1.66 \cdot 10^{-4}$	$1.52 \cdot 10^{-3}$	$1.63 \cdot 10^{-6}$
8	256	$4.24 \cdot 10^{-4}$	$8.49 \cdot 10^{-5}$	$4.21 \cdot 10^{-5}$	$3.38 \cdot 10^{-4}$	$2.66 \cdot 10^{-7}$
9	512	$1.06 \cdot 10^{-4}$	$2.12 \cdot 10^{-5}$	$3.16 \cdot 10^{-5}$	$8.15 \cdot 10^{-5}$	$7.43 \cdot 10^{-8}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
1	2	$6.40 \cdot 10^{-2}$	$5.58 \cdot 10^{-2}$	$6.40 \cdot 10^{-2}$	$5.58 \cdot 10^{-1}$	$6.40 \cdot 10^{-2}$
2	4	$6.81 \cdot 10^{-3}$	$8.44 \cdot 10^{-3}$	$1.00 \cdot 10^{-2}$	$3.89 \cdot 10^{-1}$	$1.00 \cdot 10^{-2}$
3	8	$5.41 \cdot 10^{-3}$	$5.65 \cdot 10^{-3}$	$1.30 \cdot 10^{-2}$	$1.93 \cdot 10^{-1}$	$5.84 \cdot 10^{-3}$
4	16	$7.55 \cdot 10^{-4}$	$7.93 \cdot 10^{-4}$	$1.86 \cdot 10^{-2}$	$3.94 \cdot 10^{-2}$	$8.31 \cdot 10^{-4}$
5	32	$1.51 \cdot 10^{-4}$	$1.54 \cdot 10^{-4}$	$2.37 \cdot 10^{-1}$	$1.02 \cdot 10^{-2}$	$5.14 \cdot 10^{-4}$
6	64	$3.17 \cdot 10^{-6}$	$2.29 \cdot 10^{-6}$	$1.00 \cdot 10^{\pm 0}$	$1.77 \cdot 10^{-2}$	$9.07 \cdot 10^{-3}$
7	128	$7.87 \cdot 10^{-7}$	$1.45 \cdot 10^{-6}$	—	—	—
8	256	$1.82 \cdot 10^{-7}$	$2.02 \cdot 10^{-7}$	—	—	—
9	512	$5.43 \cdot 10^{-8}$	$7.09 \cdot 10^{-8}$	—	—	—

Tabelle 5.7: Lösungsdaten des durch $x_1 \leq 0.1$ beschränkten Spline-Problems

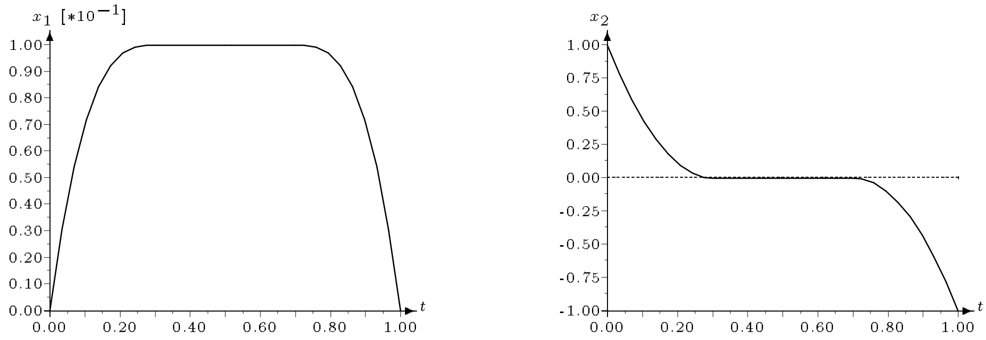


Abbildung 5.13: Optimale Trajektorien des durch $x_1 \leq 0.1$ beschränkten Spline-Problems

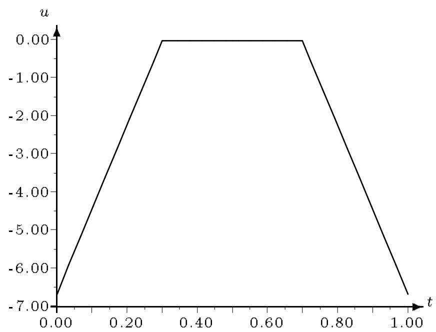


Abbildung 5.14: Optimale Steuerung des durch $x_1 \leq 0.1$ beschränkten Spline-Problems

Die Fehler sind etwas größer als die in der vorangegangenen Untersuchung, deren Ursache vermutlich in den Ein- und Austrittspunkten der Steuerung zu finden ist, die durch die gewählten Gitter nicht genau getroffen werden.
Auffällig ist, wie schon beim chemischen Reaktor für ζ_1 und ζ_4, das Verhalten des normierten globalen Diskretisationsfehlers für ζ_7, ζ_8 und ζ_9, der mit zunehmender Gitterstufe teilweise immer größer wird. Die Lösungen lassen sich anscheinend nicht weiter verbessern.
Die Ergebnisse aller hier durchgeführten Untersuchungen zeigen deutlich, daß diese Situation speziell bei den Routinen KOLLOKAT, DIROPTIM und DIRMESCH schon sehr frühzeitig (niedrige Gitterstufen) auftreten können. Eine Erklärung läßt sich in den speziellen Strukturen der Gradienten des Zielfunktionals finden. Da der jeweilige Gradient nur abhängig von wenigen Optimierungsvariablen[2] ist, kann es leicht bei der Berechnung des Differen-

[2] nämlich die Optimierungsvariablen, die sich unmittelbar auf die Trajektorien und Steuerungen am Intervallende beziehen

tenquotienten zu Rundungsfehlern durch Auslöschung kommen. Diese wiederum führen zu einem vorzeitigen Abbruch der Verfahren.
Die Routine NUDOCCCS reagiert viel unempfindlicher auf diese Probleme, da der Gradient eine andere Struktur besitzt, als die anderen Verfahren. Der Gradient ist abhängig von allen Optimierungsvariablen, so daß es nicht so leicht zu Rundungsfehlern durch Auslöschung kommen kann.

5.5 Anwendungsbeispiel 4: Ein Problem mit regulärer Hamilton-Funktion, Zustandsbeschränkungen und 'Pseudominima'

5.5.1 Knickstab

Als ein zustandsbeschränktes Problem soll nun der in [Maurer 1991] beschriebene Steuerprozeß eines Knickstabes untersucht werden.

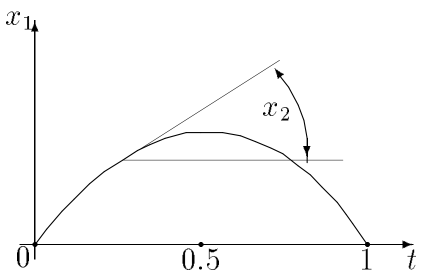


Abbildung 5.15: Knickstab

Im einzelnen ergibt sich der Steuerprozeß in Mayer-Form zu

$$\begin{array}{lrcll} \min & & & \frac{1}{2}x_3(t_f) + \alpha x_4(t_f) & \\ \text{unter} & \dot{x}_1(t) & = & \sin(x_2(t)), & x_1(0) = 0,\ x_1(1) = 0 \\ & \dot{x}_2(t) & = & u(t) & \\ & \dot{x}_3(t) & = & u(t)^2 & x_3(0) = 0 \\ & \dot{x}_4(t) & = & \cos(x_2(t)) & x_4(0) = 0 \\ & & & 0 \le t \le 1 & \end{array} \tag{5.33}$$

mit den Zustandsbeschränkungen

$$\begin{array}{rcrcl} S_1(x_1(t)) & = & x_1(t) - 0.05 & \le & 0, \\ S_2(x_1(t)) & = & -x_1(t) - 0.05 & \le & 0 \end{array} \tag{5.34}$$

der Ordnung $p = 2$, da

$$\left.\begin{array}{rcl} S_i^{(1)}(x_1(t)) & = & \pm \sin x_2(t) \\ S_i^{(2)}(x_1(t)) & = & \pm u \cos x_2(t) \end{array}\right\} \quad i = 1, 2. \tag{5.35}$$

Aus der erweiterten Hamilton-Funktion

$$\tilde{H} = \lambda_1 \sin(x_2) + \lambda_2 u + \lambda_3 u^2 + \lambda_4 \cos(x_2) + \mu_1(x_1 - 0.05) + \mu_2(-x_1 - 0.05) \quad (5.36)$$

ermittelt man zusammen mit der Transversalitätsbedingung die adjungierten Differentialgleichungen zu

$$\begin{array}{rclrcl} \dot{\lambda}_1(t) & = & -(\mu_1(t) - \mu_2(t)) & & & \\ \dot{\lambda}_2(t) & = & -\lambda_1(t)\cos(x_2(t)) + \lambda_4(t)\sin(x_2(t)) & \lambda_2(0) & = & 0,\ \lambda_2(1) = 0 \\ \dot{\lambda}_3(t) & = & 0 & \lambda_3(1) & = & \frac{1}{2} \\ \dot{\lambda}_4(t) & = & 0 & \lambda_4(1) & = & \alpha. \end{array} \quad (5.37)$$

Auf inneren Teilstücken gilt $\mu_1 = \mu_2 = 0$ und die optimale Steuerung ergibt sich aus $\tilde{H}_u = 0$ zu

$$u^*(t) = -\frac{\lambda_2(t)}{2\lambda_3(t)} \quad (5.38)$$

Auf einem Randstück $[t_1, t_2] \subset [0, 1]$ folgt dann wegen $S_i^{(j)}(x) = 0$, $j = 0, 1, 2$, $i = 1, 2$

$$\begin{array}{rcl} x_1(t) & = & \pm 0.05 \\ x_2(t) & = & 0 \\ u(t) & = & 0. \end{array} \quad (5.39)$$

Aus (5.38) und (5.39) folgt weiterhin $\lambda_2 = 0$, $\dot{\lambda}_2 = 0$, mit (5.37) führt dies schließlich auf $\lambda_1 = 0$, $\dot{\lambda}_1 = 0$. Abschließende Konsequenz ist damit ebenfalls $\mu_1 = \mu_2 = 0$ auf einem Randstück, da die Zustandsbeschränkungen nicht gleichzeitig aktiv werden können.

In Abhängigkeit vom Parameter α unterscheidet man, ebenso wie beim Splin-Problem, verschiedene Lösungsformen. Jedoch können hier, aufgrund der in (5.33) und (5.34) gegebenen Struktur, für größere α mehrere Randstücke oder Berührpunkte auftreten. Ein weiteres Problem besteht im Auftreten von Nebenminima für bestimmte α, in denen die direkten Verfahren 'hängen bleiben' können.

Konkretisierend erhält man analog zum Spline-Problem für die optimale Lösung die in Tabelle 5.8 dargestellten Strukturen, wobei für $0 \leq \alpha \leq \pi^2$ eine triviale Lösung $x_1 = 0$ voliegt.

Für eine umfangreiche Untersuchung sei an dieser Stelle auf [Maurer 1991] verwiesen.

Probleme der ersten Art (unbeschränkte) lassen sich mit den verhandenen Informationen in einfacher Weise durch die Routine BDSCO lösen. Bei Problemen der zweiten und dritten Art gelten zusätzlich die Sprungbedingungen in einem Eintrittspunkt

$$\lambda_1(t_1^-) = \lambda_1(t_1^+) - \nu_1(t) \quad (5.40)$$

[3] Für größere α setzt sich das Auftreten von Berührpunkten und Randstücken fort.

	Bereich					Beschreibung
1	0	$\leq$	α	$\leq$	9.90027	unbeschränkte Lösung
2	9.90027	$\leq$	α	$\leq$	39.3291	Berührpunkt
3	39.3291	$\leq$	α	$\leq$	105.190	Randstück
4	105.190	$\leq$	α	$\leq$	342.253	Berührpunkte
5	342.253	$\leq$	α	$\leq$	$\cdots$	Randstücke
$\vdots$ 3			$\vdots$			$\vdots$

Tabelle 5.8: Lösungsstrukturen des Knickstab-Problems

und nach (5.39) die Eintrittsbedingungen

$$\begin{aligned} x_1(t) &= \pm 0.05 \\ x_2(t) &= 0. \end{aligned} \tag{5.41}$$

Desweiteren gilt für Probleme der dritten Art nach Bemerkung 2.46 die zusätzliche Sprungbedingung

$$\lambda_2(t_1^-) = \lambda_2(t_1^+) - \nu_2(t) \tag{5.42}$$

mit den aus (5.38) und (5.39) folgenden zusätzlichen Randbedingungen

$$\begin{aligned} \lambda_2(t_1) &= 0 \\ \lambda_2(t_2) &= 0, \end{aligned} \tag{5.43}$$

so daß auch hier ausreichend viele Bedingungen für eine Bearbeitung mit der Routine BDSCO zur Verfügung stehen.
Für Probleme der vierten Art treten mehrere Berührpunkte auf, so daß in diesem Fall zusätzliche Sprung- und Schaltbedingungen der Form (5.40) und (5.41) in den anderen Berührpunkten gelten.
Probleme der fünften Art und höher sollen hier nicht näher untersucht werden.

5.5.2 Numerische Auswertungen

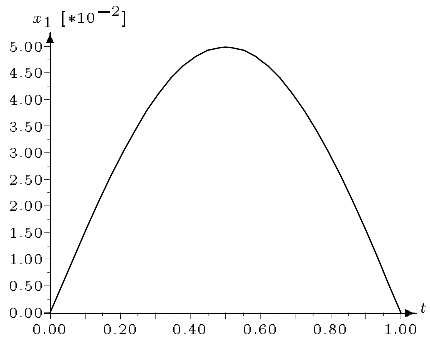
Die Auswertung der Ergebnisse im vorangegangenen Abschnitt führt mit der Routine BDSCO auf einen Wert des optimalen Zielfunktionals für $\alpha =$ 9.90027 von

$$\frac{1}{2}x_3(t_f) + \alpha x_4(t_f) = 0.9001750869$$

Tabelle 5.9 zeigt eine Zusammenfassung der berechneten Ergebnisse. Die optimalen Trajektorien x_1 und x_2, sowie die optimale Steuerung werden in den Abbildungen 5.16 und 5.17 aufgezeigt. Da die die Trajektorien für x_3 und x_4 nur zur Bestimmung des Zielfunktionals dienen, wurde auf deren Darstellung bei allen drei Anwendungen zum Knickstab verzichtet.

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
1	2	$9.50 \cdot 10^{-4}$	$3.28 \cdot 10^{-2}$	$9.59 \cdot 10^{-6}$	$5.00 \cdot 10^{-3}$	$2.95 \cdot 10^{-5}$
2	4	$2.96 \cdot 10^{-4}$	$1.17 \cdot 10^{-3}$	$9.59 \cdot 10^{-6}$	$4.74 \cdot 10^{-2}$	$1.53 \cdot 10^{-6}$
3	8	$7.81 \cdot 10^{-5}$	$3.11 \cdot 10^{-4}$	$9.59 \cdot 10^{-6}$	$2.67 \cdot 10^{-1}$	$1.15 \cdot 10^{-7}$
4	16	$1.98 \cdot 10^{-5}$	$7.90 \cdot 10^{-5}$	$9.59 \cdot 10^{-6}$	$9.59 \cdot 10^{-6}$	$2.11 \cdot 10^{-8}$
5	32	$4.98 \cdot 10^{-6}$	$1.99 \cdot 10^{-5}$	$7.35 \cdot 10^{-6}$	$9.59 \cdot 10^{-6}$	$1.52 \cdot 10^{-8}$
6	64	$1.26 \cdot 10^{-6}$	$4.98 \cdot 10^{-6}$	$2.31 \cdot 10^{-6}$	$9.59 \cdot 10^{-6}$	$1.48 \cdot 10^{-8}$
7	128	$3.25 \cdot 10^{-7}$	$1.26 \cdot 10^{-6}$	$5.96 \cdot 10^{-7}$	$9.59 \cdot 10^{-6}$	$1.47 \cdot 10^{-8}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
1	2	$1.96 \cdot 10^{-4}$	$6.17 \cdot 10^{-5}$	$8.96 \cdot 10^{-7}$	$2.28 \cdot 10^{-4}$	$8.90 \cdot 10^{-6}$
2	4	$1.01 \cdot 10^{-5}$	$1.51 \cdot 10^{-7}$	$1.05 \cdot 10^{-5}$	$1.01 \cdot 10^{-5}$	$1.74 \cdot 10^{-5}$
3	8	$6.29 \cdot 10^{-7}$	$1.68 \cdot 10^{-8}$	$4.15 \cdot 10^{-5}$	$9.70 \cdot 10^{-6}$	$1.49 \cdot 10^{-5}$
4	16	$5.07 \cdot 10^{-8}$	$5.35 \cdot 10^{-9}$	$2.17 \cdot 10^{-4}$	$7.36 \cdot 10^{-6}$	$7.97 \cdot 10^{-6}$
5	32	$1.66 \cdot 10^{-8}$	$1.35 \cdot 10^{-8}$	$9.55 \cdot 10^{-4}$	$5.01 \cdot 10^{-5}$	$4.07 \cdot 10^{-5}$
6	64	$8.18 \cdot 10^{-9}$	$1.17 \cdot 10^{-8}$	—	—	—
7	128	$1.39 \cdot 10^{-8}$	$1.43 \cdot 10^{-8}$	—	—	—

Tabelle 5.9: Lösungsdaten des Knickstab-Problems für $\alpha = 9.90027$



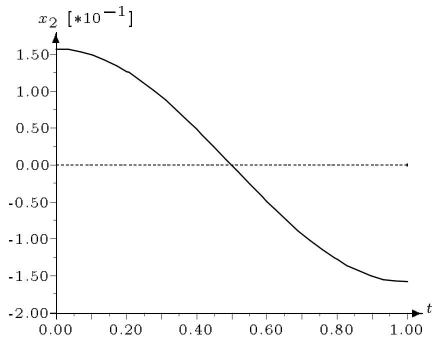


Abbildung 5.16: Optimale Trajektorien des Knickstabes für $\alpha = 9.90027$

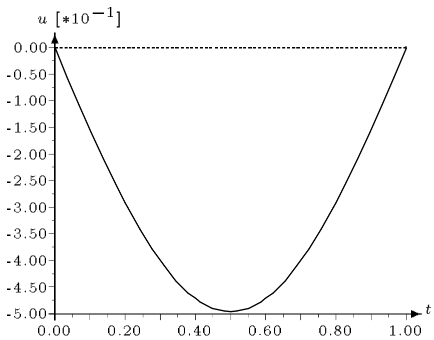


Abbildung 5.17: Optimale Steuerung des Knickstabes für $\alpha = 9.90027$

Das so beschriebene Problem nimmt, aufgrund der stetigen Differenzierbarkeit der Steuerung, eine Sonderstellung bei den bisherigen Anwendungsbeispielen ein. Da die kubische Interpolation der Steuerung bei den Fehlern ζ_5 und ζ_6 die stetige Differenzierbarkeit als Grundvoraussetzung annimmt, sind die resultierenden Fehler besonders klein. Der für ζ_2 und ζ_3 häufig auftretende Wert $9.59 \cdot 10^{-6}$ entsteht durch ein Nebenminimum, welches den Wert α besitzt, wenn die optimale Steuerung $u = 0$, also trivial ist.
Dies beschreibt auch den Grund, warum für alle Knickstabanwendungen die Startschätzung der Optimierungsvariablen zu 0.1 und nicht zu 0 gewählt werden mußte. Bei einer Startschätzung mit dem Wert 0 enden alle direkten Verfahren im 'vorgegebenen' Pseudominimum α. Die für ζ_2 und ζ_3 auftretenden Nebenminima können durch eine geeignete Veränderung der Startschätzung umgangen werden.
Da wiederum die obere Fehlerschranke der direkten Verfahren in der Größenordnung $O(10^{-8})$ angegeben wurde, wird dieser Wert für ζ_4, ζ_5 und ζ_6 nicht unterschritten. Eine Verbesserung der Lösung durch Herabsetzen der Fehlerschranke kann also durchaus erwartet werden.

Für $\alpha = 39.3291$ haben die optimalen Trajektorien und die optimale Steuerung den in den Abbildungen 5.18 und 5.19 dargestellten Verlauf. Tabelle 5.10 verdeutlicht die Ergebnisse der direkten Verfahren, wobei sich aus der Routine BDSCO der optimale Wert des Zielfunktionals zu

$$\frac{1}{2}x_3(t_f) + \alpha x_4(t_f) = 39.131899449$$

ergibt.
Ein Berührpunkt liegt bei

$$t_1 = 0.5$$

vor.

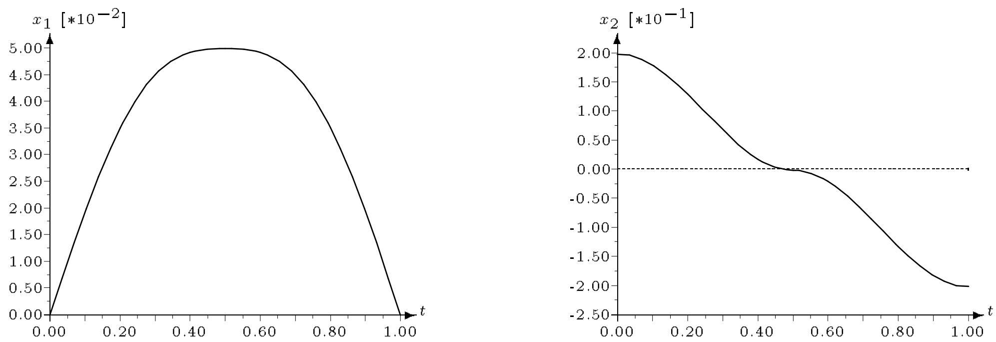


Abbildung 5.18: Optimale Trajektorien des Knickstabes für $\alpha = 39.3291$

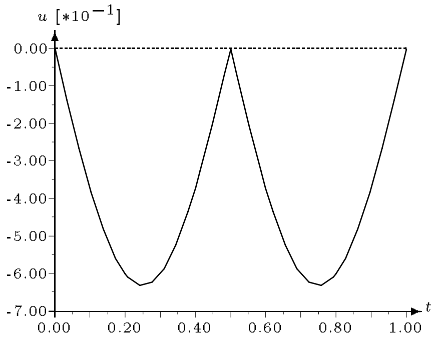


Abbildung 5.19: Optimale Steuerung des Knickstabes für $\alpha = 39.3291$

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
1	2	$1.03 \cdot 10^{-3}$	$6.60 \cdot 10^{-1}$	$2.05 \cdot 10^{-3}$	$1.56 \cdot 10^{-6}$	$3.33 \cdot 10^{-4}$
2	4	$9.69 \cdot 10^{-4}$	$2.07 \cdot 10^{-2}$	$7.19 \cdot 10^{-4}$	$4.59 \cdot 10^{-2}$	$1.32 \cdot 10^{-5}$
3	8	$1.52 \cdot 10^{-4}$	$6.36 \cdot 10^{-3}$	$2.30 \cdot 10^{-4}$	$3.04 \cdot 10^{-1}$	$6.41 \cdot 10^{-7}$
4	16	$3.40 \cdot 10^{-5}$	$1.38 \cdot 10^{-4}$	$6.31 \cdot 10^{-5}$	$5.41 \cdot 10^{-1}$	$5.60 \cdot 10^{-8}$
5	32	$8.27 \cdot 10^{-5}$	$3.32 \cdot 10^{-5}$	$1.62 \cdot 10^{-5}$	$5.04 \cdot 10^{-3}$	$1.76 \cdot 10^{-8}$
6	64	$2.06 \cdot 10^{-6}$	$8.22 \cdot 10^{-6}$	$4.06 \cdot 10^{-6}$	$5.04 \cdot 10^{-3}$	$1.53 \cdot 10^{-8}$
7	128	$6.15 \cdot 10^{-6}$	$5.60 \cdot 10^{-6}$	$1.01 \cdot 10^{-6}$	$5.04 \cdot 10^{-3}$	$1.48 \cdot 10^{-8}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
1	2	$3.88 \cdot 10^{-2}$	$2.77 \cdot 10^{-1}$	$3.33 \cdot 10^{-4}$	$3.05 \cdot 10^{-3}$	$3.41 \cdot 10^{-4}$
2	4	$4.44 \cdot 10^{-5}$	$1.80 \cdot 10^{-5}$	$1.27 \cdot 10^{-4}$	$5.18 \cdot 10^{-4}$	$6.27 \cdot 10^{-5}$
3	8	$1.20 \cdot 10^{-6}$	$2.46 \cdot 10^{-6}$	$3.12 \cdot 10^{-4}$	$1.89 \cdot 10^{-5}$	$1.45 \cdot 10^{-4}$
4	16	$1.00 \cdot 10^{-6}$	$3.03 \cdot 10^{-7}$	$5.79 \cdot 10^{-4}$	$2.39 \cdot 10^{-4}$	$2.48 \cdot 10^{-4}$
5	32	$1.33 \cdot 10^{-7}$	$6.62 \cdot 10^{-8}$	$1.47 \cdot 10^{-3}$	$2.71 \cdot 10^{-4}$	$3.09 \cdot 10^{-4}$
6	64	$8.69 \cdot 10^{-9}$	$4.86 \cdot 10^{-9}$	—	—	—
7	128	$1.01 \cdot 10^{-5}$	$1.02 \cdot 10^{-5}$	—	—	—

Tabelle 5.10: Lösungsdaten des Knickstab-Problems für $\alpha = 39.3291$

Der Wert $5.04 \cdot 10^{-3}$ für ζ_3 deutet auch hier die Existenz eines Nebenminimums, welches wiederum den Wert α besitzt, an.
Während es sich jedoch beim Knickstabbeispiel für $\alpha = 9.90027$ noch um ein relativ gutartiges Pseudominimum handelte (Fehler $9.59 \cdot 10^{-6}$), gelangt man hier bereits in Bereiche, die bei praktischen Anwendungen von Bedeutung sein können. Es ist leicht einzusehen, daß es bei anderen Anwendungen Nebenminima geben kann, die in ganz anderen Bereichen des Lösungsraums liegen.
Unterstützt werden die guten Ergebnisse aller direkten Verfahren durch die nicht stetig differenzierbare Stelle der Steuerung bei $t = 0.5$, die bereits auf Gittern unterster Stufe exakt durch die Diskretisierung erreicht wird.

Auch bei der dritten Anwendung des für $\alpha = 105.190$ in den Abbildungen

5.20 und 5.21 dargestellten Knickstab-Problems zeigt sich bei ζ_3 durch den Wert $8.34 \cdot 10^{-3}$ in Tabelle 5.11 die Existenz eines Pseudominimums.
Die Routine BDSCO liefert den Zielfunktionalwert

$$\frac{1}{2}x_3(t_f) + \alpha x_4(t_f) = 104.32039113$$

mit

$$\begin{array}{lcll} t_1 & = & 0.3047235 & \text{Aufsprungpunkt} \\ t_2 & = & 0.6952765 & \text{Absprungpunkt.} \end{array}$$

Überdeutlich zeigt sich gerade in dieser Anwendung durch die stellenweise Verschlechterung der Ergebnisse, der teilweise vorzeitige Abbruch durch Rundungsfehler.

$\hat{n}$	$l_{\hat{n}}$	ζ_0	ζ_1	ζ_2	ζ_3	ζ_4
1	2	$3.67 \cdot 10^{-3}$	$1.49 \cdot 10^{\pm 0}$	$4.05 \cdot 10^{-3}$	$3.28 \cdot 10^{-3}$	$9.01 \cdot 10^{-2}$
2	4	$1.17 \cdot 10^{-2}$	$8.89 \cdot 10^{-1}$	$2.27 \cdot 10^{-3}$	$2.09 \cdot 10^{-2}$	$3.71 \cdot 10^{-4}$
3	8	$6.08 \cdot 10^{-4}$	$2.88 \cdot 10^{-2}$	$7.46 \cdot 10^{-4}$	$3.10 \cdot 10^{-1}$	$2.13 \cdot 10^{-4}$
4	16	$1.68 \cdot 10^{-4}$	$6.71 \cdot 10^{-4}$	$2.76 \cdot 10^{-4}$	$3.78 \cdot 10^{-1}$	$2.92 \cdot 10^{-5}$
5	32	$3.88 \cdot 10^{-5}$	$1.53 \cdot 10^{-4}$	$6.79 \cdot 10^{-5}$	$8.34 \cdot 10^{-3}$	$4.51 \cdot 10^{-6}$
6	64	$9.26 \cdot 10^{-6}$	$3.72 \cdot 10^{-5}$	$1.77 \cdot 10^{-5}$	$8.34 \cdot 10^{-3}$	$4.98 \cdot 10^{-7}$
7	128	$2.01 \cdot 10^{-3}$	$2.01 \cdot 10^{-3}$	$1.49 \cdot 10^{-3}$	$2.01 \cdot 10^{-3}$	$1.44 \cdot 10^{-3}$
		ζ_5	ζ_6	ζ_7	ζ_8	ζ_9
1	2	$2.32 \cdot 10^{-1}$	$7.54 \cdot 10^{-1}$	$8.31 \cdot 10^{-4}$	$1.17 \cdot 10^{-3}$	$1.20 \cdot 10^{-3}$
2	4	$1.97 \cdot 10^{-2}$	$1.96 \cdot 10^{-2}$	$3.71 \cdot 10^{-4}$	$8.32 \cdot 10^{-5}$	$1.20 \cdot 10^{-3}$
3	8	$2.43 \cdot 10^{-4}$	$2.07 \cdot 10^{-4}$	$2.89 \cdot 10^{-4}$	$6.05 \cdot 10^{-4}$	$7.73 \cdot 10^{-4}$
4	16	$3.02 \cdot 10^{-5}$	$2.80 \cdot 10^{-5}$	$3.84 \cdot 10^{-4}$	$9.26 \cdot 10^{-4}$	$1.13 \cdot 10^{-3}$
5	32	$4.42 \cdot 10^{-5}$	$4.38 \cdot 10^{-6}$	$3.67 \cdot 10^{-3}$	$2.32 \cdot 10^{-3}$	$5.87 \cdot 10^{-3}$
6	64	$4.86 \cdot 10^{-7}$	$4.84 \cdot 10^{-7}$	—	—	—
7	128	$1.45 \cdot 10^{-3}$	$1.52 \cdot 10^{-3}$	—	—	—

Tabelle 5.11: Lösungsdaten des Knickstab-Problems für $\alpha = 105.190$

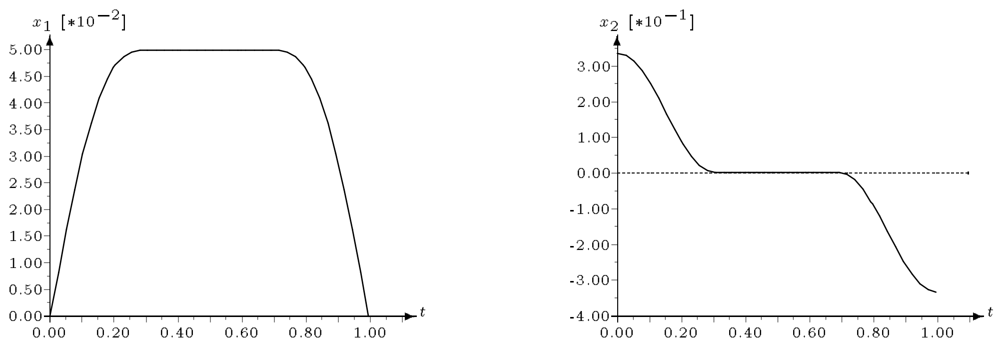


Abbildung 5.20: Optimale Trajektorien des Knickstabes für $\alpha = 105.190$

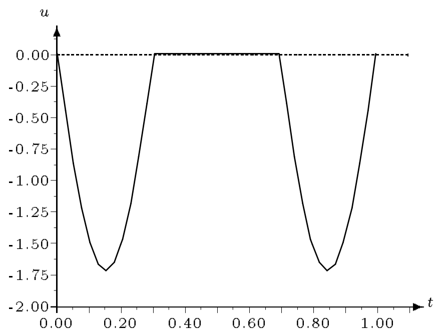


Abbildung 5.21: Optimale Steuerung des Knickstabes für $\alpha = 105.190$

5.5.3 Das Auftreten von 'Pseudominima'

Wie bereits im vorangegangenen Abschnitt angedeutet, stellen Pseudominima den einzigen wirklichen Nachteil der direkten Verfahren gegenüber den indirekten Verfahren dar. Im Gegensatz zum indirekten Ansatz, bei dem die Möglichkeit des Nachweises notwendiger und teilweise auch hinreichender Bedingungen gegeben ist, kann man bei Anwendung der direkten Verfahren ohne weitere Untersuchungen des Ausgangsproblems nicht sicher sein, die tatsächliche optimale Lösung gefunden zu haben.
Oftmals reicht aber schon eine einfache Änderung der Startschätzung aus, um Nebenminima zu umgehen. Dies wird noch einmal in Abbildung 5.23 für das Knickstab-Problem verdeutlicht. Im Sinne einer Homotopie wurde der Parameter α äquidistant variiert:

$$9.9 \leq \alpha_i \leq 149.9, \quad i = 1, \ldots, 20,$$

um zum einen für $i = 20 \longrightarrow 1$ und zum anderen für $i = 1 \longrightarrow 20$ die Lösung mit Hilfe der Routine NUDOCCCS mit 40 diskreten Punkten zu berechnen. Als Startschätzung der Steuerung wurde jeweils die Lösung des vorangegangenen Problems übernommen.
In Abbildung 5.23 sind die so gefundenen Lösungsdaten grafisch als 3-dimensionales Gitter, mit dem in Abbildung 5.22 verdeutlichten räumlichen Aufbau, aufbereitet. Trotz der Berechnung identischer Optimierungsprobleme, werden in Abhängigkeit von der Startschätzung unterschiedliche Minima gefunden.

5.6 Zusammenfassung der numerischen Ergebnisse

Die numerischen Ergebnisse dieses Kapitels belegen in deutlicher Weise die positiven Eigenschaften der direkten Verfahren. Speziell die Routine NU-

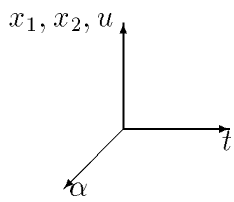


Abbildung 5.22: Koordinatensystem der drei dimensionalen Darstellungen

Abbildung 5.23: Das Auftreten von Nebenminima in Abhängigkeit von der Anfangsschätzung

	direkte Verfahren				ind. Verf.
Name des Verfahrens	NUDOCCCS	KOLLOKAT	DIROPTIM	DIRMESCH	BDSCO
Transformation des Ausgangsproblems (adj. Var., Transvers., ...)	nein	nein	nein	nein	ja
Speicheraufwand	mäßig	sehr groß	sehr groß	sehr groß	klein
Rechenaufwand	mäßig	groß	groß	sehr groß	klein
Genauigkeit	gut	gering	gering	gering	sehr gut
Konvergenzradius	groß	sehr groß	groß	groß	sehr klein
Möglichkeit des Auftretens von Pseudominima	ja	ja	ja	ja	nein
Möglichkeit der Aufnahme von Beschränkungen	ja	ja	ja	ja	math. Transf. erforderlich
Startschätzung der Steuerung benötigt	ja	ja	ja	ja	nein
Startschätzung der Zustandsvariablen benötigt	nein (bedingt)	ja	ja	ja	ja
Startschätzung der adjungierten Variablen benötigt	nein	nein	nein	nein	ja
Schaltstruktur muß bekannt sein	nein	nein	nein	nein	ja
Startschätzung der Schaltpunkte benötigt	nein	nein	nein	nein	ja
Möglichkeit zur Beschleunigung des Verfahrens	ja	ja	ja	ja	nein
Möglichkeit der Verbesserung der Lösungen	ja	ja	ja	ja	nein
Gefahr von Rundungsfehlern	gering	ja	ja	ja	nein
Mathematische Kenntnisse der Theorie erforderlich	nein	nein	nein	nein	ja
Besonderheiten	—	Startschätzung der adj. Var. kann ermittelt werden	—	—	—

Tabelle 5.12: Zusammenfassung der numerischen Ergebnisse

DOCCCS zeichnet sich durch eine hohe Genauigkeit des Zielfunktionals, einer Unempfindlichkeit gegenüber Rundungsfehlern, sowie durch eine hohe Rechengeschwindigkeit aus.
Der Verzicht auf eine Diskretisierung des Zustandes (woraus sich eine Reduzierung der Größe des Optimierungsvektors ergibt), sowie die Anpassung des Integrationsfehlers an den Interpolationsfehler der Steuerung hat sich als sehr sinnvoll erwiesen. Die Routine ist insbesondere für DOS-Rechner geeignet, da sie nur wenig Speicherplatz benötigt. Selbst dort sind Untersuchungen mit mehr als 100 diskreten Punkten möglich.
Die Genauigkeit der Ergebnisse reicht teilweise an die der Routine BDSCO heran, dabei wurden keinerlei Verbesserungstrategien berücksichtigt, mit denen die Ergebnisse noch weiter deutlich an Qualität gewinnen.
Auch die Rechengeschwindigkeit konnte auf einen, im Rahmen der Genauigkeit der Lösung, erträglichen Wert reduziert werden. Dabei können gerade die hier beschriebenen Verfahren noch deutlich verbessert werden. Zum einen helfen die aus der Diskretisierung folgenden speziellen Eigenschaften der verschiedenen Verfahren, zum anderen können auch allgemeine Ansätze, wie die in Abschnitt 4.4.4 beschriebene 'Schlauchtechnik', zu einer Senkung des Rechenaufwandes beitragen. Insbesondere die Verwertung von Informationen auf niedrigen Gitterstufen wird die Verfahren weiter beschleunigen können.

Beschreibt $\zeta_i^{(j)}$ den globalen Diskretisationsfehler auf einem Gitter der Stufe j, so wird durch

$$\Omega_i^{(j)} := \log_2 \left(\frac{\zeta_i^{(j)}}{\zeta_i^{(j+1)}} \right), \ i = 0, \ldots, 9 \tag{5.44}$$

eine Näherung der Konvergenzordnung der direkten Verfahren gegeben. Tabelle 5.13 zeigt einige Beispiele für das Spline-Problem mit der Zustandsbeschränkung $x_1 \leq 0.2$. Dieses Anwendungsbeispiel ist insbesondere für die lineare Interpolation der Steuerung geeignet, da der nicht stetig differenzierbare Intervallmittelpunkt schon auf einem Gitter unterster Stufe erreicht wird.

Es zeigt sich bei allen hier vorgestellten Beispielen, daß tatsächlich durch das Minimum der Konvergezordnungen der Interpolation der Steuerung und der Integration der Systemdynamik eine obere Schranke für die Konvergenzordnung des entsprechenden direkten Verfahrens gegeben ist. Die ermittelten Konvergenzordnungen vom Grade größer 1 scheinen dabei für praktische Anwendungen in der Regel ausreichend zu sein. Weiterhin konnten aber Wege aufgezeigt werden, diese Ordnung weiter zu vergrößern.
Die in Abbildung 5.5 beschriebene 'chattering-control' ist typisch für singuläre Probleme und Probleme mit linear auftretender Steuerung und aktiven Zustandsbeschränkungen. Da sie jedoch kaum Einfluß auf die optimale

j	$\Omega_1^{(j)}$	$\Omega_3^{(j)}$	$\Omega_5^{(j)}$
1	1.00	0.58	—
2	1.58	1.00	2.42
3	1.87	1.58	2.95
4	1.96	1.87	2.99
5	1.99	1.97	3.00
6	2.00	2.13	1.95
7	2.00	2.02	3.02
8	2.00	2.02	—

Tabelle 5.13: Konvergenzordnungen

Lösung haben, scheint eine Beseitigung nur von geringem Interesse zu sein.
Einziger wirklicher Nachteil der direkten Verfahren ist die Möglichkeit des Auftretens von Nebenminima, die insbesondere Auswirkungen auf die Kombination der direkten mit den indirekten Verfahren hat. Jedoch kann durch Variation der Anfangsschätzung der Optimierungsvariablen dieses Problem in vielen Fällen umgangen werden.
Tabelle 5.12 zeigt noch einmal eine Zusammenfassung der wesentlichen Ergebnisse.

Kapitel 6

Schlußbemerkung

In dieser Arbeit wurden verschiedene direkte Lösungsverfahren optimaler Steuerprozesse beschrieben und auf ihre speziellen Eigenschaften hin untersucht. Es konnten verschiedene Variationsmöglichkeiten aufgezeigt werden, die die unterschiedlichsten Modifikationen und damit Verbesserungen ermöglichen. Anhand der Auswertung verschiedener Anwendungsbeispiele und dem Vergleich mit der Lösung durch einen indirekten Ansatz, konnten Aussagen über Effizienz, Zuverlässigkeit und Güte der Lösung der direkten Verfahren gemacht werden.

Alle hier beschriebenen Verfahren sind gekennzeichnet durch die Rückführung eines (aus Sicht der nichtlinearen Optimierung) unendlich dimensionalen Steuerprozesses auf ein endlich dimensionales Optimierungsproblem.
In den letzten Jahren sind jedoch auch andere Verfahren bekannt geworden, wie z.B. direkte Verfahren, die Teilinformationen des indirekten Ansatzes ausnutzen ([Ziegler 1992], [Goh 1987]), direktes Lösen der optimalen Steuerprozesse mit genetischen Algorithmen ([Holland 1992]), Kombinationen des direkten Ansatzes mit den indirekten Verfahren (z.B. Hybride Methode, die theoretisch in Abschnitt 4.2.5 angedeutet wurde) oder Kombinationen mit symbolischen Algorithmen.
In der praktischen Anwendung kann es jedoch zu Problemen kommen, wie der Bestimmung einer Anfangsschätzung der adjungierten Variablen usw., hohen Rechenzeiten bei genetischen Algorithmen und auch die Hybride Methode benötigt Startschätzungen der adjungierten Variablen, was insbesondere dann zu Problemen führt, wenn das vorbereitende direkte Verfahren als 'Optimum' nur ein Nebenminimum findet.

Das Problem der 'Pseudominima' kann jedoch mit neueren Algorithmen der nichtlinearen Optimierung abgemildert werden, wie dies z.B. in [Dueck 1993] durch den Treshhold-Accepting- oder den Sintflut-Algorithmus geschieht.

Die ständigen Verbesserungen der den Zustand beschreibenden Differentialgleichungssysteme hat eine kontinuierliche Vergrößerung der Dimensionen des optimalen Steuerprozesses zur Folge. Mittlerweile sind Anwendungen bekannt, die aus über 50 Differentialgleichungen bestehen. Eine Lösung durch die direkten Verfahren, die eine Diskretisierung des Zustandes vorsehen (so auch KOLLOKAT, DIROPTIM und bedingt DIRMESCH), finden zu wollen, ist bei den heutigen Rechnerkapazitäten kaum möglich. Ebenso ist auch ein indirekter Ansatz ein vermutlich hoffnungsloses Unterfangen, da die Bestimmung einer Anfangsschätzung der resultierenden 100 Differentialgleichungen und die Aufstellung der adjungierten Differentialgleichungen kaum möglich ist. Die Routine NUDOCCCS ist jedoch in der Lage selbst auf DOS-Rechnern solche Probleme zu lösen. So konnte beispielsweise das bekannte 'Reentry-Problem', welches (nach einer Transformation der freien Endzeit) aus sechs Differentialgleichungen und einer Steuerfunktion besteht, auf einem MS/DOS-Rechner (ca. 550 KByte Arbeitsspeichre) mit weniger als 100 äquidistant gewählten Gitterpunkten und einem Fehler der Größenordnung ($O(10^{-5})$ mit der Routine NUDOCCCS ohne weitere Verbesserungsstrategien gelöst werden. Je größer das Differentialgleichungssystem ist, desto effizienter wird die Routine im Vergleich zu den anderen beschriebenen direkten Verfahren arbeiten.

Weiterhin zeigt sich deutlich, daß NUDOCCCS selbst ohne weitere Verbesserungsstrategien, Lösungen von großer Genauigkeit liefert. Während in praktischen Anwendungen eine Abweichung der berechneten Lösung vom exakten Minimum in einer Größenordnung von etwa 1% ($O(1 \cdot 10^{-2})$) als eine in der Regel ausreichende Näherung gesehen wird, unterschreitet NUDOCCCS diesen Wert bei weitem. Fehler von 0.000001% ($O(1 \cdot 10^{-8})$) konnten ohne weitere Verbesserungsstrategien erreicht werden. Damit gelangt man bereits in die Größenordnung der indirekten Verfahren und es besteht die berechtigte Hoffnung, daß dieser Wert weiter verringert werden kann.

Obwohl die Rechenzeit stark reduziert werden konnte, scheint die Aufnahme von Strategien zur Verbesserung der Güte der Lösung auf niedrigen Gitterstufen, sowie eine Beschleunigung des Verfahrens, eine für die Zukunft sinnvolle Erweiterung der vorhandenen Routine darzustellen.

Anhang A

Beschreibung der Subroutine BDSCO

A.1 Problemstellung

Die FORTRAN-Subroutine BDSCO wurde zur numerischen Behandlung von Aufgaben der optimalen Steuerung mit Zustandsbeschränkungen, beziehungsweise für Optimierungsaufgaben mit singulären Steuerungen und deren resultierenden Randwertproblemen mit Schaltbedingungen, entwickelt. Natürliche Schaltfunktionen zur numerischen Berechnung der Schaltpunkte können bei dieser speziellen Klasse von Problemen nicht verwendet werden, da sie entlang ganzer Teilstücke verschwinden oder Nullstellen höherer Ordnung besitzen.

Dieses, auf der Mehrzielmethode basierende Verfahren, stellt eine Weiterentwicklung der von Bulirsch entwickelten Routinen BOUNDSOL (für Zwei-Punkt-Randwertprobleme) und OPTSOL (für Randwertprobleme mit Schaltfunktionen) dar. BDSCO zeichnet sich gegenüber vielen anderen Verfahren durch eine große Rechengenauigkeit aus und ist vergleichsweise einfach anzuwenden.

Die Routine eignet sich ebenfalls zur Lösung klassischer Zwei-Punkt Randwertprobleme, ihre Stärke liegt jedoch in der numerischen Lösung der bereits angesprochenen Randwertprobleme mit Schaltbedingungen. Dabei werden die in derartigen Problemen auftretenden Schaltpunkte als unbekannte Parameter des Randwertproblems betrachtet und ebenfalls numerisch berechnet. Die Routine BDSCO löst konkret das folgende Randwertproblem mit Schaltbedingungen:

Gesucht seien

1. eine stückweise glatte Funktion

$$y : [x_0, x_f] \to \mathbb{R}^n,$$

2. Schaltpunkte ξ_i, $i = 1, \dots, m_s$, mit

$$x_0 =: \xi_0 < \xi_1 < \xi_2 < \dots < \xi_{m_s-1} < \xi_{m_s} < \xi_{m_s+1} := x_f,$$

mit folgenden Eigenschaften:

$$\begin{aligned}
y'(x) &= f_i(x, y(x)), && \xi_i \le x \le \xi_{i+1}, \quad i = 0, \dots, m_s && \text{(A.1)}\\
r_k(y(x_0), y(x_f)) &= 0, && k = 1, \dots, n_1 && \text{(A.2)}\\
r_k(\xi_{i_k}, y(\xi_{i_k}^-)) &= 0, && k = n_1 + 1, \dots, n + m_s && \text{(A.3)}\\
y(\xi_i^+) &= h_i(\xi_i, y(\xi_i^-)), && i = 1, \dots, m_s. && \text{(A.4)}
\end{aligned}$$

Das Differentialgleichungssystem (A.1) ist stückweise erklärt und in den Punkten $x = \xi_i$ und $x = \xi_{i+1}$ als links- beziehungsweise rechtsseitiger Grenzwert zu verstehen. Dabei sind die Schaltpunkte ξ_i, $i = 1, \dots, m_s$ unbekannt, die Schaltstruktur (das heißt die Reihenfolge der verschiedenen rechten Seiten in (A.1)) muß jedoch feststehen. Ein klassisches Zwei-Punkt-Randwertproblem ergibt sich formal durch den Fall $m_s = 0$. Durch (A.2) erhält man die üblichen Zwei-Punkt-Randbedingungen und über die $n + m_s - n_1$ Bedingungen in (A.3) Schaltbedingungen, also in Schaltpunkten zu erfüllende Bedingungen an die Trajektorie. (A.4) beschreibt schließlich Sprünge in der Trajektorie $y(x)$, die an den Schaltpunkten ξ_i auftreten können.

Neben einem aufrufenden Hauptprogramm sind vom Anwender zwei Unterroutinen zu erstellen, welche zur Auswertung der rechten Seite des Differentialgleichungssystems (A.1) und der Berechnung von Rand- und Schaltbedingungen (A.2), (A.3), sowie der Ausführung von Sprungbedingungen (A.4) dienen.

A.2 Die Parameterliste der Routine BDSCO

Die Subroutine BDSCO wird innerhalb eines FORTRAN-Programms wie folgt aufgerufen:

```
CALL BDSCO(F,R,METHOD,X,XS,Y,N,M,MS,KS,TOL,ITMAX,KP)
```

Die auftretenden Parameter haben dabei folgende Bedeutung:

`F`: Beim Aufruf von BDSCO muß `F` der Name der vom Anwender bereitzustellenden Unterroutine zur Berechnung der rechten Seite des Differentialgleichungssystems (A.1) sein.

`R`: Beim Aufruf von BDSCO muß `R` der Name der vom Anwender ebenfalls bereitzustellenden Unterroutine zur Auswertung der Rand- und Schaltbedingungen (A.2) bzw. (A.3), sowie zur Berechnung der Sprünge (A.4) sein.

`METHOD`: Beim Aufruf von BDSCO muß `METHOD` der Name einer Unterroutine zur numerischen Lösung eines Anfangswertproblems sein.

`X`: Ein reeller Vektor der Länge `M+MS`, der bei Aufruf von BDSCO mit den `M` Mehrzielknoten besetzt sein muß. Da BDSCO die Schaltpunkte unter Umständen in dieses Feld einordnet, ist dieser Vektor mit `M+MS` zu dimensionieren.

`XS`: Ein reeller Vektor der Länge `MS`. Beim Aufruf von BDSCO muß er eine Startschätzung für die `MS` Schaltpunkte enthalten. Nach Verlassen von BDSCO enthält dieser Vektor Daten für die berechneten Schaltpunkte der Lösung bzw. der letzten Iteration.

`Y`: Eine reelle Matrix des Größe (`N+MS,M+MS`). Die Feldelemente `Y(I,J)`, `I=1`, ..., `N`, `J=1`, ..., `M`, sind vor dem Aufruf der Routine mit einer Startschätzung für die `N` Komponenten der Trajektorie in jedem ihrer `M` Mehrzielknoten zu besetzen. Nach Verlassen der Routine enthält dieses Feld die Daten der Lösung bzw. die der letzten Iterierten.

`N`: Beim Aufruf von BDSCO muß der Integer-Parameter `N` die Anzahl der Differentialgleichungen (A.1) enthalten.

`M`: Beim Aufruf von BDSCO muß der Integer-Parameter `M` die Anzahl der Mehrzielknoten enthalten. Sollte `KS=1` gewählt werden (s.u.), so wird `M` mit dem Wert `M+MS` überschrieben.

`MS`: Beim Aufruf von BDSCO muß der Integer-Parameter `MS` die Anzahl der Schaltpunkte enthalten.

`KS`: Integerparameter, mit welchem bestimmt wird, ob die Schaltpunkte nach Beendigung der Iteration in die Menge der Mehrzielknoten eingeordnet werden sollen. Dabei bedeutet die Wahl von

`KS=0`: Keine Veränderung der Mehrzielknoten. Diese Wahl empfiehlt sich bei der Verwendung von BDSCO in einer Homotopie.

`KS=1`: Bei Beendigung der Routine sind die errechneten Daten für die Schaltpunkte in die Menge der vorgegebenen Mehrzielknoten ihrer Größe nach eingeordnet, so daß die Matrix der Trajektoriendaten `Y(I,J)`, `I=1`, ..., `N`, `J=1`, ..., `M+MS`, um die Daten in den Schaltpunkten erweitert ist.

`TOL`: Reelle Genauigkeitsschranke als Abbruchkriterium für die Newton-Iteration. Bei regulärer Beendigung der Routine besitzen die Lösungsdaten die relative Genauigkeit `TOL`.

`ITMAX`: Durch den Integer-Parameter `ITMAX` wird die Anzahl der maximal erlaubten Newton-Iterationen festgelegt. Das Setzen von `ITMAX=0` veranlaßt lediglich die Berechnung einer „Grundtrajektorie“.

`KP`: Integerparameter zur Steuerung der Ausgabe. Dabei bedeutet die Setzung von

`KP=-1`: Keine Ausgabe.

`KP=0`: Ausgabe der Startdaten, der Testfunktionen jeder Iteration, sowie die Ausgabe der Lösungsdaten bzw. die der letzten Iterierten.

`KP=1`: Zusätzliche Ausgabe der Iterierten `X`, `Y` und `XS`, sowie der Daten in den Schaltpunkten für jede Iteration.

Nach Verlassen von BDSCO haben die Werte von `KP` folgende Bedeutung:

Bei einem regulären Abbruch der Mehrzieliteration beinhaltet der Parameter `KP` die Anzahl der ausgeführten Iterationen, die zum Erhalt der Lösung notwendig waren. Beinhaltet `KP` nach Beendigung der Routine einen negativen Wert, so deutet das auf ein Scheitern des Mehrzielverfahrens hin. Der jeweilige negative Wert von `KP` stellt dabei eine Kennziffer für die Art des Fehlausgangs dar:

`KP=-1`: Abbruch des Verfahrens an einem stationären Punkt.

`KP=-2`: Abbruch des Verfahrens nach `ITMAX` Iterationen.

KP=-3: Abbruch aufgrund Scheiterns der numerischen Integration eines Anfangswertproblems.

KP=-4: Abbruch, da das interne Newton-Verfahren nicht konvergiert.

KP=-5: Abbruch bei der numerischen Differentiation wegen benachbarter singulärer Trajektorien.

KP=-6: Abbruch aufgrund singulärer Mehrzielmatrix.

KP=-7: Inkonsistente Schätzung der Schaltpunkte.

KP=-8: Inkonsistente Schätzung der Startdaten.

Die Steuerung der verschiedenen rechten Seiten des Differentialgleichungssystems (A.1), die Ausführung der Sprungbedingungen, sowie die Unterscheidung zwischen Rand- und Schaltbedingungen erfolgt mit Hilfe des benannten COMMON-Blocks

```
COMMON /CI/ J,L,LS,US.
```

Da ferner über den Integer-Parameter J auch Informationen vom aufrufenden Programm an die Routine BDSCO gegeben und zurückgegeben werden, hat dieser Block sowohl im aufrufenden Programm, als auch in den beiden vom Anwender bereitzustellenden Unterprogrammen F und R zu stehen. Es bedeuten hierbei

J: Während der numerischen Lösung der Anfangswertprobleme gibt dieser Integer-Parameter das Integrationsintervall an, wobei so gezählt wird, als wären die Schaltpunkte ihrer Größe nach in die Menge der Mehrzielknoten eingeordnet. Der Parameter ist nur von interner Bedeutung, wird also vom Anwender nicht unmittelbar zur Steuerung der rechten Seite des Differentialgleichungssystems angewandt, da sich während der Iteration die Lage der Schaltpunkte gegenüber den Mehrzielknoten ändern kann.

L: Integer-Parameter, der die Nummer des jeweiligen Mehrzielintervalls angibt (ohne Berücksichtigung der Schaltpunkte).

LS: Durch den Integer-Parameter LS wird die Ausführung von Sprungbedingungen in dem vom Anwender bereitzustellenden Unterprogramm R gesteuert.
Die Werte von LS haben dabei folgende Bedeutung:

LS=-1: Die Unterroutine R wird zur Besetzung der Integervektoren NYA und NSK (vor Beginn der Mehrzieliteration) aufgerufen.

- `LS=0`: Aufruf der Unterroutine `R` im Anfangs- oder Endpunkt zur Berechnung der Randbedingungen.
- `LS=K`: Die Unterroutine `R` wird im Schaltpunkt `XS(K)` aufgerufen, um Schaltbedingungen zu berechnen oder eine Sprungbedingung auszuführen.

`US`: Integermatrix der Dimension (`M+MS,MS`). Mit ihrer Hilfe steuert man die rechte Seite des Differentialgleichungssystems (A.1) im Unterprogramm `F`.
Wird die Routine `F` im Punkt `X` aufgerufen, so ist `US(J,K)` folgendermaßen besetzt:

$$\mathtt{US(J,K)}=\begin{cases} -1, & \text{falls } \mathtt{X} \leq \mathtt{XS(K)} \\ 1, & \text{falls } \mathtt{X} > \mathtt{XS(K)} \end{cases}, \mathtt{K} = 1, \ldots, \mathtt{MS}.$$

A.3 Vom Anwender bereitzustellende Unterprogramme

Die ersten drei an BDSCO zu übergebenden Parameter sind Namen von Unterprogrammen, die eine interne Bedeutung besitzen. Es handelt sich dabei zunächst um die beiden problemabhängigen Unterroutinen `F` und `R`, die zur Berechnung der rechten Seite des Differentialgleichungssystems, der Berechnung von Rand- beziehungsweise Schaltbedingungen und der Ausführung eventueller Sprungbedingungen dienen.

A.3.1 Die Subroutine `F`

Die Routine `F` zur Berechnung der rechten Seite des Differentialgleichungssystems (A.1) muß folgendermaßen aufgerufen werden:

```
CALL F(X,Y,DY).
```

Dabei bedeuten

`X`: unabhängige reelle Variable (nach der differenziert wird).

`Y`: reeller Vektor der Länge `N`, dessen Komponenten die abhängigen Variablen darstellen.

`DY`: reeller Vektor der Länge `N`, der beim Verlassen von `F` die Werte der rechten Seite des Differentialgleichungssystems (A.1) enthalten muß. Die Steuerung der rechten Seite in Bezug auf das Schaltintervall geschieht dabei durch das Integerfeld `US(J,K)`. Da das

Differentialgleichungssystem stückweise erklärt ist, ist der Vektor `DY` in Abhängigkeit davon zu berechnen, in welchem Schaltintervall sich die unabhängige Variable `X` befindet.

A.3.2 Die Subroutine `R`

Das Unterprogramm `R` zur Berechnung der Randbedingungen (A.2), der Schaltbedingungen (A.3), sowie zur Ausführung eventueller Sprünge der Trajektorie (A.4), wird folgendermaßen aufgerufen:

```
CALL R(YA,YB,ZZ,W,NYA,NSK).
```

Dabei haben die Parameter folgende Bedeutung:

`YA`: reeller Vektor der Länge `N+MS`. Beim Aufruf von `R` enthalten seine ersten `N` Komponenten die Werte der abhängigen Variablen im Punkte $x = x_0$, also die Anfangswerte.

`YB`: reeller Vektor der Länge `N+MS`. Beim Aufruf von `R` enthalten seine ersten `N` Komponenten die Werte der abhängigen Variablen im Punkte $x = x_f$, also die Endwerte.

`ZZ`: reeller Vektor der Länge `N+MS`. Beim Aufruf von `R` im Schaltpunkt `XS(K)` enthalten die ersten `N` Komponenten von `ZZ` die Werte der abhängigen Variablen im Punkte `XS(K)` (linksseitiger Grenzwert!). Beim Verlassen dieser Routine müssen dagegen die rechtsseitigen Grenzwerte auf diesen Speicherplätzen stehen (bei Ausführung von Sprungbedingungen).

`W`: reeller Vektor der Länge `N+MS`. Beim Verlassen von `R` muß dieser Vektor die Werte der Rand- und Schaltbedingungen in beliebiger Reihenfolge enthalten. Die Randbedingungen werden dabei mit Hilfe von `YA` und `YB`, die Schaltbedingungen mit Hilfe von `ZZ` berechnet.

`NYA`: Integervektor der Länge `M+MS`, der zur Festschreibung von Anfangsbedingungen dient. Unterliegt die Komponente `Y(I)` der Trajektorie einer bestimmten Anfangsbedingung, so ist die entsprechende Randbedingung `W(J)` mit Hilfe der zusätzlichen Anweisung `NYA(I)=J` zu kennzeichnen.

`NSK`: Integervektor der Länge `N+MS`, der zur Markierung von Schaltbedingungen dient. Eine Schaltbedingung `W(J)` für den Schaltpunkt `XS(K)` ist durch die Anweisung `NSK(J)=K` zu ergänzen.

In den übrigen `MS` Komponenten der Felder `YA`, `YB` und `ZZ` stehen die jeweiligen Werte der Schaltpunkte, z.B.

$$\texttt{ZZ(N+I)=XS(I)}, \qquad \texttt{I} = 1, \ldots, \texttt{MS}.$$

Eventuelle Sprünge in der Trajektorie werden dementsprechend innerhalb dieser Routine wie folgt ausgeführt. Zunächst ist die Ausführung der Sprungbedingungen (A.4) an die Bedingung `LS=K` zu koppeln, um sicherzustellen, daß sie auch nur beim Aufruf von `R` im Punkte `XS(K)` durchgeführt werden. In diesem Fall ist der linksseitige Grenzwert jeder abhängigen Variable am jeweiligen Schaltpunkt in den ersten `N` Komponenten des Feldes `ZZ` gespeichert. Der Schaltpunkt `XS(K)` ist mit `ZZ(N+K)` gegeben. Mit diesen Daten läßt sich eine Sprungbedingung (A.4) ausführen. Der damit berechnete rechtsseitige Grenzwert in einer bestimmten, von einer Sprungbedingung betroffenen Komponente `I` der Trajektorie, ist dann vor Verlassen der Routine der Variablen `ZZ(I)` zuzuweisen.

A.3.3 Die Subroutine `METHOD`

Zur Lösung von Anfangswertproblemen hat der Anwender ebenfalls eine Routine zur Verfügung zu stellen (Dritter Eintrag in der Parameterliste von BDSCO). Diese Routine muß dabei folgendermaßen aufgerufen werden:

```
CALL METHOD(N,F,X,Y,XEND,TOL,HMAX,H).
```

Unter Umständen muß eine vorhandene Integrationsroutine – etwa aus der NAG- oder IMSL-Bibliothek – erst auf diese Gestalt gebracht werden, indem sie innerhalb einer Adapterroutine verwendet wird. Ihre Parameter haben die Bedeutung

`N`: Integer-Parameter zur Festlegung der Anzahl der Differentialgleichungen. `N` ist ein reiner Eingangsparameter.

`F`: Name des Unterprogramms zur Berechnung der rechten Seite des Differentialgleichungssystems (s.o.).

`X`: unabhängige Variable, die einen Eingangs- und Ausgangsparameter darstellt.

`Y`: reeller Vektor der Länge `N`, der die abhängigen Variablen enthält und ebenfalls einen Eingangs- und Ausgangsstatus besitzt.

`XEND`: Eingangsparameter zur Festlegung des Endes des Integrationsintervalls.

`TOL:` Eingangsparameter zur Festlegung der oberen Schranke für den lokalen Diskretisierungsfehler.

`HMAX:` Eingangsparameter zur Festlegung der maximalen Schrittweite.

`H:` gibt die Anfangsschrittweite an und besitzt Eingangs- und Ausgangsstatus. Bricht die numerische Integration vor Erreichen des Zielpunktes `XEND` ab, so erwartet BDSCO den Wert `H=0`.

A.4 Anmerkungen

Die Wahl der Felddimensionen bei Übergabe von Matrizen als Aktualparameter birgt eine Gefahr in sich. Einige FORTRAN-Compiler geben während des Übersetzens keine Fehlermeldung aus, falls eine gegenüber der Deklaration in der Unterroutine anders dimensionierte Matrix an die entsprechende Routine übergeben wird. Das Programm kann dann unter Umständen fehlerfrei durchlaufen, liefert aber falsche Ergebnisse. Es ist also darauf zu achten, daß Matrizen als Übergabeparameter im aufrufenden Programm und im Unterprogramm von gleichem Format sind.

Anhang B

Beschreibung der Subroutine NUDOCCCS

B.1 Problemstellung

Die FORTRAN-Subroutine NUDOCCCS (Numerical Discretisationmethod for Optimal Controlproblems with Constraints in Controls and States) dient der direkten numerischen Lösung optimaler Steuerprozesse mit Zustands- und Steuerbeschränkungen. Das Verfahren basiert auf der Diskretisierung des Ausgangsproblems und einer Transformation in ein nichtlineares Optimierungsproblem, zu dessen Lösung die FORTRAN-Subroutine OPRQP hinzugezogen wird. Genauer teilt sich das Verfahren in zwei Unterprogramme auf, wobei das eine die diskretisierte und konstant oder linear interpolierte Steuerung, fehlende Anfangsbedingungen und die unbekannten Mehrzielknoten, die zur Erweiterung des Konvergenzradiusses eingeführt wurden, unter verschiedenen Beschränkungen zu verbessern sucht, während mit dem anderen Unterprogramm die Trajektorie und damit wichtige Beschränkungen durch ein Runge-Kutta-Integrationsverfahren mit relativ großer Schrittweite berechnet werden. Dabei zeichnet sich die Routine NUDOCCCS im Vergleich zu anderen Verfahren besonders durch den geringen Speicheraufwand, die große Rechengeschwindigkeit und hohe Genauigkeit des Zielfunktionals, der optimalen Trajektorie und der optimalen Steuerung, sowie einer besonders einfachen Anwendung und Benutzerfreundlichkeit gegenüber anderen direkten Verfahren aus.
An dieser Stelle sei noch einmal auf die allgemeine Problemstellung eines optimalen Steuerprozesses hingewiesen.

$$
\begin{array}{lrcll}
\text{Minimiere} & J[u] & = & \Phi(x,u,t) & \\
\text{unter} & & & & \\
& \dot{x} & = & f(t,x,u), & t_0 \leq t \leq t_f \\
& \Psi(x(t_0),t_0,x(t_f),t_f) & = & 0 & \\
& S_i(x,u,t) & = & 0, & i = 1,\ldots,k \\
& S_i(x,u,t) & \geq & 0, & i = k+1,\ldots,r
\end{array}
\tag{B.1}
$$

Dieses iterative Verfahren läuft unter der Annahme, daß die Funktionen Φ, Ψ und $S_i,\quad i = 1,\ldots,r$ stetig differenzierbar nach u sind. Außerdem sollte das Funktional J auch dann an gewissen Punkten berechenbar sein, wenn die Nebenbedingungen noch nicht erfüllt sind.
Um NUDOCCCS für Probleme der Art (B.1) benutzen zu können, müssen neben dem Hauptprogramm noch für folgende Zwecke fünf Unterroutinen vom Anwender bereitgestellt werden:

1. Berechnung des Zielfunktionals,

2. Berechnung der rechten Seite des Differentialgleichungssystems,

3. Berechnung der Anfangswerte des Differentialgleichungssystems,

4. Auswertung der punktuellen Gleichheitsnebenbedingungen (z.B. Endwerte des Differentialgleichungssystems),

5. Auswertung der Nebenbedingungen in Gleichungs- und Ungleichungsform.

Außerdem kann bei Bedarf die Routine CALGRD umgeschrieben werden, die den Gradienten des Zielfunktionals und aller Nebenbedingungen berechnet. Zur Zeit wird die Ableitung numerisch mittels des zentralen Differenzenquotienten ermittelt. Die numerische Berechnung der Ableitungen wurde gewählt, da eine analytische Berechnung in der Regel nur sehr umständlich und sehr zeitaufwendig durchgeführt werden kann. Sollte es dennoch möglich sein, die Ableitungen auf konventionellem Wege zu berechnen, so ist diese Methode der numerischen Differentiation vorzuziehen, da sie genauere Werte liefert und weniger rechenintensiv ist.
Unabhängig von den unten beschriebenen benannten COMMON-Blöcken (WA, WB, WC), die durch den zur Verfügung stehenden Speicherplatz des Rechners bestimmt werden, gibt es noch einige weitere Einschränkungen bezüglich der Anzahl der diskreten Punkte, der Differentialgleichungen, der Steuerfunktionen und der Nebenbedingungen, die im folgenden Abschnitt näher beschrieben werden.

Sollte eine Erweiterung dieser Grenzen erforderlich sein, so müssen die entsprechenden Parameter (s.u.) in den verantwortlichen Prozeduren den Problemanforderungen angepaßt werden.
An dieser Stelle sei noch erwähnt, daß die Namen der vom Anwender zu schreibenden Unterroutinen nicht in die Parameterliste von NUDOCCCS aufgenommen wurden, um diese nicht noch unnötig weiter zu vergrößern, d.h., daß die Namen dieser Routinen nicht modifizierbar sind und wie in der Beschreibung (s.u.) definiert werden müssen. Die lange Parameterliste wurde einer Unzahl von COMMON-Blöcken vorgezogen um die Routine anwenderfreundlicher und speicherplatzreduzierend zu gestalten.

B.2 Die NUDOCCCS-Parameterliste

Die Subroutine NUDOCCCS muß innerhalb eines FORTRAN-Programms folgendermaßen aufgerufen werden:

```
   CALL  NUDOCCCS( nDGL, nSteuer, ndiskret, nunbe, nneben, nuglnb,
 c                 nrand, nzusatz, nart, iprint, imax, istop, scale,
 c                 r1, del, n, m, me, maxnm, x, u, f, ff, g, uh, iv, ivset,
 c                 eps, t, h, unbe, uhelp, y, z, c, p, oldf, beta, atg, z1,
 c                 g1, g2, diag, nStuetz, msDGL, msStuetz)
```

Für Variablen vom Typ Real steht bei älteren Versionen von NUDOCCCS nur REAL*4 zur Verfügung. Die neueren Versionen ab 5.01 wurden auf DOUBLE PRECISION umgeschrieben, so daß eine höhere Genauigkeit erreicht werden kann.
Die einzelnen Parameter haben die folgende Bedeutung:

`nDGL`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der Differentialgleichungen enthalten.
Es muß gelten:

$$1 \leq \texttt{nDGL} \leq \texttt{dgl}$$

für den Parameter `dgl`, der am Ende dieser Auflistung beschrieben wird.

`nSteuer`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der Steuerfunktionen enthalten.
Es muß gelten:

$$1 \leq \texttt{nSteuer} \leq \texttt{ste}$$

für den Parameter `ste`, der am Ende dieser Auflistung beschrieben wird.

`ndiskret`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der (diskreten) Stützstellen enthalten. Es muß gelten:

$$1 \leq \texttt{ndiskret} \leq \texttt{dis}$$

für den Parameter `dis`, der am Ende dieser Auflistung beschrieben wird.

`nunbe`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der Differentialgleichungen enthalten, bei denen zusätzliche Mehrzielknoten eingeführt werden sollen. Anwendungsbeispiel: Setze `nunbe` auf die Anzahl der fehlenden Anfangswerte und $\texttt{nStuetz} = 1$ (s.u.) für das einfache Schießverfahren. Aus Dimensionierungsgründen muß `nunbe` eine Mindestgröße von 1 besitzen.
Es muß gelten:

$$1 \leq \texttt{nunbe} \leq \texttt{nDGL}$$

`nStuetz`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der zusätzlichen Mehrzielknoten enthalten. Aus Dimensionierungsgründen muß `nStuetz` eine Mindestgröße von 1 besitzen.
Es muß gelten:

$$1 \leq \texttt{nStuetz} \leq \texttt{ndiskret}$$

`nneben`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß sie die Anzahl aller Nebenbedingungen (Ungleichungs- und Gleichungsnebenbedingungen) enthalten, die sich auf das ganze zu bearbeitende Intervall beziehen.
Es muß gelten:

$$0 \leq \texttt{nneben} \leq \texttt{neb}$$

für den Parameter `neb`, der am Ende dieser Auflistung beschrieben wird.

`nuglnb`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl Ungleichungsnebenbedingungen enthalten, die sich auf das ganze zu bearbeitende Intervall beziehen.
Es gilt:

$$0 \leq \texttt{nuglnb} \leq \texttt{nneben}$$

`nrand`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der punktuellen Nebenbedingungen in Gleichungsform enthalten, wie sie zum Beispiel durch die evtl. Endbedingungen des optimalen Steuerprozesses beschrieben werden. Anfangswerte werden nicht von diesem Parameter als Nebenbedingungen mitgezählt, da sie durch die Parameter `nDGL` und `nunbe` ermittelbar sind. Es muß gelten:

$$0 \leq \texttt{nrand} \leq \texttt{nDGL}$$

`nzusatz`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl aller zusätzlichen Optimierungsvariablen enthalten.
Es gilt:

$$\texttt{nzusatz} = \texttt{nunbe} * \texttt{nStuetz}$$

`nart`: Eingangsparameter vom Typ Integer zur Festlegung der Art der Interpolation der Steuerung und des Verfahrens für die Integration des Differentialgleichungssystems.

Nummer	Integr.Verf.	ξ_1	Intpol.Verf.	ξ_2	$\min(\xi_1, \xi_2)$	RSA
0	Poly.	$O(h)$	—	—	$O(h)$	1
1	Heun	$O(h^2)$	—	—	$O(h^2)$	2
2	verb.Poly.	$O(h^2)$	konstant	$O(h)$	$O(h)$	2
3	verb.Poly.	$O(h^2)$	linear	$O(h^2)$	$O(h^2)$	2
4	RK 4	$O(h^4)$	linear	$O(h^2)$	$O(h^2)$	4
5	RK 4	$O(h^4)$	kubisch	$O(h^4)$	$O(h^4)$	4
6	RK 5	$O(h^5)$	kubisch	$O(h^4)$	$O(h^4)$	6

Tabelle B.1: Realisierte Kombinationen

Erläuterungen:

- ξ_1 Fehlerordnung des Integrationsverfahrens,
- ξ_2 Fehlerordnung des Interpolationsverfahrens,
- RSA Auswertungen der rechten Seite des DGL-systems.

Beim Aufruf der Routine darf er die Werte $0, \ldots, 6$ enthalten, die dann folgende Bedeutung haben (vgl. auch Tabelle B.1):

`nart` = 0: Die Steuerfunktionen $u(t)$ brauchen nicht interpoliert zu werden, denn das Differentialgleichungssystem wird mit dem Eulerschen Polygonzugverfahren der Fehlerordnung 1 ausgewertet.

`nart` = 1: Die Steuerfunktionen $u(t)$ brauchen nicht interpoliert zu werden, denn das Differentialgleichungssystem wird mit dem Verfahren von Heun der Fehlerordnung 2 ausgewertet.

`nart` = 2: Die Steuerfunktionen $u(t)$ werden durch stückweise konstante Funktionen approximiert. Das Differentialgleichungssystem wird mit dem verbesserten Polygonzugverfahren der Fehlerordnung 2 ausgewertet.

`nart` = 3: Die Steuerfunktionen $u(t)$ werden durch stückweise linear interpolierte Funktionen approximiert. Das Differentialgleichungssystem wird mit dem verbesserten Polygonzugverfahren der Fehlerordnung 2 ausgewertet.

`nart` = 4: Die Steuerfunktionen $u(t)$ werden durch stückweise linear interpolierte Funktionen approximiert. Das Differentialgleichungssystem wird mit einem Runge-Kutta-Verfahren der Fehlerordnung 4 ausgewertet.

`nart` = 5: Die Steuerfunktionen $u(t)$ werden durch stückweise kubisch interpolierte Funktionen approximiert. Das Differentialgleichungssystem wird mit einem Runge-Kutta-Verfahren der Fehlerordnung 4 ausgewertet.

`nart` = 6: Die Steuerfunktionen $u(t)$ werden durch stückweise kubisch interpolierte Funktionen approximiert. Das Differentialgleichungssystem wird mit einem Runge-Kutta-Verfahren der Fehlerordnung 5 ausgewertet.

Bei Problemen mit bang-bang oder singulärer Steuerung hat sich die stückweise konstante Approximation bezüglich der graphischen Darstellung der Funktionen als günstiger erwiesen, auch wenn der Funktionalwert für den Fall der stückweise linear interpolierten Funktion genauer berechnet wird.

`iprint`: Eingangsparameter vom Typ Integer zur Steuerung der Ausgabe auf dem Bildschirm. Dabei haben die Wert von `iprint` folgende Bedeutung:

`iprint` < 0: Jegliche Ausgabe auf dem Bildschirm wird unterdrückt.

`iprint` = 0: Nur zum Abschluß der Berechnungen gibt es eine Ausgabe.

`iprint` > 0: Alle `iprint` Iterationen und zum Abschluß wird eine Ausgabe für eine Schätzung der Lagrange-Multiplikatoren, der Lösung der Steuerfunktionen, der zusätzlichen Optimierungsvariablen, des Zielfunktionals und aller Nebenbedingungen (einschließlich der punktuellen Nebenbedingungen und der Mehrzielknoten), veranlaßt.

`imax`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der maximal zu berechnenden Iterationen enthalten. Nach Erreichen der oberen Grenze bricht die Berechnung mit einer eventuellen Ausgabe (vgl. `iprint`) ab.

`istop`: Ausgangsparameter vom Typ Integer. Beinhaltet `istop` beim Verlassen der ROUTINE einen nicht-negativen Wert, so wird eine von fünf möglichen Meldungen ausgegeben, die das Abbruchkriterium enthält. Die fünf verschiedenen Werte und deren Bedeutung, beziehungsweise deren daraus resultierenden Abbruchmeldungen lauten:

`istop` = 1: `SEARCH DIRECTION UPHILL` zeigt an, daß das Verfahren keine weiteren Verbesserungen erreichen kann. Dies kann bedeuten, daß die optimale Lösung erreicht wurde, aber die durch den Vektor `eps` geforderte Genauigkeit nicht eingehalten werden konnte.

`istop` = 2: `LAGRANGE MULTIPLIER EQUATIONS SINGULAR` bedeutet, daß einige unvereinbare Beschränkungen gefunden wurden, zum Beispiel, wenn mehr Beschränkungen erfüllt werden müssen, als es Freiheitsgrade des Problems gibt. In der Praxis tritt diese Fehlermeldung häufig dann auf, wenn bei der Implementation eines Problems Eingabefehler gemacht wurden, oder sie kann ein Hinweis darauf sein, daß die optimale Steuerung singuläre Bereiche besitzt. Weiterhin zeigt diese Meldung oftmals an, daß der numerische Differenzenquotient aufgrund von Rundungsfehlern nicht hinreichend genau berechnet werden kann.

`istop` = 3: `MORE THAN IMAX ITERATIONS` zeigt an, daß die vom Anwender vorgegebene obere Schranke `imax` der Iterationen erreicht wurde.

`istop` = 4: `NO FURTHER REDUCTION IN FUNCTION OR CONSTRAINTS` zeigt an, daß das Verfahren keine weiteren Verbesserungen erreichen kann. Dies kann bedeuten, daß die optimale Lösung erreicht wurde, aber die durch den Vektor `eps` geforderte Genauigkeit nicht eingehalten werden konnte.

`istop` = 5: `CONVERGED` bedeutet, daß ein lokales Minimum mit der durch den Vektor `eps` geforderten Genauigkeit gefunden wurde.

`iv`: Hilfsparameter vom Typ Integer, der die Anzahl aller aktiven Nebenbedingungen enthält.

`n`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl aller (diskretisierten) Optimierungsvariablen enthalten.
Es gilt:

$$\mathtt{n} = \mathtt{ndiskret} * \mathtt{nSteuer} + \mathtt{nzusatz}$$

`m`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl aller (diskretisierten) Nebenbedingungen enthalten.
Es gilt:

$$\mathtt{m} = \mathtt{ndiskret} * \mathtt{nneben} + \mathtt{nrand} + (\mathtt{nunbe} - \mathtt{1}) * \mathtt{nStuetz}$$

`me`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl aller (diskretisierten) Gleichungsnebenbedingungen enthalten. Man beachte, daß die Reihenfolge der Nebenbedingungen so gestaltet wird, daß die Gleichungsnebenbedingungen vor den Ungleichungsnebenbedingungen stehen.
Es gilt:

$$\mathtt{me} = \mathtt{m} - \mathtt{ndiskret} * \mathtt{nuglnb}$$

`maxnm`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er das Maximum von `n` und `m` enthalten.
Es gilt:

$$\mathtt{maxnm} = \mathtt{m} + (2 * (\mathtt{n}/\mathtt{m})/(\mathtt{n}/\mathtt{m} + 1)) * (\mathtt{n} - \mathtt{m})$$

Diese Formel gilt nur für FORTRAN-Compiler.

`r1`: Parameter vom Typ Real. Beim Aufruf der Routine muß er den Anfangswert des Fehlerparameters `R` enthalten, der in folgender Fehlerfunktion Verwendung findet:

$$p(W,R) = J[W] + \frac{1}{R}\sum_{i=1}^{k}(S_i(W))^2 + \frac{1}{R}\sum_{i=k+1}^{r}(\min(0,S_i(W)))^2$$

W beschreibt den Vektor `uhelp`, der alle Optimierungsvariablen enthält.

Eine typische Wahl ist `r1` = `1.0`, aber für einige Anwendungen hat sich ein kleinerer Wert (zum Beispiel: 0.1 oder 0.01) als günstiger erwiesen.
Beim Verlassen von NUDOCCCS enthält `r1` den letzten Wert des Fehlerparameters.

`scale`: Eingangsparameter vom Typ Real. Beim Aufruf der Routine muß er den Faktor enthalten, um den der Fehlerparameter bei jedem Iterationsschritt verkleinert werden soll. Wenn der optimale Steuerprozeß nichtlinear in $u(t)$ ist, so empfiehlt sich die Wahl:

$$0.25 \leq \texttt{scale} \leq 0.75$$

Bei linearen Nebenbedingungen kann `scale` bis auf den Wert 0.1 reduziert werden.
Zur schnelleren Bestimmung der Lösung kann auch folgende Strategie von Bedeutung sein: Bestimme eine Lösung für den Wert `scale = 0.1` und nutze die so erhaltenen Daten für Steuerung, weitere Optimierungsvariablen, Gradient, Jacobi- und Hesse-Matrix für eine genauere Berechnung mit $0.25 \leq \texttt{scale} \leq 0.75$.

`del`: Eingangsparameter vom Typ Real. Beim Aufruf der Routine muß er die Schrittweite enthalten, mit der die numerische Differentiation durch den Vorwärts- oder Zentraldifferenzenquotienten durchgeführt werden soll. Es bedeuten :

$$\begin{array}{ll} \texttt{del} < 0: & \text{Zentraldifferenzenquotient} \\ \texttt{del} > 0: & \text{Vorwärtsdifferenzenquotient} \end{array}$$

Bei sehr hoher Diskretisierungsrate ($\texttt{NDISKRET} > 70$) ist es oftmals zu empfehlen, den Zentraldifferenzenquotienten (Fehlerordnung $O(\texttt{DEL}^2)$) zu benutzen, weil die Ableitungen sonst nicht mit ausreichender Genauigkeit berrechnet werden können.
Die Rechendauer des Zentraldifferenzenquotienten ist etwa doppelt so lang wie beim Vorwärtsdifferenzenquotienten (Fehlerordnung $O(\texttt{DEL})$).

`ff`: Parameter vom Typ Real. Beim Verlassen der Routine enthält `ff` den Wert des berechneten Zielfunktionals $J[W]$.

`ivset`: Vektor der Länge `m` vom Typ Integer, dessen ersten `iv` Elemente die Indizes der aktiven Nebenbedingungen enthalten.

`msdgl`: Vektor der Länge `nunbe` vom Typ Integer. Beim Aufruf der Routine muß er die Indizes der Differentialgleichungen in aufsteigend sortierter Reihenfolge enthalten, für die Mehrzielknoten bereitgestellt werden sollen.

`msStuetz`: Vektor der Länge `nStuetz` vom Typ Integer. Beim Aufruf der Routine muß er die Indizes der diskreten Zeitpunkte in aufsteigend sortierter Reihenfolge enthalten, an denen die Mehrzielknoten bereitgestellt werden sollen. Dabei ist zu beachten, daß der jeweils erste Mehrzielknoten grundsätzlich dem Anfangszeitpunkt zugeordnet wird. Eine eigene Zuordnung ist demnach nicht erforderlich.

`t`: Vektor der Länge `ndiskret` vom Typ REAL. Beim Aufruf der Routine muß er die Werte der nicht zwangsweise äquidistant gewählten diskreten Unterteilung des Optimierungsintervalls enthalten.

`h`: Vektor der Länge `ndiskret-1` vom Typ Real, der während der Abarbeitung den Abstand der `t(i)` zueinander enthält.
Es gilt:

$$\mathtt{h(i)} = \mathtt{t(i+1)} - \mathtt{t(i)}$$

`g`: Vektor der Länge `m` vom Typ Real. Während der Abarbeitung der Routine werden in diesen Vektor alle (diskretisierten) Nebenbedingungen eingeordnet, dabei gilt folgende Reihenfolge:
Für

$$\begin{aligned} j &= 2,\ldots,\mathtt{nStuetz} \\ i_r &= 1,\ldots,\mathtt{nrand} \\ i_d &= 1,\ldots,\mathtt{ndiskret} \end{aligned}$$

gilt

$$\begin{aligned} \mathtt{g(i)} = \;& x_l(j)_- - x_l(j)_+ \\ & \mathtt{i} = 1,\ldots,\mathtt{nunbe} * (\mathtt{nStuetz} - 1) \\ \mathtt{g(i)} = \;& \Psi(x(t_{i_r}), t_{i_r}) = 0 \\ & \mathtt{i} = \mathtt{nunbe} * (\mathtt{nStuetz} - 1) + 1,\ldots \\ & \qquad \ldots,\mathtt{nunbe} * (\mathtt{nStuetz} - 1) + \mathtt{nrand} \\ \mathtt{g(i)} = \;& S_i(x(t_{i_d}), u(t_{i_d}), t_{i_d}) = 0 \\ & \mathtt{i} = \mathtt{nunbe} * (\mathtt{nStuetz} - 1) + \mathtt{nrand} + 1,\ldots,\mathtt{me} \\ \mathtt{g(i)} = \;& S_i(x(t_{i_d}), u(t_{i_d}, t_{i_d}) \geq 0 \\ & \mathtt{i} = \mathtt{me} + 1,\ldots,\mathtt{m} \end{aligned}$$

Dabei ist mit $x_l(j)_-$, beziehungsweise $x_l(j)_+$ der links-, beziehungsweise rechtsseitige Grenzwert der `msDGL(l)`'ten Differentialgleichung am nStuetz(j)'ten diskreten Punkt gemeint. Genauer gilt

$$\mathtt{x_l(j)_+} = \mathtt{x(msDGL(k_l), msStuetz(k_j))_+} = \mathtt{unbe(k_l, k_j)}, \\ \mathtt{k_l} = 1, \ldots, \mathtt{nunbe}, \quad \mathtt{k_j} = 2, \ldots, \mathtt{nStuetz}$$

Nach Verlassen der Routine enthält die Routine alle Werte der (diskretisierten) Nebenbedingungen in der obigen Reihenfolge.

eps: Vektor der Länge m+1 vom Typ Real. Beim Aufruf der Routine muß er die Genauigkeitsschranken für die Nebenbedingungen enthalten. Das Programm bricht ab, wenn die Nebenbedingungen die folgenden Beschränkungen erfüllen:

$$\begin{array}{rlll} \mathtt{eps(i)} \geq & \mathtt{g(i)} & \geq -\mathtt{eps(i)} & \mathtt{i} = 1, \ldots, \mathtt{me} \\ & \mathtt{g(i)} & \geq -\mathtt{eps(i)} & \mathtt{i} = \mathtt{me}+1, \ldots, \mathtt{m} \end{array}$$

und sobald zusätzlich der projizierte Gradient der gegebenen Funktion kleiner ist, als $\mathtt{eps(m+1)}$, oder wenn das Programm die Abweichung der optimalen Lösung des Steuerproblems für kleiner als $\mathtt{eps(m+1)}$ hält.
In der Regel sollte gelten:

$$0 \leq \mathtt{eps(i)} \leq 10^{-3} \qquad \mathtt{i} = 1, \ldots, \mathtt{m}+1$$

Mit Hilfe dieser Vektoren besteht die Möglichkeit, die Nebenbedingungen ihrer Bedeutung nach zu gewichten.

f: Vektor der Länge n vom Typ Real. Beim Verlassen der Routine enthält der Vektor die erste Ableitung von $J[W]$ nach $u(t_i)$, $i = 1, \ldots, \mathtt{n}$.

uh: Vektor der Länge m vom Typ Real. Beim Verlassen der Routine enthält er eine Näherung der Lagrangemultiplikatoren.

uhelp: Vektor der Länge n vom Typ Real. In ihm werden die Variablen u und unbe zusammengefaßt und gemeinsam bearbeitet.

c, y, z: Vektoren der Länge maxnm vom Typ Real, die zu Hilfszwecken zur Verfügung stehen.

g1, g2: Vektoren der Länge m vom Typ Real, die zu Hilfszwecken zur Verfügung stehen.

p, oldf, beta, atg, z1, diag: Variablen der Länge n vom Typ Real, die zu Hilfszwecken zur Verfügung stehen.

`x`: Matrix der Dimension (`nDGL,ndiskret`) vom Typ Real. Nach Verlassen der Routine enthält die Matrix die errechnete Lösung des Differentialgleichungssystems an den diskreten Punkten `t(i)`, `i=1, ...,ndiskret`.

`u`: Matrix der Dimension (`nSteuer,ndiskret`) vom Typ Real. Beim Aufruf der Routine muß diese eine Startschätzung der Steuerungen an den diskreten Punkten `t(i)`, `i=1, ...,ndiskret` enthalten. Nach Verlassen der Routine enthält die Matrix die errechnete optimale Lösung der Steuerungen an den diskreten Punkten.

`unbe`: Matrix der Dimension (`nunbe,nStuetz`) vom Typ Real. Beim Aufruf der Routine muß diese eine Startschätzung der entsprechenden Trajektorien `msdgl(j)`, `j=1, ...,nunbe` an den Mehrzielknoten `t(msStuetz(i))`, `i=1, ...,nStuetz` enthalten. Nach Verlassen der Routine enthält die Matrix die errechnete Lösung des Differentialgleichungssystems an den Mehrzielknoten.

Wie schon erwähnt, unterliegt die Routine NUDOCCCS gewissen Einschränkungen bezüglich der Anzahl der Differentialgleichungen, der Nebenbedingungen und der Steuerung. Diese werden in der Subroutine PROBDEF als Konstanten definiert und können bei Bedarf den individuellen Problemanforderungen angepaßt werden. Sie haben folgende Bedeutung:

`dgl`: Parameter vom Typ Integer, der die maximale Anzahl zulässiger Differentialgleichungen angibt.
Standardsetzung: `dgl=20`

`neb`: Parameter vom Typ Integer, der die maximale Anzahl zulässiger Nebenbedingungen angibt, die sich auf das gesamte Optimierungsintervall beziehen.
Standardsetzung: `neb=20`

`dis`: Parameter vom Typ Integer, der die maximale Anzahl diskreter Punkte angibt.
Standardsetzung: `dis=150`

`ste`: Parameter vom Typ Integer, der die maximale Anzahl zulässiger Steuerungen angibt.

Standardsetzung: `ste=5`

Sollen sehr große Modelle betrachtet werden (zum Beispiel durch viele diskrete Punkte), so stellt sich das Problem, daß die Dimensionen der folgenden Felder so groß sind, daß sie nicht mehr in der Parameterliste von NUDOCCCS aufgenommen werden können.
Es wurde daher hier die Möglichkeit ausgenutzt, durch die benannten COMMON-Blöcke `WA`, `WB` und `WC` Felder zu definieren, auf welche von jeder Stelle aus zugegriffen werden kann. Es ist darauf zu achten, daß die vorgeschriebenen Mindestgrößen der COMMON-Blöcke nicht unterschritten werden. Vielmehr ist es sinnvoll diese Felder größer zu gestalten, um sie nicht jedesmal neu anpassen zu müssen. Eine Vergrößerung der Felder hat keinerlei Auswirkungen auf die Berechnung, wenn die Mindestgröße eingehalten wird.
Es ist lediglich darauf zu achten, daß der vom Rechner zur Verfügung stehende Speicherplatz nicht überschritten wird und die Blöcke in allen entsprechenden Unterroutinen (einschließlich der aufrufenden Routine) die jeweils gleiche Größe besitzen:

$$\mathtt{COMMON/WA/a(125,120)/WB/b1(120,120)/WC/q(120,120)}$$

Die Dimensionierung wurde hierbei frei gewählt. Die drei COMMON-Blöcke haben folgende Bedeutung:

`a`: Matrix mit der Mindestdimensionierung `(m,n)` vom Typ Real. Nach Verlassen der Routine enthält sie die ersten Ableitungen der `g(i)` nach `u(i)` und der zusätzlichen Optimierungsvariablen (Jacobi-Matrix).
Genauer gilt:

$$\mathtt{a(i,j)} = \frac{\partial \mathtt{g(i)}}{\partial \mathtt{uhelp(j)}}$$

Es ist nicht der Variablenname in diesem COMMON-Block, der von Bedeutung ist, sondern vielmehr der COMMON-Bereichs-Name `WA`, der exakt benutzt werden muß.

`b1`: Matrix mit der Mindestdimensionierung `(n,n)` vom Typ Real. Beim Aufruf der Routine NUDOCCCS muß `b1` eine positiv definite Matrix sein, die $[\nabla^2 \mathtt{J}[\mathtt{W}]]^{-1}$ approximiert. Nach Verlassen der Routine enthält sie eine Schätzung dieser inversen zweiten Ableitung der Lagrangefunktion. Sind beim Aufruf keine weiteren Information vorhanden, so ist eine in der Regel erfolgversprechende Strategie die, `b1` als Einheitsmatrix zu definieren.

`q`: Matrix mit der Mindestdimensionierung `(n,n)` vom Typ Real, die als Arbeitsspeicher benötigt wird.
Es ist nicht der Variablenname in diesem COMMON-Block, der von Bedeutung ist, sondern vielmehr der COMMON-Bereichs-Name `WA`, der exakt benutzt werden muß.

B.3 Vom Anwender bereitzustellende Unterprogramme

Neben einer aufrufenden Routine und den an NUDOCCCS zu übergebenden Parametern hat der Benutzer noch fünf Unterroutinen zur Festlegung des Zielfunktionals, der rechten Seite des Differentialgleichungssystems, der Anfangswerte, der punktuellen Nebenbedingungen (zum Beispiel Endbedingungen) und Nebenbedingungen in Gleichungs- und Ungleichungsform, die sich auf das ganze Optimierungsintervall beziehen, bereitzustellen. Dabei ist es wichtig, die Namen dieser Unterroutinen exakt zu übernehmen.

B.3.1 Die Subroutine `MINFKT`

Die Unterroutine zur Festlegung des Zielfunktionals hat die folgende Form:

```
SUBROUTINE  minfkt(x,u,t,min,nDGL,nSteuer,ndiskret)
```

Dabei bedeuten:

`nDGL`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Differentialgleichungen.

`nSteuer`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Steuerfunktionen.

`ndiskret`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die maximale Anzahl diskreter Punkte. Eine Benutzung von `t(ndiskret)` ist gleichbedeutend mit t_f.

`min`: Variable vom Typ Real. Beim Verlassen dieser Routine muß sie den Wert des Zielfunktionals enthalten.

`t`: Vektor der Länge `ndiskret` vom Typ Real. Beim Aufruf dieser Routine enthält er die diskreten Zeitintervallpunkte.

`x`: Matrix der Dimension (`nDGL,ndiskret`) vom Typ Real. Beim Aufruf dieser Routine enthält sie die aktuellen Werte der Trajektorie an den diskreten Zeitpunkten.

`u`: Matrix der Dimension (`nSteuer,ndiskret`) vom Typ Real. Beim Aufruf dieser Routine enthält sie die aktuellen Werte der Steuerungen an den diskreten Zeitpunkten.

B.3.2 Die Subroutine DGLSYS

Die Unterroutine zur Festlegung der rechten Seite des Differentialgleichungssystems hat die folgende Form:

```
SUBROUTINE  DGLSYS(x,u,t,dx,nDGL,nSteuer)
```

Dabei bedeuten:

`nDGL`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Differentialgleichungen.

`nSteuer`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Steuerfunktionen.

`t`: Variable vom Typ Real. Beim Aufruf dieser Routine im Zeitpunkt t_i enthält sie den diskreten Zeitpunkt.

`x`: Vektor der Länge `nDGL` vom Typ Real. Beim Aufruf dieser Routine im Zeitpunkt t_i enthält er den aktuellen Wert der Trajektorie zum diskreten Zeitpunkt.

`u`: Vektor der Länge `nSteuer` vom Typ Real. Beim Aufruf dieser Routine im Zeitpunkt t_i enthält er den aktuellen Wert der Steuerungen zum diskreten Zeitpunkt.

`dx`: Vektor der Länge `nDGL` vom Typ Real. Beim Verlassen dieser Routine enthält er die Werte der rechten Seite des Differentialgleichungssystems zum diskreten Zeitpunkt t_i.

B.3.3 Die Subroutine `ANFANGSW`

Die Unterroutine zur Festlegung der Anfangswerte des Differentialgleichungssystems hat die folgende Form:

```
SUBROUTINE  anfangsw(awx, unknown, nDGL, nunbe)
```

Dabei bedeuten:

`nDGL`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Differentialgleichungen und damit die Anzahl der benötigten Anfangswerte.

`nunbe`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der fehlenden Anfangswerte.

`unknown`: Vektor der Länge `nunbe` vom Typ Real. Beim Aufruf dieser Routine enthält er eine Schätzung für die fehlenden Anfangswerte.

`awx`: Vektor der Länge `nDGL` vom Typ Real. Beim Verlassen dieser Routine enthält er die Anfangswerte des Differentialgleichungssystems um die entsprechenden Schätzungen der fehlenden Anfangswerte erweitert.

B.3.4 Die Subroutine `RANDBED`

Die Unterroutine zur Festlegung der punktuellen Gleichungsnebenbedingungen hat die folgende Form:

```
SUBROUTINE  randbed(x, u, t, r, nDGL, nSteuer, ndiskret, nrand)
```

Dabei bedeuten:

`nDGL`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Differentialgleichungen.

`nSteuer`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Steuerfunktionen.

`ndiskret`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die maximale Anzahl diskreter Punkte. Eine Benutzung von `t(ndiskret)` ist gleichbedeutend mit t_f.

`nrand`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der punktuellen Gleichungsnebenbedingungen (ohne die Anfangswerte).

`t`: Vektor der Länge `ndiskret` vom Typ Real. Beim Aufruf dieser Routine enthält er die diskreten Zeitintervallpunkte.

`x`: Matrix der Dimension(`nDGL,ndiskret`) vom Typ Real. Beim Aufruf dieser Routine enthält sie die aktuellen Werte der Trajektorie an den diskreten Zeitpunkten.

`u`: Matrix der Dimension(`nSteuer,ndiskret`) vom Typ Real. Beim Aufruf dieser Routine enthält sie die aktuellen Werte der Steuerungen an den diskreten Zeitpunkten.

`r`: Vektor der Länge `nrand` vom Typ Real. Beim Verlassen der Routine enthält er die punktuellen Gleichungsnebenbedingungen.

B.3.5 Die Subroutine `NEBENBED`

Die Unterroutine zur Festlegung der Nebenbedingungen in Gleichungs- und Ungleichungsform, die sich auf das ganze Optimierungsintervall beziehen, hat die folgende Form:

```
SUBROUTINE  nebenbed(x,u,t,con,nDGL,nSteuer,nneben)
```

Dabei bedeuten:

`nDGL`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Differentialgleichungen.

`nSteuer`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Steuerfunktionen.

`nneben`: Variable vom Typ Integer. Beim Aufruf dieser Routine enthält sie die Anzahl der Nebenbedingungen in Gleichungs- und Ungleichungsform, die sich auf das ganze Optimierungsintervall beziehen.

`t`: Variable vom Typ Real. Beim Aufruf dieser Routine im Zeitpunkt t_i enthält sie den diskreten Zeitintervallpunkt.

`x`: Vektor der Länge `nDGL` vom Typ Real. Beim Aufruf dieser Routine im Zeitpunkt t_i enthält er den Wert der Trajektorie zum diskreten Zeitpunkt.

`u`: Vektor der Länge `nSteuer` vom Typ Real. Beim Aufruf dieser Routine im Zeitpunkt t_i enthält er den Wert der Steuerungen zum diskreten Zeitpunkt.

`con`: Vektor der Länge `nneben` vom Typ Real. Beim Verlassen dieser Routine enthält er die Nebenbedingungen in Gleichungs- und Ungleichungsform zum diskreten Zeitpunkt t_i. Es ist darauf zu achten, daß erst die Gleichungsnebenbedingungen und dann Ungleichungsnebenbedingungen in den Vektor `con` eingeordnet werden.

B.4 Anmerkungen

Es ist darauf zu achten, daß die Dimensionierungen der Vektoren und Matrizen exakt durchgeführt werden und daß nicht auf Vektor- oder Matrizenfelder zugegriffen wird, die nicht existieren. Denn dies kann zu fehlerhaften Berechnungen der Lösungen führen. Lediglich die Dimensionierung der benannten COMMON-Blöcke ist frei wählbar, jedoch darf die vorgeschriebene Mindestgröße (s.o.) nicht unterschritten werden.

Anhang C

Beschreibung der Subroutine KOLLOKAT

C.1 Problemstellung

Die Fortran-Routine KOLLOKAT unterscheidet sich nicht in der grundsätzlichen Bedeutung von der Routine NUDOCCCS, daher soll an dieser Stelle auf eine ausführlichere Beschreibung verzichtet werden.

C.2 Die KOLLOKAT-Parameterliste

Die Subroutine KOLLOKAT muß innerhalb eines FORTRAN-Programms folgendermaßen aufgerufen werden:

```
      CALL  KOLLOKAT( nDGL, nSteuer, ndiskret, nneben, nuglnb,
     c                nrand, iprint, imax, istop, scale, r1, del,
     c                n, m, me, maxnm, x, u, f, ff, g, uh, iv, ivset,
     c                eps, t, h, uhelp, y, z, c, p, oldf, beta, atg, z1,
     c                g1, g2, diag)
```

Die einzelnen Parameter haben folgende Bedeutung:

`nrand`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl der punktuellen Nebenbedingungen in Gleichungsform enthalten, wie sie zum Beispiel durch die evtl. Anfangs- und Endbedingungen des optimalen Steuerprozesses beschrieben werden. Anfangswerte werden (im Gegensatz zur Routine `NUDOCCCS`) zu diesen Nebenbedingungen hinzugezählt.

`n`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl aller (diskretisierten) Optimierungsvariablen

enthalten.
Hier gilt:

$$\texttt{n} = \texttt{ndiskret} * (\texttt{nDGL} + \texttt{nSteuer})$$

`m`: Eingangsparameter vom Typ Integer. Beim Aufruf der Routine muß er die Anzahl aller (diskretisierten) Nebenbedingungen enthalten.
Hier gilt:

$$\texttt{m} = \texttt{ndiskret} * (\texttt{nDGL} + \texttt{nneben}) + \texttt{nrand} - 1$$

`del`: Eingangsparameter vom Typ Real. Beim Aufruf der Routine muß er die Schrittweite enthalten, mit der die numerische Differentiation durch den zentralen Differenzenquotienten durchgeführt werden soll.

`nDGL`, `nSteuer`, `ndiskret`, `nneben`, `nuglnb`, `iprint`, `imax`, `istop`, `scale`, `r1`, `x`, `u`, `f`, `ff`, `g`, `uh`, `iv`, `ivset`, `eps`, `t`, `h`, `uhelp`, `y`, `z`, `c`, `p`, `oldf`, `beta`, `atg`, `z1`, `g1`, `g2`,:
`diag` Diese Variablen besitzen die gleiche Bedeutung, wie in der Routine `NUDOCCCS`.

Auch die Routine KOLLOKAT unterliegt gewissen Einschränkungen bezüglich der Anzahl der Differentialgleichungen, der Nebenbedingungen und der Steuerung. Diese werden ebenso in der SUBROUTINE PROBDEF als Konstanten definiert und können bei Bedarf den individuellen Problemanforderungen angepaßt werden. Sie haben eine zur Routine NUDOCCCS identische Bedeutung.
Es ist zu beachten, daß für die Routine KOLLOKAT der Speicheraufwand sehr viel größer ist, als bei der Routine NUDOCCCS, dies hat insbesondere Auswirkungen auf die benannten COMMON-Blöcke, die in ihrer Bedeutung denen aus NUDOCCCS entsprechen.
Es ist auch hier darauf zu achten, daß die vorgeschriebenen Mindestgrößen der COMMON-Blöcke nicht unterschritten werden. Vielmehr ist es sinnvoll diese Felder größer zu gestalten, um sie nicht jedesmal neu anpassen zu müssen. Eine Vergrößerung der Felder hat keinerlei Auswirkungen auf die Berechnung, wenn die Mindestgröße eingehalten wird.
Es ist lediglich darauf zu achten, daß der vom Rechner zur Verfügung stehende Speicherplatz nicht überschritten wird und die Blöcke in allen entsprechenden Unterroutinen (einschließlich der aufrufenden Routine) die jeweils gleiche Größe besitzen.

C.3 Vom Anwender bereitzustellende Unterprogramme

Im wesentlichen entsprechen die Unterroutinen von KOLLOKAT denen von NUDOCCCS, lediglich die Unterroutine `anfangsw` entfällt. Die entsprechenden Anfangswerte können in der Unterroutine `randbed` als Gleichungsnebenbedingungen definiert werden.

C.4 Anmerkungen

Es ist darauf zu achten, daß die Dimensionierungen der Vektoren und Matrizen exakt durchgeführt werden und daß nicht auf Vektor- oder Matrizenfelder zugegriffen wird, die nicht existieren. Denn dies kann zu fehlerhaften Berechnungen der Lösungen führen. Lediglich die Dimensionierung der benannten COMMON-Blöcke ist frei wählbar, jedoch darf die vorgeschriebene Mindestgröße (s.o.) nicht unterschritten werden.

Anhang D

Beschreibung der Subroutine DIROPTIM

D.1 Problemstellung

Die Fortran-Routine DIROPTIM unterscheidet sich nicht in der grundsätzlichen Bedeutung von der Routine KOLLOKAT, daher soll an dieser Stelle auf eine ausführlichere Beschreibung verzichtet werden.

D.2 Die DIROPTIM-Parameterliste

Die Subroutine DIROPTIM muß innerhalb eines FORTRAN-Programms folgendermaßen aufgerufen werden:

```
      CALL  DIROPTIM( nDGL,nSteuer,ndiskret,nneben,nuglnb,
   c                  nrand,iprint,imax,istop,scale,r1,del,
   c                  n,m,me,maxnm,x,u,f,ff,g,uh,iv,ivset,
   c                  eps,t,h,uhelp,y,z,c,p,oldf,beta,atg,z1,
   c                  g1,g2,diag)
```

Die einzelnen Parameter haben die gleiche Bedeutung, wie in der Routine KOLLOKAT.
Auch alle Einschränkungen entsprechen denen der Routine Kollokat.

D.3 Vom Anwender bereitzustellende Unterprogramme

Alle Unterroutinen von DIROPTIM entsprechen denen von KOLLOKAT.

D.4 Anmerkungen

Es ist auch hier darauf zu achten, daß die Dimensionierungen der Vektoren und Matrizen exakt durchgeführt werden und daß nicht auf Vektor- oder Matrizenfelder zugegriffen wird, die nicht existieren. Denn dies kann zu fehlerhaften Berechnungen der Lösungen führen. Lediglich die Dimensionierung der benannten COMMON-Blöcke ist frei wählbar, jedoch darf die vorgeschriebene Mindestgröße (s.o.) nicht unterschritten werden.

Anhang E

Beschreibung der Subroutine DIRMESCH

E.1 Problemstellung

Die Fortran-Routine DIRMESCH unterscheidet sich nicht in der grundsätzlichen Bedeutung von der Routine KOLLOKAT, daher soll an dieser Stelle auf eine ausführlichere Beschreibung verzichtet werden.

E.2 Die DIRMESCH-Parameterliste

Die Subroutine DIRMESCH muß innerhalb eines FORTRAN-Programms folgendermaßen aufgerufen werden:

```
      CALL  DIRMESCH( nDGL,nSteuer,ndiskret,nneben,nuglnb,
  c                   nrand,iprint,imax,istop,scale,r1,del,
  c                   n,m,me,maxnm,x,u,f,ff,g,uh,iv,ivset,
  c                   eps,tol1,tol2,tol3,t,h,uhelp,y,z,c,p,
  c                   oldf,beta,atg,z1,g1,g2,diag)
```

Die einzelnen Parameter haben die gleiche Bedeutung, wie in der Routine KOLLOKAT. Ergänzend gelten jedoch die folgenden Parameter:

`tol1`: Eingangsparameter vom Typ Real. Beim Aufruf der Routine bestimmt er die obere Schranke des lokalen Diskretisierungsfehlers bei der numerischen Integration.

`tol2`: Eingangsparameter vom Typ Real. Beim Aufruf der Routine muß er die maximale Schrittweite der numerischen Integration enthalten.

`tol3`: Eingangsparameter vom Typ Real. Beim Aufruf der Routine muß er die Anfangsschrittweite der numerischen Integration enthalten. Es ist aufgrund der zu erwartenden Rechenzeit zu empfehlen, `tol3` relativ groß zu wählen.

Alle Einschränkungen entsprechen denen der Routine Kollokat.

E.3 Vom Anwender bereitzustellende Unterprogramme

Alle Unterroutinen von DIRMESCH entsprechen denen von KOLLOKAT.

E.4 Anmerkungen

Es ist auch hier darauf zu achten, daß die Dimensionierungen der Vektoren und Matrizen exakt durchgeführt werden und daß nicht auf Vektor- oder Matrizenfelder zugegriffen wird, die nicht existieren. Denn dies kann zu fehlerhaften Berechnungen der Lösungen führen. Lediglich die Dimensionierung der benannten COMMON-Blöcke ist frei wählbar, jedoch darf die vorgeschriebene Mindestgröße (s.o.) nicht unterschritten werden.

Anhang F

Beispiele einiger FORTRAN-Programme

In diesem Kapitel sollen einige Listing der vom Anwender bereitzustellenden Unterroutinen angegeben werden. Da sich die Routinen NUDOCCCS, KOLLOKAT, DIROPTIM und DIRMESCH sehr ähneln, werden nur die entsprechenden Routinen für BDSCO und NUDOCCCS ausgegeben. Nach jedem Listing werden weiterhin die entsprechenden Ausgabedaten der Routinen aufgelistet.

F.1 Anwendungsbeispiel 1

Die unbeschränkte Verladebrücke

Die entsprechenden Routinen zu BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(16,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US

C DATENDEFINITION-----------------------------------------
      NDISKRET=30
      ISCHALT=1
      N=5
      M=6
      MS=3
      TOL=1E-12
      MAXITER=300
      ITMAX=350
      KP=0
      KS=1

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE
```

```
C STARTSCHAETZUNG DER TRAJEKTORIEN------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.1D0
        Y(5,I) = 4.3D0
200   CONTINUE

C EXAKTE RANDWERTE----------------------------------------
      Y(1,1)=0.0
      Y(1,M)=1.0
      Y(2,1)=0.0
      Y(2,M)=0.0
      Y(3,1)=0.0
      Y(3,M)=0.0
      Y(4,1)=0.0
      Y(4,M)=0.0

C STARTSCHAETZUNG DER SCHALTPUNKTE------------------------
      XS(1)=0.18
      XS(2)=0.5
      XS(3)=0.82

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,1E-7,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'OPTIMALE ENDZEIT : ',Y(5,M)
      END



      SUBROUTINE F(X,Y,DY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US

C DEFINITION DER STEUERUNG--------------------------------
        IF (US(J,1).EQ.-1) U= 1.0D0
        IF ((US(J,1).EQ.1).AND.(US(J,2).EQ.-1)) U=-1.0D0
        IF ((US(J,2).EQ.1).AND.(US(J,3).EQ.-1)) U=1.0D0
        IF (US(J,3).EQ.1) U=-1.0D0

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS------------
        DY(1) = Y(2)
        DY(2) = U
        DY(3) = Y(4)
        DY(4) = -Y(3)+U
        DY(5) = 0.0D0
        DO 211 I=1,4
          DY(I) = DY(I)*Y(5)
211     CONTINUE

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US

C ANFANGSBEDINGUNGEN--------------------------------------
        W(1)=YA(1)
```

```
      W(2)=YA(2)
      W(3)=YA(3)
      W(4)=YA(4)

      NYA(1)=1
      NYA(2)=2
      NYA(3)=3
      NYA(4)=4

C ENDBEDINGUNGEN-----------------------------------------
      W(5)=YB(1)-1.0D0
      W(6)=YB(2)
      W(7)=YB(3)
      W(8)=YB(4)

C SCHALTBEDINGUNGEN--------------------------------------
C ES WERDEN KEINE SCHALTBEDINGUNGEN BENOETIGT

      RETURN
      END
```

Mit der Ausgabe:

```
   N= 5      M= 6      MS= 3
   PRESCRIBED RELATIVE PRECISION  0.10E-09
   MAXIMUM PERMITTED NUMBER OF ITERATIONS350

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)   NORM(M)
 0  0.58E+01  0.81E+03  0.81E+03  0.68E+03          0    48  0.18E+03  0.43E+01
    0.57E+01  0.79E+03  0.79E+03  0.67E+03  0.010
 1  0.57E+01  0.79E+03  0.79E+03  0.67E+03          1    48  0.22E+03  0.54E+01
    0.49E-03  0.61E-01  0.63E-01  0.11E+01  1.000
 2  0.49E-03  0.61E-01  0.63E-01  0.12E+01          2    48  0.22E+03  0.54E+01
    0.30E-02  0.40E+00  0.41E+00  0.78E+01  1.000
    0.32E-05  0.22E-03  0.22E-03  0.22E-02  0.275
 3  0.32E-05  0.14E-04  0.13E-04  0.23E-04          0    48  0.40E+02  0.54E+01
    0.62E-12  0.20E-11  0.19E-11  0.27E-10  1.000
 4  0.62E-12  0.20E-11  0.19E-11  0.26E-10          1    48  0.39E+02  0.54E+01
    0.95E-18  0.62E-17  0.61E-17  0.87E-16  1.000
 5  0.95E-18  0.63E-17  0.61E-17  0.87E-16          2    48  0.39E+02  0.54E+01
    0.38E-23  0.87E-22  0.86E-22  0.12E-21  1.000
 6  0.38E-23  0.87E-22  0.86E-22  0.12E-21          3    48  0.39E+02  0.54E+01
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER  7    ITERATION STEPS



   SWITCHING POINTS
   0.1873830E+00 0.5000000E+00 0.8126170E+00


OPTIMALE ENDZEIT :        4.28191209570454046
```

Die entsprechenden Routinen der Routine NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 5,
```

```
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 2,
     G   NUGLNB   = 2,
     H   NRAND    = 4,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.0D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 5.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX--------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
      EPS(M+1)=EP

      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END
```

```
      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = X(5,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = X(2)*X(5)
        DX(2) = U(1)*X(5)
        DX(3) = X(4)*X(5)
        DX(4) = (U(1)-X(3))*X(5)
        DX(5) = 0.0D0

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = 0.0D0
        AWX(3) = 0.0D0
        AWX(4) = 0.0D0
        AWX(5) = UNKNOWN(1)

      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)

        R(1) = X(1,NDISKRET)-1.0D0
        R(2) = X(2,NDISKRET)
        R(3) = X(3,NDISKRET)
        R(4) = X(4,NDISKRET)

       RETURN
       END



      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)
```

```
        CON(1) = -U(1)+1.0D0
        CON(2) =  U(1)+1.0D0

      RETURN
      END
```

Mit der Ausgabe:

1	3.99999999986022203
2	3.10528487987008761
3	2.68413818902764900
4	2.95960281784165113
5	3.46211673520567986
6	3.94337126540225746
7	4.10494615505443772
8	4.14237781526109661
9	4.20328437838964497
10	4.22620361167891989
11	4.23961984034954575
12	4.33246562059744988
13	4.37960080914033156
14	4.39513065532199221
15	4.39356604184506505
16	4.37884470685249028
17	4.36891876044983540
18	4.36589057187135054
19	4.36439678782131946
20	4.36365323976051123
21	4.36324302172748180
22	4.36137278547633311
23	4.36047239869747827
24	4.36002226942279414
25	4.35972432570630275
26	4.34115573508939967
27	4.33231891522073820
28	4.32752407075348877
29	4.32510191968153901
30	4.32388904693435272
31	4.32328218153073607
32	4.32297939010274757
33	4.32284172456910110
34	4.32220382578124696
35	4.32193005739795932
36	4.32166191760505836
37	4.32153245758694382
38	4.32146788326324760
39	4.32143560456497866
40	4.32207585265696892
41	4.32229705475885950
42	4.32238962866272658
43	4.32243167641890835
44	4.32245164138411120
45	4.32263002038256960
46	4.32271621034882081
47	4.33155902686179317
48	4.33566204357007923
49	4.33625409212825730
50	4.33624154848104570
51	4.33581037732439789
52	4.33558499215271098
53	4.33547161057172215
54	4.33541487097044875
55	4.33538649899470929
56	4.33509602502869829

```
  57        4.33495424823140407
  58        4.33483876199461093
  59        4.33478144880222604
  60        4.33475279693368609
  61        4.33473847165928117
  62        4.33473130924445815
  63        4.33472772812302320
  64        4.33472593759901681
  65        4.33352341976589539
  66        4.33107296503864259
  67        4.32668445027829751
  68        4.32450108656233567
  69        4.32341218112515424
  70        4.32286841984431192
  71        4.32259671189783035
  72        4.32246090116314896
  73        4.32239300665695669
  74        4.32235906214696719
  75        4.32234209059205554
  76        4.32233360499676422
  77        4.32232936224824282
  78        4.32232724088807263
  79        4.32232618021240444
  80        4.32232564987612111
  81        4.32232538470859318
  82        4.32232668442063428
  83        4.32232731166403195
  84        4.32349134143182834
  85        4.32422554603642340
  86        4.32468234714440847
  87        4.32496695707600409
  88        4.32514448834256982
  89        4.32525530842014305
  90        4.32532451743925694
  91        4.32536775218118397
  92        4.32539476574911941
  93        4.32541164605614892
  94        4.32542219501244052
  95        4.32542878762993510
  96        4.32543290782922352
  97        4.32543548288132396
  98        4.32543709226083806
  99        4.32543809811198887
 100        4.32543872676475250
 101        4.32543911967107686
 102        4.32543936523690320
 103        4.32543951871529231
 104        4.32543961463919224
 105        4.32543967459158996
 106        4.32543971206182665
 107        4.32543973548071925
 108        4.32543975011752391
 109        4.32543975926552626
 LAGRANGE MULTIPLIKATOREN
  0.5122815D+00  -0.1107933D+01   0.3982549D+00   0.5924102D+00
  0.5909055D-01   0.8011751D-01   0.3310301D-01   0.1377618D-01
  0.2383242D-01   0.2671893D-01   0.1950665D-01   0.1950641D-01
  0.2671869D-01   0.2383228D-01   0.1377613D-01   0.3310359D-01
  0.8011912D-01   0.5908006D-01
LOESUNG
  0.1000000D+01   0.1000000D+01   0.1000000D+01   0.1691398D-02
 -0.1000000D+01  -0.1000000D+01  -0.1000000D+01  -0.1000000D+01
 -0.1281016D-03   0.1000000D+01   0.1000000D+01   0.1000000D+01
  0.1000000D+01  -0.1563281D-02  -0.1000000D+01  -0.1000000D+01
```

```
      -0.1000000D+01   0.0000000D+00   0.0000000D+00   0.4325440D+01
      FUNKTION
        0.4325440D+01
      NEBENBEDINGUNGEN
      -0.1929503D-08   0.4170452D-08  -0.1498045D-08  -0.2229648D-08
      -0.2226763D-09  -0.3015206D-09  -0.1244513D-09   0.9983086D+00
       0.2000000D+01   0.2000000D+01   0.2000000D+01   0.2000000D+01
       0.1000128D+01  -0.5169887D-10  -0.8970136D-10  -0.1004918D-09
      -0.7332290D-10   0.1001563D+01   0.2000000D+01   0.2000000D+01
       0.2000000D+01   0.2000000D+01   0.2000000D+01   0.2000000D+01
       0.1001691D+01  -0.7341794D-10  -0.1006293D-09  -0.8986967D-10
      -0.5169665D-10   0.9998719D+00   0.2000000D+01   0.2000000D+01
       0.2000000D+01   0.2000000D+01   0.9984367D+00  -0.1246179D-09
      -0.3019944D-09  -0.2223988D-09
       110 FUNKTIONS AUFRUFE
       110 GRADIENTEN AUFRUFE
      KONVERGENT
min =       4.32543975926552626
```

Die durch $x_2(t) \leq 0.4$ beschränkte Verladebrücke

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(16,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US
      COMMON /SCHR/ALPHA

C DATEN DEFINITION-----------------------------------------
      N=9
      M=6
      MS=5
      TOL=1E-12
      MAXITER=300
      ITMAX=350
      KP=0
      KS=1
      NDISKRET=30
      ISCHALT=1

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIE--------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.1D0
        Y(9,I) = 6.0D0
200   CONTINUE

C EXAKTE RANDWERTE-----------------------------------------
      Y(1,1)=0.0
      Y(1,M)=1.0
      Y(2,1)=0.0
      Y(2,M)=0.0
      Y(3,1)=0.0
      Y(3,M)=0.0
      Y(4,1)=0.0
      Y(4,M)=0.0
```

```
C STARTSCHAETZUNG DER SCHALTPUNKTE------------------------
      XS(1)=0.1
      XS(2)=0.4
      XS(3)=0.5
      XS(4)=0.6
      XS(5)=0.9

      ALPHA = 0.4d0

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,1E-7,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'OPTIMALE ENDZEIT: ',Y(9,M)
      END




      SUBROUTINE F(X,Y,DY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US
      COMMON /SCHR/ALPHA

C DEFINITION DER STEUERUNG---------------------------------
        IF (US(J,1).EQ.-1) U= 1.0D0
        IF ((US(J,1).EQ.1).AND.(US(J,2).EQ.-1)) U=0.0D0
        IF ((US(J,2).EQ.1).AND.(US(J,3).EQ.-1)) U=-1.0D0
        IF ((US(J,3).EQ.1).AND.(US(J,4).EQ.-1)) U=1.0D0
        IF ((US(J,4).EQ.1).AND.(US(J,5).EQ.-1)) U=0.0D0
        IF (US(J,5).EQ.1) U=-1

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS-------------
        DY(1) = Y(2)
        DY(2) = U
        DY(3) = Y(4)
        DY(4) = -Y(3)+U
        DY(5) = 0.0D0
        DY(6) = -U*Y(5)-(DABS(U)-1.0D0)*Y(7)
        DY(7) = Y(8)
        DY(8) = -Y(7)
        DY(9) = 0.0D0
        DO 211 I=1,8
          DY(I)=DY(I)*Y(9)
211     CONTINUE

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US
      COMMON /SCHR/ALPHA

C  ANFANGSBEDINGUNGEN--------------------------------------
        W(1)=YA(1)
        W(2)=YA(2)
        W(3)=YA(3)
        W(4)=YA(4)
```

```
        NYA(1)=1
        NYA(2)=2
        NYA(3)=3
        NYA(4)=4

C  ENDBEDINGUNGEN----------------------------------------
        W(5)=YB(1)-1.0D0
        W(6)=YB(2)
        W(7)=YB(3)
        W(8)=YB(4)

C  SCHALTBEDINGUNGEN-------------------------------------
        W(9)    = ZZ(6)+ZZ(8)
        W(10)   = ZZ(2)-ALPHA
        W(11)   = ZZ(6)+ZZ(8)
        W(12)   = ZZ(2)-ALPHA
        W(13)   = ZZ(6)+ZZ(8)
        W(14)   = -YB(6)-YB(8)+ALPHA*(YB(5)+YB(7))

        NSK(9) = 3
        NSK(10)= 1
        NSK(11)= 2
        NSK(12)= 4
        NSK(13)= 5

      RETURN
      END
```

Mit der Ausgabe:

```
   N= 9      M= 6      MS= 5
   PRESCRIBED RELATIVE PRECISION  0.10E-09
   MAXIMUM PERMITTED NUMBER OF ITERATIONS350

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)   NORM(M)
 0  0.28E+01  0.15E+03  0.17E+03  0.84E+03          0   84  0.91E+03  0.60E+01
    0.28E+01  0.15E+03  0.17E+03  0.83E+03  0.010
 1  0.28E+01  0.13E+03  0.14E+03  0.77E+03          1   84  0.10E+04  0.68E+01
    0.22E+01  0.10E+03  0.11E+03  0.62E+03  0.097
 2  0.22E+01  0.10E+03  0.10E+03  0.55E+03          2   84  0.86E+03  0.66E+01
    0.80E+00  0.40E+02  0.46E+02  0.20E+03  0.472
 3  0.80E+00  0.46E+02  0.62E+02  0.23E+03          3   84  0.54E+03  0.59E+01
    0.69E+00  0.67E+02  0.36E+03  0.24E+04  1.000
    0.49E+00  0.30E+02  0.31E+02  0.59E+02  0.240
 4  0.49E+00  0.20E+02  0.21E+02  0.80E+02          0   84  0.18E+03  0.55E+01
    0.38E+00  0.28E+02  0.29E+02  0.38E+02  0.329
 5  0.38E+00  0.25E+02  0.25E+02  0.48E+02          0   84  0.20E+03  0.58E+01
    0.55E-02  0.31E+00  0.32E+00  0.12E+01  1.000
 6  0.55E-02  0.73E-01  0.77E-01  0.11E+01          1   84  0.20E+03  0.58E+01
    0.10E-02  0.98E-02  0.10E-01  0.45E-01  1.000
 7  0.10E-02  0.21E+00  0.97E+00  0.42E+01          0   84  0.24E+05  0.61E+01
    0.40E-07  0.13E-04  0.15E-04  0.69E-04  1.000
 8  0.40E-07  0.84E-01  0.84E-01  0.13E+01          1   84  0.11E+07  0.62E+01
    0.37E-08  0.35E-05  0.33E-04  0.18E+00  1.000
 9  0.37E-08  0.23E-04  0.20E+01  0.12E+02          0   84  0.91E+07  0.61E+01
    0.63E-18  0.19E-10  0.73E-10  0.55E-09  1.000
10  0.63E-18  0.19E-10  0.73E-10  0.55E-09          1   84  0.91E+07  0.61E+01
    0.10E-26  0.52E-21  0.12E-17  0.85E-17  1.000
11  0.10E-26  0.52E-21  0.12E-17  0.85E-17          2   84  0.91E+07  0.61E+01
    0.17E-29  0.38E-28  0.97E-26  0.79E-25  1.000
```

```
12  0.17E-29  0.38E-28  0.97E-26  0.79E-25         3   84  0.91E+07  0.61E+01
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER 13    ITERATION STEPS


   SWITCHING POINTS
   0.8551394E-01 0.3197298E+00 0.5000000E+00 0.6802702E+00 0.9144861E+00


OPTIMALE ENDZEIT:       4.67759969812937282
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 5,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 3,
     G   NUGLNB   = 3,
     H   NRAND    = 4,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.0D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 5.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX-------------------------
```

```
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
      EPS(M+1)=EP

      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = X(5,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = X(2)*X(5)
        DX(2) = U(1)*X(5)
        DX(3) = X(4)*X(5)
        DX(4) = (U(1)-X(3))*X(5)
        DX(5) = 0.0D0

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = 0.0D0
        AWX(3) = 0.0D0
        AWX(4) = 0.0D0
        AWX(5) = UNKNOWN(1)

      RETURN
      END
```

```
      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)

        R(1) = X(1,NDISKRET)-1.0D0
        R(2) = X(2,NDISKRET)
        R(3) = X(3,NDISKRET)
        R(4) = X(4,NDISKRET)

       RETURN
       END



      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

        CON(1) = -U(1)+1.0D0
        CON(2) =  U(1)+1.0D0
        CON(3) = -X(2)+0.4D0

      RETURN
      END
```

Mit der Ausgabe:

```
     1        3.99999999986022203
     2        3.10528487987008761
     3        2.68413818902764900
     4        2.95960281784165113
     5        3.46211673520567986
     6        3.94912767062018144
     7        4.34980375781566764
     8        4.40372617747588535
     9        4.49367920108338392
    10        4.55007697754608564
    11        4.57071693879304775
    12        4.58992671912099048
    13        4.65857719926206926
    14        4.69798892693372849
    15        4.71652526690467777
    16        4.72097476316173559
    17        4.72144447025829450
    18        4.72495984003336567
    19        4.72629573038978723
    20        4.73457518234720087
    21        4.73937681615665518
    22        4.74165225387216172
    23        4.74231815821597280
    24        4.74196994616124723
    25        4.73598818690899304
    26        4.73301623917117631
    27        4.73152632543444351
    28        4.73078509452901930
    29        4.73041592556313528
    30        4.73022973386683088
    31        4.73013658782107527
    32        4.73008935214540305
    33        4.73006572235597034
    34        4.73005385495570785
    35        4.73004791595888197
```

36	4.72978088946499575
37	4.72964798054375368
38	4.72957078198968084
39	4.72953159021439973
40	4.72951198987080179
41	4.72950247743250696
42	4.72949772171415539
43	4.72949534372974245
44	4.72702648845163331
45	4.72208611981672544
46	4.72083688004305735
47	4.72021328846041044
48	4.71990175956781854
49	4.71974606178098988
50	4.71966822955546572
51	4.71962931209187975
52	4.71960985576427561
53	4.71960012737115697
54	4.71959526488298575
55	4.71959283337368962
56	4.71959164950752275
57	4.71959105758079822
58	4.71959076161756119
59	4.71959061363594756
60	4.71959053964512787
61	4.71959050264970692
62	4.71954659907472252
63	4.71952447240165807
64	4.71951276695165856
65	4.71950690385733385
66	4.71950397201854788
67	4.71950250550627093
68	4.71950177224231826
69	4.71950141540518509
70	4.71950123699076496
71	4.71950114778350649
72	4.71950110317984794
73	4.71950108087800491
74	4.71628330038573296
75	4.71284437121504940
76	4.71209785649743562
77	4.71172498218288460
78	4.71153864096604114
79	4.71144549327502560
80	4.71139892535611438
81	4.71137564271573872
82	4.71136400182373638
83	4.71135818140172002
84	4.71135527180691227
85	4.71135381701534417
86	4.71135308962101806
87	4.71135272592421561
88	4.71135254407590320
89	4.71135245315176832
90	4.71135240768970664
91	4.71135238495867714
92	4.71135237359316239
93	4.71135236791040501
94	4.71135236506902633
95	4.71135236364833698
96	4.71135236293799231
97	4.71135236258347501
98	4.71135128786176249
99	4.71135075044614116

```
   100        4.71135046069244279
   101        4.71135031570866669
   102        4.71135024321596862
   103        4.71135020698551976
   104        4.71135018887029489
   105        4.71135017984269844
   106        4.71135017532888334
   107        4.71135017316219606
   108        4.71135017207889106
   109        4.71135017149792290
   110        4.71135017120743793
   111        4.71135017106201293
   112        4.71136616536195074
   113        4.71137647285733241
   114        4.71138296420422709
   115        4.71138704095022387
   116        4.71138959677569158
   117        4.71139119730970979
   118        4.71139219890057603
   119        4.71139282539768001
   120        4.71139321715948967
   121        4.71139346209106780
   122        4.71139361520548317
   123        4.71139371091486403
   124        4.71139377073837551
   125        4.71139380813013009
   126        4.71139383150080082
   127        4.71139384610779910
   128        4.71139385523730514
   129        4.71139386094329904
  LAGRANGE MULTIPLIKATOREN
   0.7760506D+00  -0.1156481D+01   0.6315116D+00   0.6322471D+00
   0.5753186D-01   0.4281922D-01   0.8512135D-02   0.1049980D-01
   0.1049970D-01   0.8512095D-02   0.4282114D-01   0.5751505D-01
   0.3412826D+00   0.1410546D+00   0.1893279D+00   0.1893314D+00
   0.1410595D+00   0.3412928D+00
 LOESUNG
   0.1000000D+01   0.1000000D+01  -0.2831819D+00   0.2831819D+00
  -0.2831819D+00  -0.2167722D+00  -0.1000000D+01  -0.1000000D+01
  -0.4149793D-03   0.1000000D+01   0.1000000D+01   0.2171872D+00
   0.2831819D+00  -0.2831819D+00   0.2831819D+00  -0.1000000D+01
  -0.1000000D+01   0.0000000D+00   0.0000000D+00   0.4711394D+01
 FUNKTION
   0.4711394D+01
 NEBENBEDINGUNGEN
  -0.1195190D-08   0.1771504D-08  -0.9778550D-09  -0.9786418D-09
  -0.8818568D-10  -0.6570566D-10   0.1283182D+01   0.7168181D+00
   0.1283182D+01   0.1216772D+01   0.2000000D+01   0.2000000D+01
   0.1000415D+01  -0.1288081D-10  -0.1604850D-10   0.7828128D+00
   0.7168181D+00   0.1283182D+01   0.7168181D+00   0.2000000D+01
   0.2000000D+01   0.2000000D+01   0.2000000D+01   0.7168181D+00
   0.1283182D+01   0.7168181D+00   0.7832278D+00  -0.1649902D-10
  -0.1288725D-10   0.9995850D+00   0.2000000D+01   0.2000000D+01
   0.1217187D+01   0.1283182D+01   0.7168181D+00   0.1283182D+01
  -0.6557443D-10  -0.8809420D-10   0.4000000D+00   0.1055379D+00
  -0.5211376D-09  -0.2159704D-09  -0.2883432D-09   0.7360877D-01
   0.2527554D+00   0.5472175D+00   0.6945097D+00   0.5473397D+00
   0.2528776D+00   0.7366987D-01  -0.2892974D-09  -0.2150727D-09
  -0.5220991D-09   0.1055379D+00   0.4000000D+00
  134 FUNKTIONS AUFRUFE
  130 GRADIENTEN AUFRUFE
 KONVERGENT
min =        4.71139386094329904
```

F.2 Anwendungsbeispiel 2

Optimale Temperaturverteilung in einem chemischen Reaktor

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION X(15),Y(15,15),XS(4),X1(15)
      INTEGER US(15,4)
      EXTERNAL F,R,RKF7
      COMMON /CI/J,L,LS,US

CDATENDEFINITION------------------------------------------
      N     =  4
      M     =  5
      MS    =  1
      KP    =  0
      KS    =  1
      ITMAX = 20
      TOL   = 1.0D-12
      NDISKRET=30
      ISCHALT=1

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIE--------------------------
      DO 200 I=0,M-1
        X(I+1)  = 0.0D0+DBLE(I)/DBLE(M-1)
        Y(1,I+1)= 1.0D0-DBLE(I)/DBLE(M-1)*0.4D0
        Y(2,I+1)= 0.0D0+DBLE(I)/DBLE(M-1)*0.3D0
        Y(3,I+1)=-0.3D0+DBLE(I)/DBLE(M-1)*0.3D0
        Y(4,I+1)=-0.4D0-DBLE(I)/DBLE(M-1)*0.6D0
200   CONTINUE

C STARTSCHAETZUNG DER SCHALTPUNKTE-------------------------
        XS(1)  =   0.1D0

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,TOL,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*)'MIN =',Y(2,M)
      STOP
      END




      SUBROUTINE F(X,Y,DY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION X,Y(9),DY(9)
      INTEGER US(15,4)
      COMMON /CI/J,L,LS,US

C DEFINITION DER STEUERUNG---------------------------------
        IF (US(J,1).EQ.-1) U=1.0D0
        IF (US(J,1).EQ.1) U=(Y(3)-Y(4))*Y(1)*0.5D0
     F                    /((Y(3)-3.0D0*Y(4))*Y(2))

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS-------------
        DY(1) = -U*Y(1)+U*U*Y(2)
        DY(2) = U*Y(1)-3.0D0*U*U*Y(2)
```

```
        DY(3) = (Y(3)-Y(4))*U
        DY(4) = (3.0D0*Y(4)-Y(3))*U*U

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      DOUBLE PRECISION YA(5),YB(5),ZZ(5),W(5),Z0,RR
      INTEGER J,L,LS,NYA(5),NSK(5),US(15,4)
      COMMON /CI/J,L,LS,US

C ANFANGSBEDINGUNGEN--------------------------------------
        W(1)=YA(1)-1.0D0
        W(2)=YA(2)

        NYA(1)=1
        NYA(2)=2

C ENDBEDINGUNGEN------------------------------------------
        W(3) = YB(3)
        W(4) = YB(4)+1.0D0

C SCHALTBEDINGUNGEN---------------------------------------
        W(5)=(ZZ(3)-ZZ(4))*ZZ(1)*0.5D0/((ZZ(3)
     F         -3.0D0*ZZ(4))*ZZ(2))-1.0D0

        NSK(5)=1

      RETURN
      END
```

Mit der Ausgabe:

```
   N= 4      M= 5      MS= 1
   PRESCRIBED RELATIVE PRECISION  0.10E-09
   MAXIMUM PERMITTED NUMBER OF ITERATIONS 20

*************************************************************************************
*************************************************************************************
 IT    ABS.ERR.   LEVEL1     LEVEL2     LEVEL3    RELAX. NEW RANK   COND(M)    NORM(M)
  0   0.89E-01   0.64E+00   0.70E+00   0.80E+00           0   25   0.11E+02   0.12E+01
      0.89E-01   0.64E+00   0.70E+00   0.80E+00  0.001
  1   0.89E-01   0.80E+00   0.86E+00   0.87E+00           1   25   0.12E+02   0.13E+01
      0.23E-01   0.87E-01   0.11E+00   0.82E-01  1.000
  2   0.23E-01   0.75E-01   0.88E-01   0.48E-01           2   25   0.11E+02   0.13E+01
      0.43E-02   0.19E-01   0.21E-01   0.77E-02  1.000
  3   0.43E-02   0.17E-01   0.22E-01   0.18E-01           3   25   0.98E+01   0.13E+01
      0.19E-03   0.74E-03   0.91E-03   0.68E-03  1.000
  4   0.19E-03   0.75E-03   0.10E-02   0.12E-02           4   25   0.10E+02   0.13E+01
      0.16E-05   0.12E-04   0.14E-04   0.12E-04  1.000
  5   0.16E-05   0.12E-04   0.15E-04   0.16E-04           5   25   0.10E+02   0.13E+01
      0.36E-06   0.23E-05   0.29E-05   0.25E-05  1.000
  6   0.36E-06   0.23E-05   0.26E-05   0.14E-05           6   25   0.11E+02   0.13E+01
      0.38E-07   0.25E-06   0.29E-06   0.18E-06  1.000
  7   0.38E-07   0.25E-06   0.35E-06   0.43E-06           7   25   0.11E+02   0.13E+01
      0.13E-09   0.51E-09   0.78E-09   0.10E-08  1.000
  8   0.13E-09   0.51E-09   0.75E-09   0.91E-09           8   25   0.11E+02   0.13E+01
      0.23E-12   0.29E-11   0.30E-11   0.68E-12  1.000
  9   0.23E-12   0.29E-11   0.30E-11   0.65E-12           9   25   0.11E+02   0.13E+01
      0.33E-14   0.39E-13   0.40E-13   0.61E-14  1.000
 10   0.33E-14   0.39E-13   0.40E-13   0.78E-14          10   25   0.11E+02   0.13E+01
      0.15E-17   0.71E-17   0.80E-17   0.38E-17  1.000
```

```
 11   0.15E-17   0.71E-17   0.79E-17   0.34E-17         11   25   0.11E+02   0.13E+01
      0.44E-19   0.19E-18   0.22E-18   0.11E-18  1.000
 12   0.44E-19   0.19E-18   0.23E-18   0.16E-18         12   25   0.11E+02   0.13E+01
      0.43E-23   0.19E-22   0.23E-22   0.23E-22  1.000
 13   0.43E-23   0.19E-22   0.23E-22   0.22E-22         13   25   0.11E+02   0.13E+01
****************************************************************************************
****************************************************************************************

   SOLUTION OBTAINED AFTER 14    ITERATION STEPS

   SOLUTION DATA

    0.00000000E+00  0.10000000E+01  0.00000000E+00 -0.28810479E+00 -0.39626078E+00
    0.10585067E+00  0.90426522E+00  0.85975292E-01 -0.27017515E+00 -0.50938814E+00
    0.25000000E+00  0.81776434E+00  0.15140751E+00 -0.23376384E+00 -0.64026586E+00
    0.50000000E+00  0.72583795E+00  0.21550334E+00 -0.16330065E+00 -0.78687867E+00
    0.75000000E+00  0.66160150E+00  0.25726246E+00 -0.85119789E-01 -0.90098417E+00
    0.10000000E+01  0.61149253E+00  0.28810479E+00 -0.12858848E-13 -0.10000000E+01


   SWITCHING POINTS
    0.10585067E+00

MIN =        0.28810478668175854
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 2,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     f   NNEBEN   = 2,
     g   NUGLNB   = 2,
     h   NRAND    = 0,
     i   NART     = 4,
     j   IPRINT   = 1000,
     k   imax     = 500,
     l   SCALE    = 0.25D0,
     m   R1       = 1.0D0,
     n   DEL      = -1.0D-6,
     o   EP       = 1.D-8,
     p   NZUSATZ  = NUNBE*NSTUETZ,
     q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     r   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     s   ME       = M-NDISKRET*NUGLNB,
     t   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

c Aequidistante Diskretisierung----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

c Startschaetzung der Steuerung----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
```

```
          u(i,j) = 0.0d0
200   CONTINUE

c Startschaetzung der freien Anfangswerte------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 0.0D0
300   CONTINUE

c Startschaetzung der hessematrix--------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0d0
1100      B1(I,J)=0.0D0

      do 1300 i=1,m
1300    EPS(I)=EP
      EPS(M+1)=EP

      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'min = ',ff
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = -x(2,ndiskret)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        dx(1) = -u(1)*x(1)+u(1)*u(1)*x(2)
        dx(2) =  u(1)*x(1)-3.0d0*u(1)*u(1)*x(2)

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        awx(1) = 1.0d0
        awx(2) = 0.0d0
```

```
      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)


       RETURN
       END



      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

        con(1) = -u(1)+1.0d0
        con(2) =  u(1)

      RETURN
      END
```

Mit der Ausgabe:

```
    1  -2.65217291010315485E-02
    2      -0.27371678732031984
    3      -0.27542854080378781
    4      -0.27893640544524290
    5      -0.28304554802185056
    6      -0.28411082544879179
    7      -0.28449317322355233
    8      -0.28503248167097650
    9      -0.28539579397123665
   10      -0.28561505552566629
   11      -0.28573706573013158
   12      -0.28580111399106389
   13      -0.28647093831642312
   14      -0.28653009561616510
   15      -0.28659776234413620
   16      -0.28683103932158627
   17      -0.28701631085183787
   18      -0.28711731166573146
   19      -0.28716677903828330
   20      -0.28719120363201883
   21      -0.28720359635742604
   22      -0.28720998296781708
   23      -0.28721329499251791
   24      -0.28721501567302560
   25      -0.28721590986454021
   26      -0.28721637477990736
   27      -0.28721661703346002
   28      -0.28721674406851594
   29      -0.28721681167133212
   30      -0.28721684877661707
   31      -0.28747426053804659
   32      -0.28763951896209183
   33      -0.28772695732702114
   34      -0.28777753336355755
   35      -0.28780427992142832
   36      -0.28781797422135458
```

```
    37       -0.28782490484327849
    38       -0.28782839779942687
    39       -0.28783015529975769
    40       -0.28783103886542805
    41       -0.28783148282574461
    42       -0.28783170580880335
    43       -0.28783181776636796
    44       -0.28783187396525489
    45       -0.28783190217607885
    46       -0.28783191634198546
    47       -0.28783192346181191
    48       -0.28783192705266075
    49       -0.28783192887534653
    50       -0.28783192980419403
    51       -0.28783193030283904
    52       -0.28797481765006749
    53       -0.28800034444730288
    54       -0.28801235898909550
    55       -0.28805622166854616
    56       -0.28806969895950441
    57       -0.28807164677600960
    58       -0.28807423876432497
    59       -0.28807994774908052
    60       -0.28808050751921921
    61       -0.28808089835890444
    62       -0.28808115144816193
    63       -0.28808153119373431
    64       -0.28808184774770890
    65       -0.28808188599679890
    66       -0.28808192272262745
    67       -0.28808199783014471
    68       -0.28808211951590690
    69       -0.28808213505003344
    70       -0.28808214317657038
    71       -0.28808215724825470
    72       -0.28808219403752422
    73       -0.28808219521966860
    74       -0.28808219673828184
    75       -0.28808220124940381
    76       -0.28808220141756713
    77       -0.28808220149284081
    78       -0.28808220162041653
    79       -0.28808220195490308
    80       -0.28808220196062018
    81       -0.28808220199385587
    82       -0.28808220200159268
    83       -0.28808220200342632
    84       -0.28808220200384599
    85       -0.28808220200466922
    86       -0.28808220200644424
    87       -0.28808220200682894
    88       -0.28808220200687823
    89       -0.28808220200689305
    90       -0.28808220200692652
    91       -0.28808220200697865
    92       -0.28808220200699791
    93       -0.28808220200699997
    94       -0.28808220200700041
    95       -0.28808220200700058
 LAGRANGE MULTIPLIKATOREN
  0.1381132D-02   0.1501298D-02
LOESUNG
  0.1000000D+01   0.1000000D+01   0.9506601D+00   0.7266054D+00
  0.6510701D+00   0.5859097D+00   0.5414915D+00   0.5059666D+00
```

```
    0.4774518D+00   0.4536519D+00   0.4334223D+00   0.4159136D+00
    0.4005539D+00   0.3869233D+00   0.3747104D+00   0.3636790D+00
    0.3536395D+00   0.0000000D+00   0.0000000D+00   0.0000000D+00
  FUNKTION
   -0.2880822D+00
  NEBENBEDINGUNGEN
    0.0000000D+00   0.0000000D+00   0.4933985D-01   0.2733946D+00
    0.3489299D+00   0.4140903D+00   0.4585085D+00   0.4940334D+00
    0.5225482D+00   0.5463481D+00   0.5665777D+00   0.5840864D+00
    0.5994461D+00   0.6130767D+00   0.6252896D+00   0.6363210D+00
    0.6463605D+00   0.1000000D+01   0.1000000D+01   0.9506601D+00
    0.7266054D+00   0.6510701D+00   0.5859097D+00   0.5414915D+00
    0.5059666D+00   0.4774518D+00   0.4536519D+00   0.4334223D+00
    0.4159136D+00   0.4005539D+00   0.3869233D+00   0.3747104D+00
    0.3636790D+00   0.3536395D+00
   128 FUNKTIONS AUFRUFE
    97 GRADIENTEN AUFRUFE
  KONVERGENT
min =      -0.28808220200700291
```

F.3 Anwendungsbeispiel 3

Das unbeschränkte Spline-Problem

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(15,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US

C DATENDEFINITION-----------------------------------------
      N=5
      M=4
      MS=0
      TOL=1E-12
      ITMAX=75
      KP=0
      KS=1
      NDISKRET=30
      ISCHALT=1

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIEN-------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.1D0
200   CONTINUE

C EXAKTE RANDBEDINGUNGEN-----------------------------------
        Y(1,1) = 0.0
        Y(1,M) = 0.0
        Y(2,1) = 1.0
        Y(2,M) =-1.0
        Y(3,1) = 0.0
```

```
      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,1E-7,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'MIN = ',Y(3,M)
      END



      SUBROUTINE F(X,Y,DY)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US

C DEFINITION DER STEUERUNG---------------------------------
        U = -Y(5)

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS-------------
        DY(1) = Y(2)
        DY(2) = U
        DY(3) = 0.5D0*U*U
        DY(4) = 0.0D0
        DY(5) = -Y(4)

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US

C ANFANGSBEDINGUNGEN-------------------------------------
        W(1) = YA(1)
        W(2) = YA(2)-1.0D0
        W(3) = YA(3)

        NYA(1) = 1
        NYA(2) = 2
        NYA(3) = 3

C ENDBEDINGUNGEN-----------------------------------------
          W(4) = YB(1)
          W(5) = YB(2)+1.0D0

C SCHALTBEDINGUNGEN--------------------------------------
C ES WERDEN KEINE SCHALTBEDINGUNGEN BENOETIGT

      RETURN
      END
```

Mit der Ausgabe:

```
   N= 5      M= 4      MS= 0
   PRESCRIBED RELATIVE PRECISION  0.10E-09
   MAXIMUM PERMITTED NUMBER OF ITERATIONS 75

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)   NORM(M)
 0  0.20E+01  0.27E+02  0.39E+03  0.11E+04          0    20  0.63E+02  0.58E+00
    0.20E+01  0.26E+02  0.38E+03  0.11E+04  0.010
 1  0.20E+01  0.26E+02  0.38E+03  0.11E+04          1    20  0.12E+03  0.11E+01
    0.73E+00  0.74E+02  0.74E+02  0.18E+03  1.000
```

```
 2  0.73E+00  0.87E+02  0.87E+02  0.18E+03          0    20   0.12E+03  0.11E+01
    0.43E-16  0.35E-14  0.35E-14  0.35E-14  1.000
 3  0.43E-16  0.35E-14  0.35E-14  0.35E-14          1    20   0.24E+03  0.23E+01
    0.55E-31  0.41E-29  0.50E-28  0.14E-27  1.000
 4  0.55E-31  0.73E-31  0.27E-30  0.67E-30          2    20   0.19E+02  0.24E+01
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER  5    ITERATION STEPS

MIN =       1.99999994039535700
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 3,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 0,
     G   NUGLNB   = 0,
     H   NRAND    = 2,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.0D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 0.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX--------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
```

```
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
      EPS(M+1)=EP

      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = 0.5D0*X(3,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = X(2)
        DX(2) = U(1)
        DX(3) = U(1)*U(1)

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = 1.0D0
        AWX(3) = 0.0D0

      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)
```

```
      R(1) = X(1,NDISKRET)
      R(2) = X(2,NDISKRET)+1.0D0

     RETURN
     END



    SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)


    RETURN
    END
```

Mit der Ausgabe:

```
   1   7.19755870052206648E-02
   2       0.66135992782722242
   3       1.06158911208759199
   4       1.36245806107364964
   5       1.57985125237029234
   6       1.72894582918916773
   7       1.82873146747016002
   8       1.89437738209834228
   9       1.93490842523156803
  10       1.96031645136486876
  11       1.97610683631137185
  12       1.98510793950614106
  13       1.99003896751300258
  14       1.99411973595030600
  15       1.99668160446088483
  16       1.99792484030450845
  17       1.99852411013086373
  18       1.99904207824553337
  19       1.99941328960808073
  20       1.99966144304431492
  21       1.99980454805734387
  22       1.99987748673892152
  23       1.99991428291612405
  24       1.99994295154766299
  25       1.99996565308364649
  26       1.99998043665681413
  27       1.99998881018773544
  28       1.99999299264780772
  29       1.99999510013541060
  30       1.99999682578003046
  31       1.99999817652762712
  32       1.99999904065682399
  33       1.99999949396544130
  34       1.99999968404578232
  35       1.99999975973728450
  36       1.99999984564491284
  37       1.99999992198933052
  38       1.99999997017335440
  39       1.99999998988417094
  40       1.99999999182642108
  41       1.99999998930537992
  42       1.99999999242385607
  43       1.99999999665950812
  44       1.99999999934141925
  45       2.00000000009301848
  46       1.99999999975960829
```

```
   47        1.99999999937034612
   48        1.99999999953114127
   49        1.99999999978585308
   50        1.99999999994482014
   51        1.99999999998536460
   52        1.99999999996428057
   53        1.99999999994475375
   54        1.99999999995793121
   55        1.99999999997486744
   56        1.99999999998610445
   57        1.99999999999235678
   58        1.99999999999534372
   59        1.99999999999639688
   60        1.99999999999730860
   61        1.99999999999825140
   62        1.99999999999892397
   63        1.99999999999934119
   64        1.99999999999958300
   65        1.99999999999972844
   66        1.99999999999982547
   67        1.99999999999989009
   68        1.99999999999993161
       LAGRANGE MULTIPLIKATOREN
        0.2169391D-08  -0.1000000D+01
      LOESUNG
       -0.2000000D+01  -0.2000000D+01  -0.2000000D+01  -0.2000000D+01
       -0.2000000D+01  -0.2000000D+01  -0.2000000D+01  -0.2000000D+01
       -0.2000000D+01  -0.2000000D+01  -0.2000000D+01  -0.2000000D+01
       -0.2000000D+01  -0.2000000D+01  -0.2000000D+01  -0.2000000D+01
       -0.2000000D+01   0.0000000D+00   0.0000000D+00   0.0000000D+00
      FUNKTION
        0.2000000D+01
      NEBENBEDINGUNGEN
        0.1434027D-15   0.2176037D-13
        70 FUNKTIONS AUFRUFE
        70 GRADIENTEN AUFRUFE
      KONVERGENT
min =       1.99999999999995715
```

Das durch $x_1(t) \leq 0.2$ beschränkte Spline-Problem

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(15,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US

C DATENDEFINITION------------------------------------------
      N=6
      M=4
      MS=1
      TOL=1.0D-20
      ITMAX=75
      KP=0
      KS=1
      NDISKRET=30
      ISCHALT=1

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
```

```
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIEN------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.1D0
200   CONTINUE

C EXAKTE RANDWERTE----------------------------------------
        Y(1,1) = 0.0
        Y(1,M) = 0.0
        Y(2,1) = 1.0
        Y(2,M) =-1.0
        Y(3,1) = 0.0

C SCHAETZUNG DES SCHALTPUNKTES----------------------------
        XS(1) = 0.49D0

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,1E-7,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'MIN = ',Y(3,M)
      END



      SUBROUTINE F(X,Y,DY)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US

C DEFINITION DER STEUERUNG--------------------------------
        U = -Y(5)

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYTEMS-------------
        DY(1) = Y(2)
        DY(2) = U
        DY(3) = 0.5D0*U*U
        DY(4) = 0.0D0
        DY(5) = -Y(4)
        DY(6) = 0.0D0

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US

C ANFANGSBEDINGUNGEN--------------------------------------
        W(1) = YA(1)
        W(2) = YA(2)-1.0D0
        W(3) = YA(3)

        NYA(1) = 1
        NYA(2) = 2
        NYA(3) = 3

C ENDBEDINGUNGEN------------------------------------------
        W(4) = YB(1)
        W(5) = YB(2)+1.0D0
```

```
C SCHALTBEDINGUNGEN---------------------------------------
        W(6) = ZZ(1)/0.2D0-1.0D0
        W(7) = ZZ(2)

        NSK(6) = 1
        NSK(7) = 1

C SPRUNGBEDINGUNGEN---------------------------------------
        IF (LS.EQ.1) THEN
          ZZ(4)=ZZ(4)-ZZ(6)
        ENDIF

      RETURN
      END
```

Mit der Ausgabe:

```
   N= 6      M= 4      MS= 1
   PRESCRIBED RELATIVE PRECISION  0.10E-09
   MAXIMUM PERMITTED NUMBER OF ITERATIONS 75

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)    NORM(M)
 0  0.22E+01  0.28E+02  0.19E+05  0.54E+05          0   28  0.22E+04  0.69E+00
    0.21E+01  0.27E+02  0.18E+05  0.53E+05  0.010
 1  0.21E+01  0.83E+02  0.17E+05  0.50E+05          1   28  0.33E+04  0.11E+01
    0.79E+00  0.36E+02  0.13E+04  0.38E+04  0.431
 2  0.79E+00  0.29E+02  0.54E+03  0.16E+04          2   28  0.20E+04  0.11E+01
    0.29E+01  0.27E+03  0.12E+06  0.37E+06  1.000
    0.61E+00  0.21E+02  0.56E+03  0.17E+04  0.100
 3  0.61E+00  0.16E+02  0.18E+02  0.24E+03          0   28  0.26E+03  0.74E+01
    0.37E+00  0.99E+01  0.11E+02  0.15E+03  0.227
 4  0.37E+00  0.78E+01  0.86E+01  0.84E+02          1   28  0.17E+03  0.81E+01
    0.47E-01  0.40E+01  0.40E+01  0.10E+02  1.000
 5  0.47E-01  0.40E+01  0.40E+01  0.15E+02          2   28  0.47E+03  0.11E+02
    0.88E-01  0.13E+01  0.14E+01  0.44E+01  1.000
 6  0.88E-01  0.94E-01  0.11E+00  0.63E-01          0   28  0.15E+03  0.14E+02
    0.11E-02  0.12E-02  0.19E-02  0.42E-02  1.000
 7  0.11E-02  0.12E-02  0.18E-02  0.36E-02          1   28  0.14E+03  0.14E+02
    0.26E-04  0.34E-04  0.38E-04  0.44E-04  1.000
 8  0.26E-04  0.33E-04  0.36E-04  0.38E-04          2   28  0.14E+03  0.14E+02
    0.33E-07  0.21E-06  0.29E-06  0.39E-06  1.000
 9  0.33E-07  0.21E-06  0.31E-06  0.46E-06          3   28  0.14E+03  0.14E+02
    0.68E-08  0.19E-07  0.20E-07  0.49E-08  1.000
10  0.68E-08  0.19E-07  0.20E-07  0.46E-08          4   28  0.15E+03  0.14E+02
    0.37E-10  0.60E-10  0.91E-10  0.19E-09  1.000
11  0.37E-10  0.60E-10  0.83E-10  0.15E-09          5   28  0.14E+03  0.14E+02
    0.42E-12  0.73E-12  0.15E-11  0.54E-11  1.000
12  0.42E-12  0.73E-12  0.16E-11  0.60E-11          6   28  0.14E+03  0.14E+02
    0.17E-14  0.11E-14  0.31E-14  0.63E-14  1.000
13  0.17E-14  0.11E-14  0.32E-14  0.66E-14          7   28  0.14E+03  0.14E+02
    0.62E-18  0.75E-18  0.12E-17  0.28E-17  1.000
14  0.62E-18  0.75E-18  0.12E-17  0.29E-17          8   28  0.14E+03  0.14E+02
    0.68E-23  0.91E-23  0.18E-22  0.33E-22  1.000
15  0.68E-23  0.91E-23  0.18E-22  0.34E-22          9   28  0.14E+03  0.14E+02
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER 16    ITERATION STEPS
```

```
   SWITCHING POINTS
   0.5000000E+00

MIN =        2.23999999046328391
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 3,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 1,
     G   NUGLNB   = 1,
     H   NRAND    = 2,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.0D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 0.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX--------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
      EPS(M+1)=EP
```

```
      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1             N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = 0.5D0*X(3,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = X(2)
        DX(2) = U(1)
        DX(3) = U(1)*U(1)

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = 1.0D0
        AWX(3) = 0.0D0

      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)

        R(1) = X(1,NDISKRET)
        R(2) = X(2,NDISKRET)+1.0D0

       RETURN
       END
```

```
      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

         CON(1) = -X(1)+0.2D0

      RETURN
      END
```

Mit der Ausgabe:

```
  1     0.11863097944239904
  2     0.62961623531712374
  3     1.05048185751404488
  4     1.35118771092879109
  5     1.61144161852106538
  6     1.78024597115049210
  7     1.88412201765659026
  8     1.96863842906938702
  9     2.04457905139818852
 10     2.09835491086024728
 11     2.13897231822044498
 12     2.17610002283571768
 13     2.19768759717041240
 14     2.21104643219467212
 15     2.21913365748867708
 16     2.22439438892878449
 17     2.23219642504974525
 18     2.23580063297970533
 19     2.23771735572255448
 20     2.23774723968434186
 21     2.23861511021919934
 22     2.23918939153223384
 23     2.23952238114794122
 24     2.23971010000642634
 25     2.23981021066271380
 26     2.23987680189161109
 27     2.23992453872425301
 28     2.23995463211088675
 29     2.23997210124794721
 30     2.23998351046809319
 31     2.23999110883315256
 32     2.23999552273074665
 33     2.23999732071396540
 34     2.23999789064970622
 35     2.23999850015894442
 36     2.23999919135161596
 37     2.23999967036469316
 38     2.23999991700150902
 39     2.23999997917193694
 40     2.23999993328984548
 41     2.23999991214279781
 42     2.23999993940893249
 43     2.23999996791572276
 44     2.23999998589072336
 45     2.23999999407174277
 46     2.23999999619043777
 47     2.23999999582577880
 48     2.23999999688126827
 49     2.23999999837466612
 50     2.23999999938317895
 51     2.23999999978672015
 52     2.23999999980330378
```

```
          53        2.23999999971890595
          54        2.23999999977537279
          55        2.23999999986905607
          56        2.23999999993362620
          57        2.23999999996691113
          58        2.23999999997920307
          59        2.23999999998141686
          60        2.23999999998588661
          61        2.23999999999137955
          62        2.23999999999521160
          63        2.23999999999729260
          64        2.23999999999818611
          65        2.23999999999858312
          66        2.23999999999900412
          67        2.23999999999937138
          68        2.23999999999961430
          69        2.23999999999976307
          70        2.23999999999985278
          71        2.23999999999990473
        LAGRANGE MULTIPLIKATOREN
         0.2400000D+01  -0.1600000D+01   0.4800000D+01
       LOESUNG
        -0.3200000D+01  -0.2900000D+01  -0.2600000D+01  -0.2300000D+01
        -0.2000000D+01  -0.1700000D+01  -0.1400000D+01  -0.1100000D+01
        -0.8000000D+00  -0.1100000D+01  -0.1400000D+01  -0.1700000D+01
        -0.2000000D+01  -0.2300000D+01  -0.2600000D+01  -0.2900000D+01
        -0.3200000D+01   0.0000000D+00   0.0000000D+00   0.0000000D+00
       FUNKTION
         0.2240000D+01
       NEBENBEDINGUNGEN
        -0.2339563D-14   0.1665335D-14   0.2000000D+00   0.1435547D+00
         0.9843750D-01   0.6347656D-01   0.3750000D-01   0.1933594D-01
         0.7812500D-02   0.1757813D-02  -0.4773959D-14   0.1757812D-02
         0.7812500D-02   0.1933594D-01   0.3750000D-01   0.6347656D-01
         0.9843750D-01   0.1435547D+00   0.2000000D+00
         73 FUNKTIONS AUFRUFE
         73 GRADIENTEN AUFRUFE
       KONVERGENT
min =       2.23999999999993848
```

Das durch $x_1(t) \leq 0.1$ beschränkte Spline-Problem

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(15,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US

C DATENDEFINITION------------------------------------------
      N=7
      M=3
      MS=2
      TOL=1E-12
      ITMAX=75
      KP=0
      KS=1
      NDISKRET=30
      ISCHALT=1

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
```

```
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIEN------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.1D0
200   CONTINUE

C EXAKTE RANDWERTE----------------------------------------
        Y(1,1) = 0.0
        Y(1,M) = 0.0
        Y(2,1) = 1.0
        Y(2,M) =-1.0
        Y(3,1) = 0.0

C STARTSCHAETZUNG DER SCHALTPUNKTE------------------------
        XS(1)=0.29
        XS(2)=0.71

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,1E-7,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'MIN = ',Y(3,M)
      END



      SUBROUTINE F(X,Y,DY)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US

C DEFINITION DER STEUERUNG--------------------------------
        IF (US(J,1).EQ.-1) U = -Y(5)
        IF ((US(J,1).EQ.1).AND.(US(J,2).EQ.-1)) U = 0.0D0
        IF (US(J,2).EQ.1) U = -Y(5)

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS------------
        DY(1) = Y(2)
        DY(2) = U
        DY(3) = 0.5D0*U*U
        DY(4) = 0.0D0
        DY(5) = -Y(4)
        DY(6) = 0.0D0
        DY(7) = 0.0D0

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US

C ANFANGSBEDINGUNGEN--------------------------------------
        W(1) = YA(1)
        W(2) = YA(2)-1.0D0
        W(3) = YA(3)

        NYA(1) = 1
        NYA(2) = 2
```

```
      NYA(3) = 3

C ENDBEDINGUNGEN------------------------------------------
      W(4) = YB(1)
      W(5) = YB(2)+1.0D0

C SCHALTBEDINGUNGEN---------------------------------------
      W(6) = ZZ(1)/0.1D0-1.0D0
      W(7) = ZZ(2)
      W(8) = ZZ(5)
      W(9) = ZZ(5)

      NSK(6) = 1
      NSK(7) = 1
      NSK(8) = 1
      NSK(9) = 2

C SPRUNGBEDINGUNGEN---------------------------------------
      IF (LS.EQ.1) THEN
        ZZ(4)=ZZ(4)-ZZ(6)
      ENDIF
      IF (LS.EQ.1) THEN
        ZZ(5)=ZZ(5)-ZZ(7)
      ENDIF

     RETURN
     END
```

Mit der Ausgabe:

```
   N= 7      M= 3      MS= 2
   PRESCRIBED RELATIVE PRECISION  0.10E-09
   MAXIMUM PERMITTED NUMBER OF ITERATIONS 75

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)   NORM(M)
 0  0.66E+01  0.15E+03  0.30E+06  0.59E+06          0    27  0.32E+03  0.16E+01
    0.64E+01  0.15E+03  0.29E+06  0.58E+06  0.010
 1  0.64E+01  0.26E+02  0.29E+06  0.58E+06          1    27  0.32E+03  0.16E+01
    0.48E+01  0.48E+03  0.65E+03  0.83E+03  1.000
 2  0.48E+01  0.43E+03  0.19E+04  0.33E+04          0    27  0.13E+05  0.14E+02
    0.72E-03  0.49E-01  0.32E+01  0.63E+01  1.000
 3  0.72E-03  0.95E-03  0.77E-03  0.73E-03          1    27  0.15E+04  0.56E+02
    0.84E-06  0.18E-05  0.13E-05  0.77E-06  1.000
 4  0.84E-06  0.67E-06  0.16E-06  0.46E-06          2    27  0.14E+04  0.54E+02
    0.88E-09  0.55E-09  0.18E-09  0.21E-09  1.000
 5  0.88E-09  0.55E-09  0.18E-09  0.21E-09          3    27  0.14E+04  0.54E+02
    0.98E-11  0.50E-11  0.13E-11  0.20E-11  1.000
 6  0.98E-11  0.50E-11  0.12E-11  0.17E-11          4    27  0.14E+04  0.54E+02
    0.46E-14  0.12E-14  0.13E-14  0.27E-14  1.000
 7  0.46E-14  0.12E-14  0.13E-14  0.27E-14          5    27  0.14E+04  0.54E+02
    0.26E-17  0.18E-17  0.41E-17  0.12E-16  1.000
 8  0.26E-17  0.18E-17  0.37E-17  0.11E-16          6    27  0.14E+04  0.54E+02
    0.25E-19  0.73E-20  0.12E-20  0.37E-21  1.000
 9  0.25E-19  0.73E-20  0.12E-20  0.37E-21          7    27  0.14E+04  0.54E+02
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER 10    ITERATION STEPS


   SWITCHING POINTS
```

```
  0.3000000E+00 0.7000000E+00

MIN =       4.44444444444376252
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 3,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 1,
     G   NUGLNB   = 1,
     H   NRAND    = 2,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.0D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 0.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX--------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
      EPS(M+1)=EP
```

```
      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = 0.5D0*X(3,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = X(2)
        DX(2) = U(1)
        DX(3) = U(1)*U(1)

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = 1.0D0
        AWX(3) = 0.0D0

      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)

        R(1) = X(1,NDISKRET)
        R(2) = X(2,NDISKRET)+1.0D0

       RETURN
       END
```

```
      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

         CON(1) = -X(1)+0.1D0

      RETURN
      END
```

Mit der Ausgabe:

```
  1       0.13725168515299913
  2       0.71538315630791627
  3       1.17403287341236551
  4       1.47825333484290367
  5       1.76563663384719871
  6       2.24059463646067236
  7       2.74660577701605702
  8       3.16948813465996926
  9       3.52441187483953700
 10       3.80035369138433099
 11       4.01194899394041204
 12       4.16591626477833188
 13       4.27578516290486110
 14       4.34244084381750195
 15       4.38361380393627442
 16       4.40793323919777791
 17       4.41529534585902539
 18       4.41864293332909064
 19       4.42720558071353221
 20       4.43434345637486960
 21       4.43816831961147962
 22       4.43922342715387241
 23       4.44083377612836117
 24       4.44176181894548172
 25       4.44204473789821463
 26       4.44203784713620742
 27       4.44114904640578967
 28       4.44076290245937688
 29       4.44079416002897087
 30       4.44076492971392689
 31       4.44074584369729486
 32       4.44073519338847333
 33       4.44072962895418488
 34       4.44072678214525940
 35       4.44072533785442314
 36       4.44072460790830625
 37       4.44072423966589547
 38       4.44072405400844783
 39       4.44072396035892769
 40       4.44075437567771569
 41       4.44074954300119895
 42       4.44075561605604996
 43       4.44074648154367946
 44       4.44074770510040473
 45       4.44075090495977687
 46       4.44075344637790259
 47       4.44075462001812138
 48       4.44075481545469497
 49       4.44075454511819334
 50       4.44075453676613829
 51       4.44075470879298262
 52       4.44075483987941144
 53       4.44075487046316386
```

```
         54        4.44075481950600182
         55        4.44075474289817329
         56        4.44075471687146539
         57        4.44075472257428938
         58        4.44075472920341241
         59        4.44075472907020430
         60        4.44075472369355762
         61        4.44075471796874499
         62        4.44075471656320442
         63        4.44075471706098224
         64        4.44075471751863127
         65        4.44075471757722795
         66        4.44075471732768445
         67        4.44075471698100710
         68        4.44075471686715328
         69        4.44075471689411927
         70        4.44075471692829815
         71        4.44075471695022106
         72        4.44075471696243618
         73        4.44075471696900781
         74        4.44075471697260937
         75        4.44075471697547908
         76        4.44075471697756452
       LAGRANGE MULTIPLIKATOREN
        0.1077736D+02  -0.3298113D+01   0.6339620D+00   0.1014340D+02
        0.1014340D+02   0.6339624D+00
      LOESUNG
       -0.6596226D+01  -0.5249057D+01  -0.3901887D+01  -0.2554717D+01
       -0.1207547D+01   0.6037739D-01   0.6037726D-01   0.6037751D-01
        0.6037721D-01   0.6037746D-01   0.6037732D-01   0.6037736D-01
       -0.1207547D+01  -0.2554717D+01  -0.3901887D+01  -0.5249057D+01
       -0.6596226D+01   0.0000000D+00   0.0000000D+00   0.0000000D+00
      FUNKTION
        0.4440755D+01
      NEBENBEDINGUNGEN
       -0.3594116D-13   0.1110223D-13   0.1000000D+00   0.4950619D-01
        0.1951651D-01   0.4768573D-02  -0.2123302D-14  -0.3394507D-13
        0.5896226D-03   0.9433963D-03   0.1061321D-02   0.9433963D-03
        0.5896227D-03  -0.3398670D-13  -0.2192690D-14   0.4768573D-02
        0.1951651D-01   0.4950619D-01   0.1000000D+00
        78 FUNKTIONS AUFRUFE
        78 GRADIENTEN AUFRUFE
      KONVERGENT
min =       4.44075471697890212
```

F.4 Anwendungsbeispiel 4

Das Knickstab-Problem für $\alpha = 9.90027$

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(16,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C DATENDEFINITION------------------------------------------
      N=7
      M=6
```

```
      MS=1
      TOL=1E-12
      ITMAX=75
      KP=0
      KS=1
      ISCHALT=1
      NDISKRET=30

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIEN-------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.05D0
200   CONTINUE

C EXAKTE RANDWERTE-----------------------------------------
      Y(1,1) = 0.0
      Y(1,M) = 0.0
      Y(3,1) = 0.0
      Y(4,1) = 0.0
      Y(6,1) = 0.0
      Y(6,M) = 0.0

C STARTSCHAETZUNG DES SCHALTPUNKTES------------------------
      XS(1)=0.49D0

      ALPHA = 9.90027D0

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,TOL,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'MIN = ',0.5D0*Y(3,M)+ALPHA*Y(4,M)

      STOP
      END



      SUBROUTINE F(X,Y,DY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C DEFINITION DER STEUERUNG---------------------------------
        U=-Y(6)

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS-------------
        DY(1) = DSIN(Y(2))
        DY(2) = U
        DY(3) = U*U
        DY(4) = DCOS(Y(2))
        DY(5) = 0.0D0
        DY(6) = -Y(5)*DCOS(Y(2))+ALPHA*DSIN(Y(2))
        DY(7) = 0.0D0

      RETURN
      END
```

```
      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C ANFANGSBEDINGUNGEN-------------------------------------
        W(1) = YA(1)
        W(2) = YA(3)
        W(3) = YA(4)
        W(4) = YA(6)

        NYA(1) = 1
        NYA(3) = 2
        NYA(4) = 3
        NYA(6) = 4

C ENDBEDINGUNGEN-----------------------------------------
        W(5) = YB(1)
        W(6) = YB(6)

C SCHALTBEDINGUNGEN--------------------------------------
        W(7) = ZZ(1)-0.05D0
        W(8) = ZZ(2)

        NSK(7) = 1
        NSK(8) = 1

C SPRUNGBEDINGUNGEN--------------------------------------
        IF (LS.EQ.1) THEN
          ZZ(5)=ZZ(5)-ZZ(7)
        ENDIF

      RETURN
      END
```

Mit der Ausgabe:

```
   N= 7      M= 6      MS= 1
   PRESCRIBED RELATIVE PRECISION  0.10E-09
   MAXIMUM PERMITTED NUMBER OF ITERATIONS 75

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)   NORM(M)
 0  0.19E+00  0.10E+03  0.11E+03  0.99E+03           0   48  0.27E+03  0.69E+00
    0.19E+00  0.10E+03  0.11E+03  0.97E+03  0.010
 1  0.19E+00  0.99E+02  0.11E+03  0.97E+03           1   48  0.40E+03  0.10E+01
    0.17E-02  0.55E+00  0.12E+01  0.31E+02  1.000
 2  0.17E-02  0.74E+00  0.11E+01  0.45E+02           2   48  0.39E+03  0.10E+01
    0.11E-01  0.37E+00  0.12E+02  0.75E+03  1.000
 3  0.11E-01  0.53E+00  0.12E+01  0.90E+01           0   48  0.99E+03  0.10E+01
    0.65E-02  0.31E+00  0.57E+00  0.40E+01  0.257
 4  0.65E-02  0.14E+00  0.82E+00  0.28E+01           1   48  0.13E+04  0.10E+01
    0.29E-03  0.28E-01  0.16E+01  0.77E+01  1.000
 5  0.29E-03  0.38E-01  0.93E+00  0.52E+01           0   48  0.14E+04  0.10E+01
    0.76E-05  0.11E-02  0.13E-01  0.66E-01  1.000
 6  0.76E-05  0.11E-02  0.16E-01  0.82E-01           1   48  0.14E+04  0.10E+01
    0.41E-06  0.94E-06  0.54E-04  0.49E-03  1.000
 7  0.41E-06  0.77E-06  0.47E-04  0.38E-03           2   48  0.13E+04  0.10E+01
    0.38E-08  0.81E-06  0.14E-04  0.63E-04  1.000
```

```
 8  0.38E-08  0.79E-06  0.26E+00  0.13E+01          0    48  0.43E+05  0.10E+01
    0.15E-15  0.80E-14  0.11E-07  0.57E-07  1.000
 9  0.15E-15  0.80E-14  0.11E-07  0.57E-07          1    48  0.43E+05  0.10E+01
    0.76E-23  0.24E-20  0.90E-15  0.45E-14  1.000
10  0.76E-23  0.24E-20  0.31E-10  0.15E-09          2    48  0.11E+08  0.14E+01
    0.10E-29  0.28E-27  0.67E-17  0.34E-16  1.000
11  0.10E-29  0.28E-27  0.69E-17  0.34E-16          3    48  0.11E+08  0.14E+01
    0.51E-31  0.37E-30  0.53E-16  0.26E-15  1.000
12  0.51E-31  0.37E-30  0.53E-16  0.26E-15          0    48  0.11E+08  0.14E+01
    0.98E-31  0.63E-30  0.28E-18  0.14E-17  1.000
13  0.98E-31  0.63E-30  0.28E-18  0.14E-17          0    48  0.11E+08  0.14E+01
    0.21E-31  0.24E-30  0.55E-17  0.28E-16  1.000
14  0.21E-31  0.24E-30  0.55E-17  0.28E-16          0    48  0.11E+08  0.14E+01
    0.44E-31  0.60E-30  0.47E-16  0.23E-15  1.000
    0.57E-31  0.41E-30  0.20E-16  0.10E-15  0.255
    0.16E-31  0.21E-30  0.21E-18  0.11E-17  0.050
15  0.16E-31  0.21E-30  0.37E-21  0.18E-20          1    48  0.11E+08  0.14E+01
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER 16    ITERATION STEPS



   SWITCHING POINTS
   0.5000000E+00

MIN =        9.90017508686360159
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 4,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 2,
     G   NUGLNB   = 2,
     H   NRAND    = 1,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
```

```
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.1D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 0.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX-------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
      EPS(M+1)=EP

      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = 0.5D0*X(3,NDISKRET)+9.90027D0*X(4,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = DSIN(X(2))
        DX(2) = U(1)
        DX(3) = U(1)*U(1)
        DX(4) = DCOS(X(2))

      RETURN
      END
```

```
      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = UNKNOWN(1)
        AWX(3) = 0.0D0
        AWX(4) = 0.0D0

      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)

        R(1) = X(1,NDISKRET)

       RETURN
       END



      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

        CON(1) = -X(1)+0.05D0
        CON(2) =  X(1)+0.05D0

      RETURN
      END
```

Mit der Ausgabe:

```
    1       9.89601621670883347
    2       9.89927999469165876
    3       9.90035018314736881
    4       9.90066295091069470
    5       9.90073054627498550
    6       9.90072153432933000
    7       9.90069059998126377
    8       9.90065525302877880
    9       9.90058762948166304
   10       9.90028474636444145
   11       9.90028323156404433
   12       9.90027948767520982
   13       9.90027177924848623
   14       9.90026211793505517
   15       9.90026073966301468
   16       9.90026038999736535
   17       9.90025984615086330
   18       9.90025790510386727
   19       9.90025786596651614
   20       9.90025783082181832
   21       9.90025760509946551
   22       9.90025756826399572
   23       9.90025754869789054
   24       9.90025752110668300
   25       9.90025685686821433
   26       9.90025639150120007
```

```
   27        9.90025609123630268
   28        9.90025579311586057
   29        9.90025436973669493
   30        9.90024925523080057
   31        9.90023830835776764
   32        9.90021638733075093
   33        9.90020246706468043
   34        9.90020136821339491
   35        9.90019643417225836
   36        9.90018164863830030
   37        9.90017801810399689
   38        9.90017701725889587
   39        9.90017667321609984
   40        9.90017650883540767
   41        9.90017640101105378
   42        9.90017632477850640
   43        9.90017627173162396
   44        9.90017623536346214
   45        9.90017621060167485
   46        9.90017619409551308
   47        9.90017618207069283
   48        9.90017617486521750
   49        9.90017616028698022
   50        9.90017495045009888
   51        9.90017492275142708
   52        9.90017491825200402
   53        9.90017491097320068
   54        9.90017488262202683
   55        9.90017487868076174
   56        9.90017487862236578
   57        9.90017487838126442
   58        9.90017487838075283
   59        9.90017487838091093
   60        9.90017487838099264
   61        9.90017487838103705
   62        9.90017487838106192
   63        9.90017487838107790
   64        9.90017487838108146
   65        9.90017487838103172
   66        9.90017487838099974
   67        9.90017487838099086
   68        9.90017487838099086
   69        9.90017487838098908
   70        9.90017487838098376
   71        9.90017487838097132
   72        9.90017487838096955
   73        9.90017487838096599
 LAGRANGE MULTIPLIKATOREN
 -0.6381807D-06  0.1273819D-05
LOESUNG
  0.4461349D-04  0.9672375D-01  0.1897897D+00  0.2755977D+00
  0.3508775D+00  0.4127138D+00  0.4587017D+00  0.4870390D+00
  0.4966108D+00  0.4870392D+00  0.4587015D+00  0.4127138D+00
  0.3508776D+00  0.2755977D+00  0.1897898D+00  0.9672369D-01
  0.4482189D-04  0.0000000D+00  0.0000000D+00 -0.1574857D+00
FUNKTION
  0.9900175D+01
NEBENBEDINGUNGEN
  0.1880911D-10  0.5000000D-01  0.5973996D-01  0.6910883D-01
  0.7774871D-01  0.8532790D-01  0.9155356D-01  0.9618347D-01
  0.9903636D-01  0.1000000D+00  0.9903636D-01  0.9618347D-01
  0.9155356D-01  0.8532790D-01  0.7774871D-01  0.6910883D-01
  0.5973996D-01  0.5000000D-01  0.5000000D-01  0.4026004D-01
  0.3089117D-01  0.2225129D-01  0.1467210D-01  0.8446445D-02
```

```
        0.3816531D-02   0.9636354D-03  -0.3824396D-10   0.9636352D-03
        0.3816531D-02   0.8446445D-02   0.1467210D-01   0.2225129D-01
        0.3089117D-01   0.4026004D-01   0.5000000D-01
       101 FUNKTIONS AUFRUFE
        75 GRADIENTEN AUFRUFE
      KONVERGENT
min =       9.90017487838096599
```

Das Knickstab-Problem für $\alpha = 39,3291$

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(16,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C DATENDEFINITION------------------------------------------
      N=7
      M=6
      MS=1
      TOL=1E-8
      ITMAX=75
      KP=0
      KS=1
      ISCHALT=1
      NDISKRET=30

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIEN-------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.05D0
200   CONTINUE

C EXAKTE RANDWERTE-----------------------------------------
        Y(1,1) = 0.0
        Y(1,M) = 0.0
        Y(3,1) = 0.0
        Y(4,1) = 0.0
        Y(6,1) = 0.0
        Y(6,M) = 0.0

C STARTSCHAETZUNG DER SCHALTPUNKTE-------------------------
        XS(1)=0.49D0

      ALPHA = 39.3291D0

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,TOL,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'min = ',0.5D0*Y(3,M)+ALPHA*Y(4,M)
      STOP
      END
```

```
      SUBROUTINE F(X,Y,DY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C DEFINITION DER STEUERUNG---------------------------------
        U=-Y(6)

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS------------
        DY(1) = DSIN(Y(2))
        DY(2) = U
        DY(3) = U*U
        DY(4) = DCOS(Y(2))
        DY(5) = 0.0D0
        DY(6) = -Y(5)*DCOS(Y(2))+ALPHA*DSIN(Y(2))
        DY(7) = 0.0D0

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C ANFANGSBEDINGUNGEN--------------------------------------
        W(1) = YA(1)
        W(2) = YA(3)
        W(3) = YA(4)
        W(4) = YA(6)

        NYA(1) = 1
        NYA(3) = 2
        NYA(4) = 3
        NYA(6) = 4

C ENDBEDINGUNGEN------------------------------------------
        W(5) = YB(1)
        W(6) = YB(6)

C SCHALTBEDINGUNGEN
        W(7) = ZZ(1)-0.05D0
        W(8) = ZZ(2)

        NSK(7) = 1
        NSK(8) = 1

C SPRUNGBEDINGUNGEN---------------------------------------
        IF (LS.EQ.1) THEN
          ZZ(5)=ZZ(5)-ZZ(7)
        ENDIF

      RETURN
      END
```

Mit der Ausgabe:

```
   N= 7      M= 6      MS= 1
```

```
   PRESCRIBED RELATIVE PRECISION  0.10E-07
   MAXIMUM PERMITTED NUMBER OF ITERATIONS 75

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)   NORM(M)
 0  0.51E+00  0.32E+03  0.92E+04  0.47E+05          0    48  0.18E+04  0.69E+00
    0.50E+00  0.31E+03  0.91E+04  0.46E+05  0.010
 1  0.50E+00  0.31E+03  0.21E+05  0.11E+06          0    48  0.49E+04  0.10E+01
    0.48E+00  0.30E+03  0.21E+05  0.10E+06  0.018
 2  0.48E+00  0.30E+03  0.31E+05  0.15E+06          1    48  0.56E+04  0.10E+01
    0.41E+00  0.25E+03  0.25E+05  0.13E+06  0.084
 3  0.41E+00  0.25E+03  0.19E+05  0.96E+05          2    48  0.51E+04  0.10E+01
    0.10E+03  0.41E+05  0.88E+06  0.43E+07  1.000
    0.28E+00  0.17E+03  0.25E+05  0.12E+06  0.157
 4  0.28E+00  0.17E+03  0.21E+05  0.11E+06          0    48  0.74E+04  0.10E+01
    0.16E+00  0.14E+03  0.29E+04  0.14E+05  1.000
 5  0.16E+00  0.70E+01  0.15E+04  0.77E+04          1    48  0.54E+04  0.10E+01
    0.88E+02  0.35E+05  0.41E+05  0.40E+05  1.000
    0.41E-03  0.11E+00  0.58E+01  0.29E+02  0.295
 6  0.41E-03  0.11E+00  0.11E+00  0.18E+00          0    48  0.13E+04  0.11E+02
    0.44E-06  0.25E-04  0.25E-04  0.37E-04  1.000
 7  0.44E-06  0.22E-04  0.22E-04  0.31E-04          1    48  0.14E+04  0.11E+02
    0.57E-11  0.95E-09  0.98E-09  0.38E-08  1.000
 8  0.57E-11  0.11E-08  0.12E-08  0.38E-08          2    48  0.13E+04  0.11E+02
    0.14E-14  0.60E-12  0.61E-12  0.84E-12  1.000
 9  0.14E-14  0.60E-12  0.61E-12  0.85E-12          3    48  0.13E+04  0.11E+02
    0.22E-18  0.31E-16  0.32E-16  0.98E-16  1.000
10  0.22E-18  0.31E-16  0.32E-16  0.96E-16          4    48  0.13E+04  0.11E+02
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER 11    ITERATION STEPS


   SWITCHING POINTS
   0.5000000E+00

min =       39.13189944886674710
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 4,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 2,
     G   NUGLNB   = 2,
     H   NRAND    = 1,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
```

```
     S   ME        = M-NDISKRET*NUGLNB,
     T   MAXNM     = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.1D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 0.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX-------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
      EPS(M+1)=EP

      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = 0.5D0*X(3,NDISKRET)+39.3291D0*X(4,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = DSIN(X(2))
        DX(2) = U(1)
        DX(3) = U(1)*U(1)
        DX(4) = DCOS(X(2))

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = UNKNOWN(1)
        AWX(3) = 0.0D0
        AWX(4) = 0.0D0

      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F           T(NDISKRET)

        R(1) = X(1,NDISKRET)

       RETURN
       END



      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

        CON(1) = -X(1)+0.05D0
        CON(2) =  X(1)+0.05D0

      RETURN
      END
```

Mit der Ausgabe:

```
        1       39.29286890841993340
        2       39.29958709258175280
        3       39.29486251286548540
        4       39.27292691886557920
        5       39.22423084177101060
        6       39.18670329001790260
        7       39.15664860051880680
        8       39.13696285848853050
        9       39.13460163176522140
       10       39.12841579266373770
       11       39.12910063569287900
       12       39.12999619830152650
       13       39.13073069390583210
       14       39.13128545877647470
```

```
   15        39.13167344420620710
   16        39.13176966408823890
   17        39.13185565593330750
   18        39.13184272127839590
   19        39.13184992452583800
   20        39.13186853015521650
   21        39.13188528106214650
   22        39.13189500289722390
   23        39.13189827734217370
   24        39.13189767418734280
   25        39.13189681440015020
   26        39.13189744447644360
   27        39.13189825111452790
   28        39.13189861296724810
   29        39.13189836184337620
   30        39.13189772912585340
   31        39.13189724189699350
   32        39.13189716610731490
   33        39.13189721131325880
   34        39.13189724718414910
   35        39.13189726079488650
   36        39.13189725911997390
   37        39.13189725084322390
   38        39.13189724945078750
   39        39.13189725210217060
   40        39.13189725437293730
   41        39.13189725572802760
   42        39.13189725634585870
   43        39.13189725657433370
   44        39.13189725673281320
   45        39.13189725692146940
   46        39.13189725706185130
   47        39.13189725715076860
   48        39.13189725720364010
   49        39.13189725723376710
   50        39.13189725724888750
   51        39.13189725725868580
   52        39.13189725726581970
   53        39.13189725727029610
   54        39.13189725727171720
   55        39.13189725726978450
   56        39.13189725726073930
   57        39.13189725724281940
   58        39.13189725721896650
   59        39.13189725720405930
   60        39.13189725719252720
   61        39.13189725711620070
   62        39.13189725702143560
   63        39.13189725696842200
   64        39.13189725694172690
   65        39.13189725680675220
   66        39.13189725668458150
   67        39.13189725657132100
   68        39.13189725653533200
   69        39.13189725646096660
   70        39.13189725645969470
   71        39.13189725645708710
   72        39.13189725645565180
   73        39.13189725645477070
   74        39.13189725645469960
 LAGRANGE MULTIPLIKATOREN
 -0.1981422D+01   0.3962843D+01
LOESUNG
  0.4652087D-03   0.2443828D+00   0.4519327D+00   0.5905707D+00
```

```
         0.6393062D+00   0.5905641D+00   0.4519210D+00   0.2443663D+00
         0.4511083D-03   0.2443675D+00   0.4519234D+00   0.5905664D+00
         0.6393095D+00   0.5905727D+00   0.4519353D+00   0.2443816D+00
         0.4696462D-03   0.0000000D+00   0.0000000D+00  -0.2008441D+00
       FUNKTION
         0.3913190D+02
       NEBENBEDINGUNGEN
         0.1731835D-15   0.5000000D-01   0.6231196D-01   0.7370969D-01
         0.8341147D-01   0.9088630D-01   0.9593985D-01   0.9874981D-01
         0.9984031D-01   0.1000000D+00   0.9984033D-01   0.9874986D-01
         0.9593992D-01   0.9088636D-01   0.8341154D-01   0.7370974D-01
         0.6231198D-01   0.5000000D-01   0.5000000D-01   0.3768804D-01
         0.2629031D-01   0.1658853D-01   0.9113703D-02   0.4060148D-02
         0.1250192D-02   0.1596931D-03   0.5224987D-14   0.1596660D-03
         0.1250143D-02   0.4060084D-02   0.9113635D-02   0.1658846D-01
         0.2629026D-01   0.3768802D-01   0.5000000D-01
         95 FUNKTIONS AUFRUFE
         76 GRADIENTEN AUFRUFE
       NO FURTHER REDUCTION IN FUNCTION OR CONSTRAINTS
MIN =       39.13189725645469960
```

Das Knickstab-Problem für $\alpha = 105.190$

Die entsprechenden Routinen von BDSCO lauten:

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X(15),Y(15,15),XS(5)
      INTEGER US(16,5)
      EXTERNAL F,R,RKF7
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C DATENDEFINITION------------------------------------------
      N=8
      M=6
      MS=2
      TOL=1E-8
      ITMAX=75
      KP=0
      KS=1
      ISCHALT=1
      NDISKRET=40

C AEQUIDISTANTE MEHRZIELKNOTEN-----------------------------
      DO 100 I=1,M
        X(I)=0.0+1.0/(M-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER TRAJEKTORIEN-------------------------
      DO 200 I=1,M
       DO 200 J=1,N
        Y(J,I) = 0.05D0
200   CONTINUE

C EXAKTE RANDWERTE-----------------------------------------
      Y(1,1) = 0.0
      Y(1,M) = 0.0
      Y(3,1) = 0.0
      Y(4,1) = 0.0
      Y(6,1) = 0.0
      Y(6,M) = 0.0

C STARTSCHAETZUNG DER SCHALTPUNKTE------------------------
```

```
      XS(1)=0.3D0
      XS(2)=0.7D0

      ALPHA = 105.190D0

      CALL BDSCO(F,R,RKF7,X,XS,Y,N,M,MS,KS,TOL,ITMAX,KP)
      CALL DATEN(X,Y,XS,N,M,MS,NDISKRET,TOL,ISCHALT)
      WRITE(*,*) 'MIN = ',0.5D0*Y(3,M)+ALPHA*Y(4,M)
      END



      SUBROUTINE F(X,Y,DY)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 X,Y(15),DY(15),U
      INTEGER US(15,5)
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C DEFINITION DER STEUERUNG---------------------------------
        IF (US(J,1).EQ.-1) U = -Y(6)
        IF ((US(J,1).EQ.1).AND.(US(J,2).EQ.-1)) U = 0.0D0
        IF (US(J,2).EQ.1) U = -Y(6)

C DEFINITION DES DIFFERENTIALGLEICHUNGSSYSTEMS-------------
        DY(1) = DSIN(Y(2))
        DY(2) = U
        DY(3) = U*U
        DY(4) = DCOS(Y(2))
        DY(5) = 0.0D0
        DY(6) = -Y(5)*DCOS(Y(2))+ALPHA*DSIN(Y(2))
        DY(7) = 0.0D0
        DY(8) = 0.0D0
        IF ((US(J,1).EQ.1).AND.(US(J,2).EQ.-1)) THEN
          DY(1) = 0.0D0
          DY(2) = 0.0D0
          DY(3) = 0.0D0
          DY(4) = 1.0D0
          DY(5) = 0.0D0
          DY(6) = 0.0D0
        ENDIF

      RETURN
      END



      SUBROUTINE R(YA,YB,ZZ,W,NYA,NSK)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL*8 YA(15),YB(15),ZZ(15),W(15),U
      INTEGER US(15,5),NYA(15),NSK(15)
      COMMON /CI/ J,L,LS,US
      COMMON /WERT/ALPHA

C ANFANGSBEDINGUNGEN---------------------------------------
        W(1) = YA(1)
        W(2) = YA(3)
        W(3) = YA(4)
        W(4) = YA(6)

        NYA(1) = 1
        NYA(3) = 2
        NYA(4) = 3
```

```
      NYA(6) = 4

C ENDBEDINGUNGEN------------------------------------------
      W(5) = YB(1)
      W(6) = YB(6)

C SCHALTBEDINGUNGEN---------------------------------------
      W(7) = ZZ(1)-0.05D0
      W(8) = ZZ(2)
      W(9) = ZZ(6)
      W(10)= ZZ(6)

      NSK(7) = 1
      NSK(8) = 1
      NSK(9) = 1
      NSK(10)= 2

C SPRUNGBEDINGUNGEN---------------------------------------
      IF (LS.EQ.1) THEN
        ZZ(5)=ZZ(5)-ZZ(7)
      ENDIF
      IF (LS.EQ.1) THEN
        ZZ(6)=ZZ(6)-ZZ(8)
      ENDIF

     RETURN
     END
```

Mit der Ausgabe:

```
   N= 8      M= 6      MS= 2
   PRESCRIBED RELATIVE PRECISION  0.10E-07
   MAXIMUM PERMITTED NUMBER OF ITERATIONS 75

********************************************************************************
********************************************************************************
IT   ABS.ERR.  LEVEL1    LEVEL2    LEVEL3    RELAX. NEW RANK  COND(M)   NORM(M)
 0  0.10E+01  0.50E+03  0.61E+06  0.31E+07          0    60  0.59E+04  0.99E+00
    0.99E+00  0.49E+03  0.60E+06  0.30E+07  0.010
 1  0.99E+00  0.49E+03  0.60E+06  0.30E+07          1    60  0.60E+04  0.10E+01
    0.37E+01  0.17E+04  0.14E+05  0.50E+05  1.000
 2  0.37E+01  0.17E+04  0.25E+04  0.34E+04          0    60  0.13E+05  0.10E+01
    0.69E-01  0.31E+02  0.19E+04  0.66E+04  1.000
 3  0.69E-01  0.26E+02  0.26E+02  0.14E+01          0    60  0.61E+04  0.50E+02
    0.57E-02  0.25E+01  0.25E+01  0.74E-01  1.000
 4  0.57E-02  0.25E+01  0.25E+01  0.53E-01          1    60  0.56E+04  0.50E+02
    0.46E-03  0.19E+00  0.19E+00  0.80E-03  1.000
 5  0.46E-03  0.11E-02  0.97E-03  0.61E-03          2    60  0.52E+04  0.50E+02
    0.30E-06  0.17E-05  0.16E-05  0.19E-05  1.000
 6  0.30E-06  0.16E-05  0.15E-05  0.17E-05          3    60  0.53E+04  0.50E+02
    0.36E-07  0.42E-07  0.28E-07  0.87E-07  1.000
 7  0.36E-07  0.42E-07  0.27E-07  0.79E-07          4    60  0.52E+04  0.50E+02
    0.10E-10  0.96E-10  0.10E-09  0.50E-10  1.000
 8  0.10E-10  0.96E-10  0.10E-09  0.54E-10          5    60  0.51E+04  0.50E+02
    0.87E-12  0.12E-11  0.89E-12  0.20E-11  1.000
 9  0.87E-12  0.12E-11  0.88E-12  0.18E-11          6    60  0.51E+04  0.50E+02
    0.34E-15  0.52E-15  0.40E-15  0.51E-15  1.000
10  0.34E-15  0.52E-15  0.40E-15  0.53E-15          7    60  0.51E+04  0.50E+02
********************************************************************************
********************************************************************************

   SOLUTION OBTAINED AFTER 11    ITERATION STEPS
```

```
   SWITCHING POINTS
   0.3047235E+00 0.6952765E+00

MIN =     104.32039113136208600
```

Die entsprechenden Routinen von NUDOCCCS lauten:

```
      PROGRAM NUDDELS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(
     A   NDGL     = 4,
     B   NSTEUER  = 1,
     C   NDISKRET = 17,
     D   NUNBE    = 1,
     E   NSTUETZ  = 1,
     F   NNEBEN   = 2,
     G   NUGLNB   = 2,
     H   NRAND    = 1,
     I   NART     = 4,
     J   IPRINT   = 1000,
     K   IMAX     = 500,
     L   SCALE    = 0.25D0,
     M   R1       = 1.0D0,
     N   DEL      = -1.0D-6,
     O   EP       = 1.D-8,
     P   NZUSATZ  = NUNBE*NSTUETZ,
     Q   N        = (NDISKRET+2)*NSTEUER+NZUSATZ,
     R   M        = NDISKRET*NNEBEN+NRAND+NZUSATZ-NUNBE,
     S   ME       = M-NDISKRET*NUGLNB,
     T   MAXNM    = M+(2*(N/M)/(N/M+1))*(N-M))
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET+2),F(N),
     1   G(M),UH(M),IVSET(M),EPS(M+1),T(NDISKRET),UNBE(NUNBE,NSTUETZ),
     2   UHELP(N),C(MAXNM),P(N),OLDF(N),DIAG(N),BETA(N),ATG(N),
     3   Y(MAXNM),Z(MAXNM),G1(M),G2(M),Z1(N),H(NDISKRET),
     4   MSDGL(NUNBE),MSSTUETZ(NSTUETZ)
      COMMON/WA/A(105,105)/WB/B1(105,105)/WC/Q(105,105)

C AEQUIDISTANTE DISKRETISIERUNG----------------------------
      DO 100 I=1,NDISKRET
         T(I)=1.0D0/(NDISKRET-1)*(I-1)
100   CONTINUE

C STARTSCHAETZUNG DER STEUERUNG----------------------------
      DO 200 I=1,NSTEUER
       DO 200 J=1,NDISKRET
         U(I,J) = 0.1D0
200   CONTINUE

C STARTSCHAETZUNG DER FREIEN ANFANGSWERTE------------------
      DO 300 I=1,NUNBE
        DO 300 J=1,NSTUETZ
          UNBE(1,J) = 0.0D0
300   CONTINUE

C STARTSCHAETZUNG DER HESSEMATRIX--------------------------
      DO 1100 I=1,N
        B1(I,I)=1.0D0
        DO 1100 J=I+1,N
          B1(J,I)=0.0D0
1100      B1(I,J)=0.0D0

      DO 1300 I=1,M
1300    EPS(I)=EP
```

```
      EPS(M+1)=EP

      CALL NUDOCCCS(NDGL,NSTEUER,NDISKRET,NUNBE,NNEBEN,NUGLNB,NRAND,
     1    NZUSATZ,NART,IPRINT,IMAX,ISTOP,SCALE,R1,DEL,N,M,ME,MAXNM,
     2    X,U,F,FF,G,UH,IV,IVSET,EPS,T,H,UNBE,UHELP,Y,Z,C,P,OLDF,
     3    BETA,ATG,Z1,G1,G2,DIAG,NSTUETZ,MSDGL,MSSTUETZ)
      CALL AUSGABE(FF,X,UHELP,T,G,NDGL,NSTEUER,NDISKRET,NNEBEN,NRAND,
     1              N,M)
      WRITE(*,*) 'MIN = ',FF
      STOP
      END



      SUBROUTINE MINFKT(X,U,T,MIN,NDGL,NSTEUER,NDISKRET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION MIN
      DIMENSION U(NSTEUER,NDISKRET),X(NDGL,NDISKRET),T(NDISKRET)

        MIN = 0.5D0*X(3,NDISKRET)+105.190D0*X(4,NDISKRET)

      RETURN
      END



      SUBROUTINE DGLSYS(X,U,T,DX,NDGL,NSTEUER)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),U(NSTEUER),DX(NDGL)

        DX(1) = DSIN(X(2))
        DX(2) = U(1)
        DX(3) = U(1)*U(1)
        DX(4) = DCOS(X(2))

      RETURN
      END



      SUBROUTINE ANFANGSW(AWX,UNKNOWN,NDGL,NUNBE)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION AWX(NDGL),UNKNOWN(NUNBE)

        AWX(1) = 0.0D0
        AWX(2) = UNKNOWN(1)
        AWX(3) = 0.0D0
        AWX(4) = 0.0D0

      RETURN
      END



      SUBROUTINE RANDBED(X,U,T,R,NDGL,NSTEUER,NDISKRET,NRAND)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL,NDISKRET),U(NSTEUER,NDISKRET),R(NRAND),
     F          T(NDISKRET)

        R(1) = X(1,NDISKRET)

       RETURN
       END
```

```
      SUBROUTINE NEBENBED(X,U,T,CON,NDGL,NSTEUER,NNEBEN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(NDGL),CON(NNEBEN),U(NSTEUER)

        CON(1) = -X(1)+0.05D0
        CON(2) =  X(1)+0.05D0

      RETURN
      END
```

Mit der Ausgabe:

```
   1     105.03781575500963200
   2     104.92294253186298600
   3     104.67432365606131800
   4     104.35601298671367900
   5     104.41297397934553700
   6     104.45747917197796500
   7     104.48639791972688300
   8     104.49830559912985000
   9     104.50127191453482600
  10     104.49867662593925400
  11     104.49246569497096000
  12     104.48408295434116200
  13     104.47475684553722900
  14     104.45564383252279600
  15     104.44006418423873600
  16     104.43035225015650300
  17     104.42533083559149500
  18     104.42278101045859300
  19     104.42149419949757300
  20     104.42084658713062600
  21     104.42052109543573100
  22     104.41953388478894700
  23     104.41908417931934100
  24     104.41885995162898800
  25     104.41874797735644600
  26     104.41031161997190700
  27     104.37795351400579600
  28     104.36276532461360500
  29     104.35541955304619000
  30     104.35181767409933700
  31     104.35004023492403700
  32     104.34916032126311300
  33     104.34872398011010800
  34     104.34850741794656400
  35     104.34839993652192900
  36     104.34834668110991600
  37     104.34832045964414200
  38     104.34830780145709600
  39     104.34830226932423600
  40     104.34829685155070000
  41     104.34811052675418400
  42     104.34802332665999800
  43     104.34798074530664300
  44     104.34795970404313400
  45     104.34794912006313400
  46     104.34794364071368000
  47     104.34794074419738800
  48     104.34786218656019200
  49     104.33785882878497400
```

50	104.33356385517097900
51	104.33151694926182500
52	104.33052212250395000
53	104.33002300154998900
54	104.32977303528673500
55	104.32964795036011200
56	104.32958538220685800
57	104.32955409157952400
58	104.32953844456298500
59	104.32953062059696900
60	104.32952670848284500
61	104.32952475238424500
62	104.32952377432113200
63	104.32388153254170500
64	104.32111529340768600
65	104.31938681375594100
66	104.31844727399938200
67	104.31802345056998100
68	104.31782339892807000
69	104.31772626672068300
70	104.31767840883902900
71	104.31765465676454600
72	104.31764282621598700
73	104.31763692320610900
74	104.31763397500381500
75	104.31763250230190500
76	104.31763176621674900
77	104.31763139822712100
78	104.31763121474764900
79	104.31763112292944100
80	104.31763107673371800
81	104.31763105386427500
82	104.31757877276713500
83	104.31754730019456200
84	104.31745962922443000
85	104.31737911119208200
86	104.31736841132544700
87	104.31736581348856400
88	104.31735274863851300
89	104.31734749135522100
90	104.31734601410246900
91	104.31734583819060400
92	104.31734560403268100
93	104.31734509526160100
94	104.31734414564829200
95	104.31734393601300300
96	104.31734389985840000
97	104.31734386187274300
98	104.31734375817973200
99	104.31734341895659700
100	104.31734316834020100
101	104.31734314742280400
102	104.31734312432939800
103	104.31734304495258000
104	104.31734292347400100
105	104.31734288289217700
106	104.31734287908068400
107	104.31734287748953700
108	104.31734287232697500
109	104.31734286773614400
110	104.31734286732857700
111	104.31734286731925500
112	104.31734286722051800
113	104.31734286720839600

```
    114      104.31734286720561000
    115      104.31734286720549700
    116      104.31734286720521300
    117      104.31734286720471500
    118      104.31734286720410400
    119      104.31734286720387700
    120      104.31734286720386300
    121      104.31734286720386300
    122      104.31734286720384800
    123      104.31734286720384800
    124      104.31734286720384800
    125      104.31734286720383400
    126      104.31734286720382000
   LAGRANGE MULTIPLIKATOREN
   -0.8846393D+01   0.1582025D+01   0.7264368D+01   0.7264369D+01
    0.1582024D+01
  LOESUNG
    0.5956014D-02   0.1103438D+01   0.1769724D+01   0.1722334D+01
    0.9835439D+00   0.4614174D-01  -0.3682891D-01  -0.1018563D+00
   -0.1273739D+00  -0.1018557D+00  -0.3682896D-01   0.4614160D-01
    0.9835436D+00   0.1722335D+01   0.1769722D+01   0.1103439D+01
    0.5955085D-02   0.0000000D+00   0.0000000D+00  -0.3391117D+00
  FUNKTION
    0.1043173D+03
  NEBENBEDINGUNGEN
   -0.1734723D-16   0.5000000D-01   0.7010327D-01   0.8634771D-01
    0.9629070D-01   0.1000000D+00   0.1000000D+00   0.9926352D-01
    0.9865921D-01   0.9842705D-01   0.9865921D-01   0.9926352D-01
    0.1000000D+00   0.1000000D+00   0.9629070D-01   0.8634771D-01
    0.7010327D-01   0.5000000D-01   0.5000000D-01   0.2989673D-01
    0.1365229D-01   0.3709303D-02  -0.1387779D-16  -0.3469447D-16
    0.7364811D-03   0.1340788D-02   0.1572951D-02   0.1340787D-02
    0.7364805D-03  -0.4163336D-16  -0.2775558D-16   0.3709303D-02
    0.1365229D-01   0.2989673D-01   0.5000000D-01
   154 FUNKTIONS AUFRUFE
   128 GRADIENTEN AUFRUFE
  NO FURTHER REDUCTION IN FUNCTION OR CONSTRAINTS
min =     104.31734286720382000
```

Literaturverzeichnis

[Betts 1990] Betts, J.T., Huffmann, W.P. *Path Constrained Trajektory Optimization Using Sparse Sequential Quadratic Programming* Applied Mathematics and Statistics Group, Boeing Computer Services, Seattle, Washington, 1991

[Biggs 1975] Biggs, M.C. *User's Guide to OPRQP* Numerical Optimization Centre, Optima Manual, The Hatfield Polytechnic, 1975

[Bock 1984] Bock, H.G., Pitt, K.J. *A multiple Shooting Algorithm for Direkt Solution of Optimal Control Problems* IFAC 9th World Congress, Budapest, Hungary, 1984

[Bock 1987] Bock, H.G. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen* Bonner Mathematische Schriften, 1987

[Breakwell 1963] Breakwell, J.V., Speyer, J.L., Bryson, A.E. *Optimization and Control of Nonlinear Systems using then Second Variation* SIAM Series A: Control, Volume 1, 1962-1963

[Bryson 1963] Bryson, A.E., Denham, W.F., Dreyfus, S.E. *Optimal Programming Problems with Inequality Constraints* AIAA Journal, Vol. 1, No. 11 (1963)

[Bryson 1975] Bryson, A.E., Yu-Chi-Ho, Jr. *Applied Optimal Control Optimization, Estimation, and Control* a Holsted Press Book, 1975

[Conte 1980] Conte, S.D., de Boor, C. *Elementary Numerical Analysis* McGraw-Hill, New York, 1980

[Dueck 1993] Dueck, G., Scheuer, T., Wallmeier, H.M. *Toleranzschwelle und Sintflut: neue Ideen zur Optimierung* Spektrum der Wissenschaft, Nr.3, 1993

[Feichtinger 1986] Feichtinger, G., Hartl, R.F. *Optimale Kontrolle ökonomischer Prozesse* de Gruyter, 1986

[Forster 1987] Forster, O. *Analysis 2* Vieweg Braunschweig, 1987

[Gellert 1984] Gellert, W. *Kleine Enzyklopädie Mathematik* Harri Deutsch Verlag Thun und Frankfurt/M., 1984

[Goh 1987] Goh, C.J., Teo, K.L. *MISER: An Optimal Control Software* Applied Research Corporation, National University of Singapore, Kent Ridge, Singapore (1987)

[Heuser 1989] Heuser, H. *Gewöhnliche Differentialgleichungen* B. G. Teubner Stuttgart, 1989

[Holland 1992] Holland, J.H. *Genetische Algorithmen* Spektrum der Wissenschaft, Nr. 9, 1992

[Horn 1989] Horn, M.K. *Solution of the Optimal Control Problem Using the Software Package STOMP* IFAC Workshop on Control Applications of Nonlinear Programming and Optimization, Paris 1989

[Jacobson 1971] Jacobson, D.H., Lele, M.M., Speyer, J.L. *New Necessary Conditions of Optimality for Control Problems with State Variable Inequality Constraints* Journal of Mathematical Analysis and Applications, Bd. 35, Academic Press, 1971

[Knobloch 1974] Knobloch, H.W., Kappel, F. *Gewöhnliche Differentialgleichungen.* B.G. Teubner Stuttgart, 1974

[Krämer-Eis 1985] *Ein Mehrzielverfahren zur numerischen Berechnung optimaler Feedback-Steuerungen bei beschränkten nichtlinearen Steuerungsproblemen* Bonner Mathematische Schriften, 1985

[Kraft 1980] Kraft, D. *FORTRAN-Programme zur numerischen Lösung optimaler Steuerprobleme* Mitteilung, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, 1980

[Kraft 1985] Kraft, D. *Optimalsteuerungen - ein systematisches Hilfsmittel zur rechnergestützten Erforschung der dynamischen Möglichkeiten eines Tieftemperaturwindkanals* Forschungsbericht, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, 1985

[Maurer 1976] Maurer, H. *Numerical Solution of Singular Control Problems Using Multiple Shooting Techniques* Journal of Optimization Theory and Application, Vol. 18, 1976

[Maurer 1977] Maurer, H. *On Optimal Control Problems with bounded State Variables and Control appearing linearly* SIAM Journal on Control and Optimization, Vol. 15, No. 3, 1976

[Maurer 1979] Maurer, H. *On the Minimum Principle for Optimal Control Problems with State Constraints* Schriftenreihe des Rechenzentrums der Universität Münster, Nr. 41, 1979

[Maurer 1989] Maurer, H. *Optimierung Dynamischer Systeme* Skript, Universität Münster, 1990

[Maurer 1991] Maurer, H., Mittelmann H.D. *The Non-Linear Beam via Optimal Control with Bounded State Variables* Optimal Control Applications and Methodes, Vol. 12, 1991

[Oberle 1982] Oberle, H.J. *BOUNDSCO - Hinweise zur Benutzung des Mehrziel-Verfahrens für die numerische Lösung von Randwertproblemen mit Schaltbedingungen.* Math. Institut der TU München, 1982.

[Oberle 1985] Oberle, H.J., Grimm, W., Berger, E. *BNDSCO, Rechenprogramm zur Lösung beschränkter optimaler Steuerungsprobleme.* Preprint TUM-M8509, Math. Institut der TU München, 1985.

[Pesch 1989] Pesch, H.J. *Real-Time Computation of Feedback Controls for Constrained Optimal Control, Part2: A Correction Method Based on Multiple Shooting.* Optimal Control Applications and Methodes, No. 2, 1989

[Pontrjagin 1967] Pontrjagin, L.S., Boltjanskij, V.G., Gamkrelidze, R.V., Miscenko, E.F. *Mathematische Theorie optimaler Prozesse.* R.Oldenbourg München Wien, 1967.

[Scharmack 1967] Scharmack, D.K. *An Initial Value Method for Trajectory Optimization Problems* Advances in Control Systems, Theory and Applications, Volume 5, Academic Press New York London, 1967

[Schittkowski 1980] Schittkowski, K. *Nonlinear Programming Codes* Lecture Notes in Economics and Mathematical System, Bd. 183, Springer Berlin Heidelberg, 1980

[Schittkowski 1981] Schittkowski, K. *Organization, Test and Performance of Optimization Programs* Lecture Notes in Control and Information Science, Bd. 30, Springer Berlin Heidelberg, 1981

[Schwarz 1988] Schwarz, H.R. *Numerische Mathematik* B. G. Teubner Stuttgart, 1988

[Sethi 1981] Sethi, S.P., Thompson, G.L. *Optimal Control Theory* Nijhoff, 1981

[Stoer 1978] Stoer, J., Bulirsch, R. *Einführung in die Numerische Mathematik II* Springer-Verlag, Berlin Heidelberg, 1978.

[Stoer 1983] Stoer, J. *Einführung in die Numerische Mathematik I* Springer-Verlag, Berlin Heidelberg, 1983.

[Stryk 1991] Stryk, von O., Bulirsch, R. *Direct and Indirect Methods for Trajectory Optimization* Schwerpunktprogramm der Deutschen Forschungsgemeinschaft, 'Anwenderbezogene Optimierung und Steuerung', TU München, Sonderforschungsbereich 255, Report No. 271, 1991

[Stryk 1991] Stryk, von O., Bulirsch, R. *Numerical Solution of Optimal Control Problems by Direct Collocation* Schwerpunktprogramm der Deutschen Forschungsgemeinschaft, 'Anwenderbezogene Optimierung und Steuerung', TU München, Sonderforschungsbereich 255, Report No. 322, 1991

[Walter 1990] Walter W. *Gewöhnliche Differentialgleichungen* Springer-Verlag Berlin Heidelberg, 1990

[Ziegler 1992] Ziegler, R.A. *Analyse des Verfahrens MISER zur Parametrisierung optimaler Steuerungen* Diplomarbeit Technische Universität München, 1992

Herrn Prof. Dr. Maurer danke ich für die interessante Themenstellung sowie für wertvolle Diskussionen. Desweiteren gilt Herrn Laurenz Göllmann und Herrn Michael Heukamp mein Dank für die mühevolle Arbeit des Korrekturlesens.

Ich versichere, daß ich die hier vorliegende Arbeit selbständig angefertigt habe, und keine weiteren Hilfsmittel als die im Literaturverzeichnis aufgeführten Quellen verwendet habe.

Rheine, den 31. März 1993