Ruprecht-Karls-Universität Heidelberg

Fakultät für Mathematik und Informatik

Effiziente Ableitungserzeugung in einem adaptiven BDF-Verfahren

Diplomarbeit

Betreuer: Prof. Dr. Dr. h.c. H. G. Bock

vorgelegt von Jan Albersmeyer

Heidelberg, April 2005

Vorwort

An dieser Stelle möchte ich einigen Personen danken, die mich bei der Fertigstellung dieser Arbeit unterstützt, bzw. diese erst ermöglicht haben.
Zu allererst möchte ich Prof. Dr. Dr. h.c. Hans Georg Bock und Dr. Johannes Schlöder danken für die anregenden Vorlesungen, Seminare und Gespräche in den letzten Jahren, sowie für die Möglichkeit zur Arbeit an interessanten Themen in einem sehr angenehmen Arbeitsklima. Weiterhin danke ich Dr. Ulrich Brandt-Pollmann für seine stets freundliche Betreuung und seine hilfreichen Anmerkungen bei der Entstehung dieser Arbeit. Zu Dank verpflichtet bin ich außerdem Carmen Ellsässer für das wiederholte Korrekturlesen der Zwischenfassungen.
Ein ganz besonderer Dank geht an meine Eltern, die mir stets in meinen Plänen beigestanden und mich gefördert haben und mir das Studium hier in Heidelberg ermöglichten. Abschließend möchte ich mich ganz herzlich bei Carole bedanken für ihre große Hilfe und Motivation besonders in der letzten hektischen Phase kurz vor der Abgabe dieser Arbeit.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die numerische Simulation und die systematische Optimierung von komplexen realen Prozessen hat gerade in der letzten Zeit sowohl im wissenschaftlichen wie auch im industriellen Bereich immer mehr an Bedeutung gewonnen. Dabei verstehen wir unter einem Prozeß einen zeitabhängigen Vorgang, der sich durch ein System von Differential- bzw. Differential-algebraischen Gleichungen beschreiben läßt. Hierbei ermöglicht die numerische Simulation die Gewinnung von quantitativen und qualitativen Ergebnissen sowie von neuen Erkenntnissen über komplexe Systeme und Mechanismen, beispielsweise in der Systembiologie, die sich sonst einer genaueren Untersuchung und einem detaillierteren Verständnis entziehen würden. Ferner bietet die Simulation die Möglichkeit der Beurteilung von Entwürfen und Modellen, die bisher in der Realität nicht, oder nur in kleinerem Maßstab existieren.
Die Wichtigkeit, die eine effiziente, systematische und auf Ableitungsinformationen basierende Optimierung bzw. optimale Steuerung von Abläufen und Prozessen für die Wirtschaft und Industrie besitzt, kann in einer Zeit des durch die Globalisierung und die Offenheit der Märkte stetig steigenden Konkurrenzdrucks kaum überschätzt werden.
Die Modellierung von chemischen, verfahrenstechnischen und auch biotechnologischen Prozessen führt dabei oft auf nichtlineare und steife Systeme von differential-algebraischen Gleichungen (DAEs), die durchaus Tausende von Gleichungen umfassen können. Obwohl wir in dieser Arbeit vor allem Beispiele aus diesen Bereichen diskutieren, lassen sich die im Folgenden vorgestellten Strategien auch effizient auf andere Probleme, beispielsweise aus

der Wirtschaft, anwenden, solange sie sich als Anfangswertprobleme für steife gewöhnliche Differentialgleichungen oder differential-algebraische Gleichungen formulieren lassen.
Zur Simulation dieser Probleme werden sehr zuverlässige und effiziente Anfangswertproblemlöser benötigt. Für die systematische, ableitungsbasierte Optimierung der Prozesse, die durch diese Modelle beschrieben werden, werden zudem sehr genaue Ableitungen der Lösungen der DAEs nach Anfangswerten, Parametern und auch Kontrollen benötigt, die sogenannten Sensitivitäten.
Bei der Lösung steifer DAEs haben sich im allgemeinen aufgrund ihrer guten Stabilitätseigenschaften die sogenannten BDF-Verfahren bewährt, die in den fünfziger Jahren erstmals von Curtiss und Hirschfelder [CH52] für gewöhnliche Differentialgleichungen (ODEs) vorgestellt, und von Gear später auf DAEs angewendet wurden [Gea71].

Der im Zuge dieser Diplomarbeit implementierte Integrator DAESOL-II stellt eine zuverlässige, effiziente und einfach zu handhabende Möglichkeit zur Lösung eines großen Teils der oben dargestellten Probleme dar. DAESOL-II löst Anfangwertprobleme (IVPs) für linear implizite DAEs vom Index 1. Es beruht auf einem BDF-Verfahren variabler Schrittweite und Ordnung und nutzt zur effizienten Generierung von Sensitivitäten das Prinzip der Internen Numerischen Differentiation (IND), welches von Bock [Boc81] entwickelt wurde. Wir verwenden außerdem einige Ergebnisse aus den Arbeiten von Bleser [Ble86], Eich [Eic87] und Bauer [Bau99]. Diese entstanden über die letzten Jahre hinweg in der Arbeitsgruppe von Bock und manifestierten sich bisher in dem in einigen Varianten vorliegenden Vorgänger-Code DAESOL. Dieser stabile und zuverlässige in Fortran-77 geschriebene Integrator wird seit Jahren mit Erfolg zur effizienten Lösung von Anfangswertproblemen und zur Berechnung von Sensitivitäten im Optimierungskontext eingesetzt. Er besitzt aber auch eine Reihe von, zum Teil durch die Implementierung bedingten, Nachteilen und Schwächen wie dem Fehlen einer dynamischen Speicherverwaltung oder eines präzise definierten, erweiterbaren modularen Aufbaus. Dies macht ihn nur schwer erweiterbar und erfordert dazu noch Expertenwissen über den gesamten Integrator. Letzteres wird mit der zunehmenden Komplexität moderner Integratoren, sowohl hinsichtlich der zugrun-

deliegenden Theorie, wie auch des Programmcodes an sich, immer schwerer vermittelbar.
Hier setzt das von Grund auf neu im ANSI-C Standard implementierte DAESOL-II an. Die Hauptziele bei seiner Entwicklung waren folgende:

- Schnelle, zuverlässige und exakte Lösung von IVPs für steife linear implizite DAEs vom Index 1,
- effiziente Erzeugung von Ableitungen dieser Lösungen nach Anfangswerten, Parametern und Kontrollen und kombinierten Richtungen all dieser,
- einfache Erweiterbarkeit ohne Expertenwissen über den gesamten Integrator durch einen modularen Aufbau mit präzise definierten Schnittstellen,
- klare und benutzerfreundliche Bedienung für den Anwender bei umfangreichen Konfigurationsmöglichkeiten und
- Einsetzbarkeit auf vielen Systemen mit effizienter Ausnutzung der vorhandenen Hardware.

Aufbau der Arbeit

Die vorliegende Diplomarbeit gliedert sich in 7 Kapitel. Die einzelnen Kapitel sind größtenteils unabhängig voneinander, lediglich Kapitel 4 und 5 stützen sich mehr auf die in den vorhergehenden behandelte Materie.

Kapitel 1 enthält die Einleitung und diese Gliederung der Arbeit.

In Kapitel 2 stellen wir die notwendigen theoretischen Grundlagen vor, die zur Lösung von Anfangswertproblemen bei differential-algebraischen Gleichungen benötigt werden und behandeln dabei neben den verschiedenen auftretenden Typen von DAEs auch die Existenz- und Eindeutigkeit der Lösungen, sowie den Begriff des Index.

In Kapitel 3 wird die Klasse der linearen Mehrschrittverfahren zur numerischen Behandlung von Anfangswertproblemen vorgestellt. Wir erläutern Aussagen über Konsistenz, Stabilität und Konvergenz dieser Verfahren zunächst auf äquidistanten Gittern und für konstante Ordnung und transfe-

rieren die Ergebnisse dann im Anschluß soweit möglich auf variable Gitter. Schließlich werden noch die BDF-Verfahren mit ihren grundlegenden Eigenschaften eingeführt und ihre Anwendung auf DAEs erläutert.

Kapitel 4 beschäftigt sich mit der Berechnung von Sensitivitätsinformationen für DAEs. Wir gehen dabei zunächst auf die theoretischen Grundlagen zur Sensitivitätsberechnung ein und motivieren die Variationsdifferentialgleichungen bzw. die Variations-DAEs. Danach werden wir zwei grundsätzliche Ansätze zur numerischen Berechnung der Sensitivitäten vorstellen und ihre Vor- und Nachteile kurz erläutern. Hierbei gehen wir insbesondere auf die Gewinnung von Sensitivitäten mit Hilfe des Prinzips der Internen Numerischen Differentiation ein.

Kapitel 5 widmet sich dem während dieser Diplomarbeit entstandenen Integrator DAESOL-II. Wir präsentieren und entwickeln die speziell in DAESOL-II verwendeten Strategien und gehen auf Fragen der Implementierung ein. Im Zuge dessen beschäftigen wir uns mit der Darstellung der Interpolationspolynome im BDF-Verfahren, der Fehlerschätzung, einer effizienten Schrittweiten- und Ordnungssteuerung, sowie der Lösung der beim impliziten BDF-Verfahren auftretenden nichtlinearen Gleichungssysteme und den Strategien zum Start des Mehrschrittverfahrens. Außerdem betrachten wir die Generierung von konsistenten Anfangswerten für ein DAE-Anfangswertproblem und die im Optimierungskontext wünschenwerte Möglichkeit der relaxierten Formulierung der algebraischen Gleichungen. Ferner beschäftigen wir uns detaillierter mit der konkreten Umsetzung des Prinzips der IND bei der Berechnung der Sensitivitäten. Abschließend diskutieren wir noch einige Möglichkeiten zur Generierung der benötigten Ableitungen der Modellfunktionen und geben einen Überblick über die Schnittstelle, den Aufbau des Programms und die verwendeten Softwarebibliotheken.

Die Fähigkeit zur Lösung komplizierter Probleme und die Effizienz des von uns implementierten Integrators demonstrieren wir in Kapitel 6. Dazu motivieren und lösen wir eine Reihe von Anfangswertproblemen für ODEs und DAEs. Neben einigen einfachen Testbeispielen und anspruchsvolleren Problemen aus Problemsammlungen behandeln wir auch Beispiele aus der Anwendung in der Verfahrenstechnik und der Biochemie, eine Destillationskolonne und die Peroxidase-Oxidase-Reaktion.

In Kapitel 7 wird abschließend das Wesentliche der Arbeit nocheinmal zusammengefaßt und es werden in einem Ausblick einige weitergehende Aspekte und Möglichkeiten der Erweiterung von DAESOL-II andiskutiert.

Kapitel 2

Theoretische Grundlagen zur Lösung differential-algebraischer Gleichungen

In diesem Kapitel werden die notwendigen theoretischen Grundlagen für die Lösung von differential-algebraischen Gleichungen (DAEs) dargestellt. Zuerst wird formal ein DAE-System eingeführt, danach werden grundlegende Eigenschaften (Index, Existenz und Eindeutigkeit von Lösungen, etc.) erläutert. Desweiteren werden grundlegende Begriffe definiert und einige Spezialfälle von DAEs vorgestellt.

2.1 Verschiedene Typen von DAEs

Definition 2.1.1 (Voll-implizite DAE)
Eine *voll-implizite* DAE ist ein System von Gleichungen

$$F(t, y(t), \dot{y}(t)) = 0, \tag{2.1}$$

wobei $F : \mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$ und $y : \mathbb{R} \to \mathbb{R}^{n_y}$ vektorwertige Funktionen sind und t eine unabhängige Variable beschreibt. $\dot{y}$ entspricht der Ableitung nach t. (Diese Festlegung wird im Folgenden stets beibehalten.)

Oft treten in der Modellierung von realen Prozessen folgende Spezialfälle auf:

Definition 2.1.2 (Semi-explizite DAEs)
Eine differential-algebraische Gleichung heißt *semi-explizit*, wenn sie wie folgt in einen differentiellen und einen algebraischen Teil aufgespaltet werden kann:

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), z(t)) \in \mathbb{R}^{n_x} \\ 0 &= g(t, x(t), z(t)) \in \mathbb{R}^{n_z}, \end{aligned} \tag{2.2}$$

wobei $x(t) \in \mathbb{R}^{n_x}$ die differentiellen und $z(t) \in \mathbb{R}^{n_z}$ die algebraischen Variablen sind. Entsprechend heißen die oberen Gleichungen *differentielle Gleichungen* und die unteren *algebraische Gleichungen* der DAE.

Definition 2.1.3 (Linear implizite DAEs)
Eine DAE heißt *linear implizit*, wenn sie linear in den Ableitungen von x nach t ist

$$\begin{aligned} A(t, x(t), z(t))\, \dot{x}(t) &= f(t, x(t), z(t)) \in \mathbb{R}^{n_x} \\ 0 &= g(t, x(t), z(t)) \in \mathbb{R}^{n_z}, \end{aligned} \tag{2.3}$$

wobei $A(t, x(t), z(t)) \in \mathbb{R}^{n_x \times n_x}$, $x : \mathbb{R} \to \mathbb{R}^{n_x}$ und $z : \mathbb{R} \to \mathbb{R}^{n_z}$.

Definition 2.1.4 (Lösung einer DAE)
Eine *(klassische) Lösung* auf einem Intervall $\mathcal{I} \subset \mathbb{R}$ der obigen Klassen von DAEs ist eine stetig differenzierbare vektorwertige Funktion

$$y(t) = \begin{pmatrix} x(t) \\ z(t) \end{pmatrix} : \mathbb{R} \to \mathbb{R}^{n_y} = \mathbb{R}^{n_x} \times \mathbb{R}^{n_z},$$

welche die gegebenen Gleichungen für alle $t \in \mathcal{I}$ erfüllt.

Aus obigen Definitionen wird ersichtlich, dass der Unterschied zwischen gewöhnlichen Differentialgleichungen (ODEs) und DAEs darin liegt, dass die Lösung einer DAE auf der von den algebraischen Gleichungen (vollständig) definierten Mannigfaltigkeit bleibt.
Jede hinreichend glatte DAE kann in eine gewöhnliche Differentialgleichung transformiert werden, indem man den algebraischen Teil des Systems differenziert. In diesem Zusammenhang ist der *differentielle Index*, welcher die algebraischen Gleichungen charakterisiert, eine bedeutende Grösse. Dieser wurde zuerst von Gear [Gea88] eingeführt.

2.2 Differentieller Index einer DAE

Definition 2.2.1 (Differentieller Index einer DAE)
Die implizite DAE

$$F(t, y(t), \dot{y}(t)) = 0$$

ist vom *differentiellen Index* $k \in \mathbb{N}$ (kurz: *Index*), wenn k die kleinstmögliche Zahl ist, so dass $\dot{y}(t)$ vollständig durch die folgenden $(k+1)$ Gleichungen bestimmt ist:

$$\begin{aligned} F(t, y(t), \dot{y}(t)) &= 0 \\ \frac{d}{dt} F(t, y(t), \dot{y}(t)) &= 0 \\ &\vdots \\ \frac{d^k}{dt^k} F(t, y(t), \dot{y}(t)) &= 0. \end{aligned}$$

Beispiel 2.2.2 (Beispiel für ein System vom Index 1)
Betrachtet man die semi-explizite DAE (2.2) und die Ableitung der algebraischen Gleichungen g nach t, so erhält man (ohne das Argument t):

$$g_t + g_x \dot{x} + g_z \dot{z} = 0.$$

Sei g_z regulär, dann erhält man durch Umformen

$$\dot{z} = -g_z^{-1}(g_t + g_x \dot{x})$$

und somit folgendes gekoppelte ODE-System

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), z(t)) \\ \dot{z}(t) &= -g_z^{-1}(t, x(t), z(t)) \left[g_t(t, x(t), z(t)) + g_x(t, x(t), z(t)) \dot{x}(t) \right]. \end{aligned}$$

Nach Definition 2.2.1 ist diese DAE *vom differentiellen Index 1.*
Ist g_z singulär, so iteriert man das Schema und erhält nach einer endlichen Anzahl von Iterationen ein System von gewöhnlichen Differentialgleichungen.

Bemerkung 2.2.3 (DAEs mit Index grösser als 1)
Eine DAE mit Index k grösser als 1 kann durch $(k-1)$-faches Ableiten der algebraischen Gleichungen nach t in eine DAE mit Index 1 umgeformt werden.
Die analytische Lösung eines indexreduzierten Systems erfüllt die algebraischen Gleichungen des ursprünglichen Problems und deren erste $(k-1)$

Ableitungen. Sie ist somit identisch mit der analytischen Lösung des ursprünglichen Problems.
Das System der algebraischen Gleichungen und ihrer ersten $(k-1)$ Ableitungen nennt man die *Invarianten* des indexreduzierten Systems.

Bemerkung 2.2.4 (Numerische Schwierigkeiten)
Die numerische Lösung von DAEs mit Index größer als 1 ist wesentlich komplizierter als es nach den obigen Überlegungen den Anschein hat. In der numerischen Berechnung besteht das Problem, dass man, aufgrund der Diskretisierungs- und Rundungsfehler, die durch die Invarianten definierte Mannigfaltigkeit verlässt. Es muss somit sichergestellt sein, dass die Invarianten im Laufe der Integration noch immer erfüllt sind. Die numerische Behandlung von solchen Gleichungen wird detaillierter in den Werken von Eich [Eic91, Eic93], von Schwerin [vS97], Petzold [PRM97] oder Pantelides [PSV94] beschrieben und wird in dieser Arbeit nicht weiter behandelt.

2.3 Störungsindex einer DAE

Eine weitere, insbesondere für die numerische Behandlung, wichtige Grösse ist der *Störungsindex*. Dieser beschreibt die Sensitivität des Systems in Bezug auf kleine Störungen in der rechten Seite oder den Anfangswerten des Systems und wurde von Hairer et al. [HLR89] eingeführt.

Definition 2.3.1 (Störungsindex einer DAE)
Für eine DAE (2.1) ist der *Störungsindex* entlang der Lösung $y(t), t \in [t_0, t_f]$, definiert als die kleinste ganze Zahl k, so dass für alle Funktionen $\hat{y}(t)$ mit dem Defekt

$$F(t, \hat{y}(t), \dot{\hat{y}}(t)) = \delta(t)$$

die Abschätzung

$$\begin{aligned} \| \hat{y}(t) - y(t) \| \quad \leq \quad & C(F, |t_f - t_0|) \Big(\| \hat{y}(0) - y(0) \| \\ & + \max_{0 \leq \tilde{t} \leq t} \| \int_0^{\tilde{t}} \delta(\tau) d\tau \| \\ & + \max_{0 \leq \tilde{t} \leq t} \| \delta(\tilde{t}) \| + \ldots + \max_{0 \leq \tilde{t} \leq t} \| \delta^{(k-1)}(\tilde{t}) \| \Big) \end{aligned} \tag{2.4}$$

gilt, unter der Voraussetzung, dass $\delta(t)$ klein genug ist.

Beispiel 2.3.2
Der Störungsindex ist 0, wenn folgende Abschätzung gilt

$$\| \hat{y}(t) - y(t) \| \leq C(F, |t_f - t_0|) \left(\| \hat{y}(0) - y(0) \| + \max_{0 \leq \tilde{t} \leq t} \| \int_0^{\tilde{t}} \delta(\tau) d\tau \| \right).$$

Für gewöhnliche Differentialgleichungen mit Lipschitz-stetiger rechter Seite gilt diese Abschätzung immer und somit ist der Störungsindex 0.

Für semi-explizite DAEs stimmt der differentielle Index mit dem Störungsindex überein. Dies gilt im Allgemeinen jedoch nicht, wie folgendes Beispiel belegt [HLR89]:
Man betrachte das System:

$$\begin{aligned} y_1'(t) - y_3(t)y_2(t) + y_2(t)y_3(t) &= 0 \\ y_2(t) &= 0 \\ y_3(t) &= 0. \end{aligned}$$

Dieses Problem ist vom differentiellen Index 0.
Für die Anfangswerte $y(0) = (0,0,0)^T$ erhält man die Lösung $y \equiv 0$. Nimmt man die kleine Störung $\delta(t) = (0, \epsilon \sin(\omega t), \epsilon \cos(\omega t))^T$, so erhält man als Lösung $\hat{y} = (\epsilon^2 \omega t, \epsilon \sin(\omega t), \epsilon \cos(\omega t))^T$. In diesem Fall ist es unmöglich für kleines aber festes ϵ und $\omega \to \infty$ eine Abschätzung der Form (2.4) mit $k = 1$ zu finden, also ohne $\max ||\delta(t)||$ zu berücksichtigen.

Gear bewies folgenden allgemeinen Zusammenhang zwischen differentiellem Index und Störungsindex: (Beweis, siehe [HNW96])

Theorem 2.3.3 (Gear, 1990)
Für die DAE (2.1) gilt

$$pi \leq di + 1$$

falls der differentielle Index di und der Störungsindex pi existieren.

Bemerkung 2.3.4
Die Wichtigkeit des Störungsindex für die numerische Behandlung wird durch folgende Überlegung deutlich: Der Störungsindex zeigt den Einfluss von Rundungsfehlern in F. Hat die DAE Störungsindex k, so steht in (2.4) die $(k-1)$-te Ableitung von δ. Obwohl die Störung δ sehr klein sein kann (z.B. von der

Ordnung der Maschinengenauigkeit aufgrund von Rundungsfehlern), kann ihre Ableitung sehr groß werden. Dies kann zu Problemen in der numerischen Behandlung von DAEs mit Störungsindex größer als 1 führen.
Für die numerische Lösung kann man beweisen, dass die Diskretisierungs- und Rundungsfehler des Integrators sie mit der Ordnung $\mathcal{O}(h^{(1-k)})$ beeinflussen, wobei h die maximale Schrittweite während der Integration ist.

2.4 Existenz und Eindeutigkeit der Lösung

Betrachtet man Systeme von gewöhnlichen Differentialgleichungen, so gibt es einfache Sätze über die Existenz und Eindeutigkeit von Anfangswertproblemen, wie z.B. die bekannten Sätze von *Peano* und *Picard-Lindelöf* (siehe u.a. [Wal93]). Manche der Aussagen aus der Theorie der gewöhnlichen Differentialgleichungen können auf die Theorie der DAEs übertragen werden, indem man DAEs wie oben beschrieben als gewöhnliche Differentialgleichungen betrachtet mit der Bedingung, dass die Lösungen in der von den algebraischen Bedingungen beschriebene Mannigfaltigkeit liegt. Diesen Ansatz verfolgte z.B. Rheinboldt in [Rhe84].
Für die allgemeine Lösbarkeit von DAEs gilt:

Definition 2.4.1 (Lösbarkeit von DAEs)
Sei $\mathcal{I} \subset \mathbb{R}$ offen, Ω eine offene und zusammenhängende Teilmenge aus dem $\mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_y}$ und $F : \Omega \to \mathbb{R}^{n_y}$ differenzierbar. Die DAE (2.1) ist dann in Ω im Intervall $\mathcal{I}$ lösbar, wenn es eine r-dimensionale Familie von Lösungen $Y(t, c)$ gibt, welche auf einer zusammenhängenden offenen Menge $\mathcal{I} \times \tilde{\Omega}$, $\tilde{\Omega} \subset \mathbb{R}^r$, definiert sind, so dass

1. $Y(t, c)$ für alle $t \in \mathcal{I}$ und für alle $c \in \tilde{\Omega}$ definiert ist,
2. $(t, Y(t, c), \dot{Y}(t, c)) \in \Omega$ für $(t, c) \in \mathcal{I} \times \tilde{\Omega}$,
3. falls $\psi(t)$ eine weitere Lösung ist, mit $(t, \psi(t), \dot{\psi}(t)) \in \Omega$, dann gilt $\psi(t) = Y(t, c)$ für ein gewisses $c \in \tilde{\Omega}$,
4. der Graph von Y als Funktion von (t, c) eine $(r + 1)$-dimensionale Mannigfaltigkeit ist.

Dieses Konzept von Lösbarkeit beinhaltet, dass keine Bifurkationen existieren.

Für den Fall einer linear impliziten DAE (2.3) vom Index 1 erhält man folgende Aussagen:

Satz 2.4.2 (Existenz und Eindeutigkeit von DAEs vom Index 1)
Seien $A : \mathbb{R} \times S \longrightarrow \mathbb{R}^{n_x} \times \mathbb{R}^{n_z}$, $f : \mathbb{R} \times S \longrightarrow \mathbb{R}^{n_x}$ und $g : \mathbb{R} \times S \longrightarrow \mathbb{R}^{n_z}$ $\mathcal{C}^r$-Funktionen, $r \geq 2$, $S \subset \mathbb{R}^{n_x+n_z}$ offen und $y = (x^T, z^T)^T \in \mathbb{R}^{n_y}$.
Dann folgt, dass

$$S_0 = \left\{ (t,y) \in \mathbb{R} \times S : \text{Rang} \begin{pmatrix} A(t,y) & 0 & -f(t,y) \\ g_x(t,y) & g_z(t,y) & g_t(t,y) \end{pmatrix} = n_y \right\}$$

eine offene Teilmenge von $\mathbb{R}^{1+n_y}$ ist.

Betrachtet man die Mannigfaltigkeit

$$M(g,S) = \{(t,y) \in \mathbb{R} \times S : g(t,y) = 0\},$$

so gilt, falls $M_0 = M(g,S) \cap S_0 \neq \emptyset$, dass M_0 eine Untermannigfaltigkeit von $M(g,S)$ ist und für alle (t_0, y_0) eine $\mathcal{C}^{r-1}$-Lösung von (2.3) existiert, die durch (t_0, y_0) verläuft und eindeutig ist.

Bemerkung 2.4.3 (Numerische Lösung von DAEs vom Index 1)
Sei

$$S_1 = \left\{ (t,y) \in \mathbb{R} \times S : \text{Rang} \begin{pmatrix} A(t,y) & 0 \\ g_x(t,y) & g_z(t,y) \end{pmatrix} = n \right\}$$

und ferner

$$M_1 = M(g,S) \cap S_1 \neq \emptyset,$$

so muss für die numerische Lösung der DAE (2.3) mit Standardmethoden zusätzlich gelten, dass $(t,y) \in M_1$ für alle Punkte der Lösung.
Dann sind die Matrizen A und g_z regulär für alle $(t,y) \in M_1$, ihre Inversen sind beschränkt und für konsistente Anfangswerte $(t_0, y_0) \in M_1$ hat das Anfangswertproblem (2.3) eine eindeutig bestimmte Lösung $y(t)$. Diese hängt stetig und $(r-1)$-mal differenzierbar von den Anfangswerten x_0 ab. z_0 ist eindeutig durch x_0 und $g(t_0, x_0, z_0) = 0$ bestimmt.

Kapitel 3

Numerische Lösung von Anfangswertproblemen für DAEs vom Index 1

In diesem Kapitel werden die *linearen Mehrschrittverfahren (LMM)* zur Lösung von Anfangswertproblemen (IVPs) für gewöhnliche Differentialgleichungen und DAEs eingeführt. Hierbei werden wir eine bestimmte Klasse von LMMs, die *Backward-Differentiation-Formulas (BDF)*, genauer betrachten, da sich diese als sehr effizient für die Behandlung von steifen Problemen erwiesen haben.
Zuerst werden wir diese Methoden und ihre Eigenschaften für gewöhnliche Differentialgleichungen auf äquidistanten Gittern untersuchen, später wird die Theorie auf variable Gitter erweitert. Zum Schluss wird erläutert, wie BDF-Methoden auf linear implizite DAEs angewendet werden. (Weitere Details hierzu, siehe [SW95] oder [HNW96, HNW93].)

3.1 Lineare Mehrschrittverfahren auf äquidistanten Gittern

Definition 3.1.1 (IVP für gewöhnliche Differentialgleichungen)
Sei $\mathcal{I} = [t_0, t_f] \subset \mathbb{R}$, $f : \mathbb{R} \times \mathbb{R}^{n_y} \longrightarrow \mathbb{R}^{n_y}$. Ein *Anfangswertproblem (IVP)* ist definiert als das Problem, eine Funktion $y : \mathbb{R} \longrightarrow \mathbb{R}^{n_y}$ zu finden, welche

dem System

$$\dot{y}(t) \quad = \quad f(t, y(t))$$

und den *Anfangswerten*

$$y(t_0) = y_0$$

genügt.

Bemerkung 3.1.2 (Steifheit)
Bei der Simulation von vor allem chemischen und biologischen Prozessen, aber auch bei manchen mechanischen oder elektrischen Prozessen, sowie der Diskretisierung von parabolischen partiellen Differentialgleichungen mit der Linienmethode, erhält man oft Modelle, die eine bestimmte Eigenschaft, genannt *Steifheit*, besitzen. Die ersten, die diese Eigenschaft entdeckten und beschrieben, waren Curtiss und Hirschfelder [CH52]. Zusammengefaßt lautet ihre Beschreibung: Steife Gleichungen sind solche, bei denen bestimmte implizite Verfahren, insbesondere BDF-Verfahren, besser, normalerweise erheblich besser, als explizite funktionieren. Hairer und Wanner charakterisieren dies in [HNW96] noch etwas prägnanter: Steife Gleichungen sind Probleme bei denen explizite Verfahren nicht funktionieren. Steifheit entzieht sich bis heute einer einheitlichen Definition, gebräuchliche Charakterisierungen für steife Systeme sind aber beispielsweise:

- Die Existenz von sich langsam verändernden Lösungen des Anfangswertproblems und die Eigenschaft, dass Lösungen, die sich in einer unmittelbaren Umgebung von diesen sich langsam ändernden Lösungen befinden, sich diesen schnell annähern.

- Es existieren Eigenwerte λ_i der Matrix $\frac{df}{dy}$, wobei f rechte Seite einer ODE ist, mit
$$\left\| \frac{df}{dy}(t, y(t)) \right\| |t_f - t_0| \gg 1,$$
für welche
$$\Re(\lambda_i) \ll 0$$
gilt.

Ein bekanntes Beispiel für ein steifes System ist ein System von chemischen Reaktionen, bei denen einige Reaktionen erheblich schneller ablaufen als andere.

Genau genommen sollte man i.a. nicht von einem steifen System an sich sprechen, sondern einem Anfangswertproblem, dass für gewisse Anfangswerte steif ist.

Definition 3.1.3 (Allgemeine lineare Mehrschrittverfahren)
Die allgemeine Form eines *linearen Mehrschrittverfahrens* mit k Schritten für die Berechnung einer Gitterfunktion $u_h(t)$ auf einem äquidistanten Gitter

$$I_h = \{t \in [t_0, t_f] : t = t_m, m = 0, 1, \ldots, N, t_m = t_0 + mh\}$$

als Näherung der Lösung $y(t)$ eines IVPs ist definiert durch:

1. Die Festlegung der k Startwerte $u_m = u_h(t_m), m = 0, 1, \ldots, k-1$ und
2. einer Differenzengleichung, um den nächsten Näherungswert u_{m+k} auszurechnen

$$\sum_{l=0}^{k} \alpha_l u_{m+l} = h \sum_{l=0}^{k} \beta_l f(t_{m+l}, u_{m+l}), m = 0, 1, \ldots, N-k, \qquad (3.1)$$

mit $\alpha_l, \beta_l \in \mathbb{R}$ und $\alpha_k \neq 0, |\alpha_0| + |\beta_0| \neq 0$.

Bemerkung 3.1.4

- Das Verfahren heißt linear, weil die Verfahrensfunktion

$$\phi(t_m, u_m, \ldots, u_{m+k}; h) = \sum_{l=0}^{k} \beta_l f(t_{m+l}, u_{m+l})$$

linear von den Werten $f(t_{m+l}, u_{m+l})$ abhängt.

- Die Bedingung $a_k \neq 0$ stellt sicher, dass die implizite Gleichung (3.1) eine eindeutige Lösung u_{m+k} besitzt (zumindest für h genügend klein).
- Die Bedingung $|\alpha_0| + |\beta_0| > 0$ stellt sicher, dass die Anzahl k von Schritten eindeutig ist.
- Für $\beta_k = 0$ ist das Verfahren explizit und die Lösung kann direkt berechnet werden. Andernfalls ist das Verfahren implizit und in jedem Schritt muss ein System von Gleichungen der Form

$$u_{m+k} = h\frac{\beta_k}{\alpha_k} f(t_{m+k}, u_{m+k}) + v, \qquad (3.2)$$

wobei

$$v = \frac{1}{\alpha_k} \sum_{l=0}^{k-1} [h\beta_l f(t_{m+l}, u_{m+l}) - \alpha_l u_{m+l}]$$

unabhängig von (t_{m+k}, u_{m+k}) ist, gelöst werden. Dieses System ist im Allgemeinen nichtlinear und wird normalerweise, abhängend von der behandelten Klasse von Problemen, mit Funktionaliterationen oder einem Newton- oder Newton-ähnlichen Verfahren gelöst.

Bemerkung 3.1.5
Aus Definition 3.1.3 erkennt man, dass ein lineares Mehrschrittverfahren aus 2 Phasen besteht:

- Einer Anfangsphase, in der die Näherungen $u_1, \ldots, u_{k-1}$ z.B. mit einem Runge-Kutta-Verfahren oder mit einem LMM mit niedrigerer Schrittzahl berechnet werden, und
- einer Laufphase, beschrieben durch (3.1), in der in jedem Schritt ein System von nichtlinearen Gleichungen gelöst werden muss.

3.1.1 Ordnung eines linearen Mehrschrittverfahrens

Definition 3.1.6 (Erzeugende Polynome)
Für ein lineares Mehrschrittverfahren 3.1.3 sind die *erzeugenden Polynome* definiert als

$$\rho(\xi) = \alpha_k \xi^k + \alpha_{k-1}\xi^{k-1} + \cdots + \alpha_0 \tag{3.3}$$

$$\sigma(\xi) = \beta_k \xi^k + \beta_{k-1}\xi^{k-1} + \cdots + \beta_0. \tag{3.4}$$

Definition 3.1.7 (Lokaler Diskretisierungsfehler, Konsistenzfehler)
Der *lokale Diskretisierungsfehler* σ bzw. der *Konsistenzfehler* τ eines LMM ist definiert als

$$h\tau(y(t), h) := \sigma[y(t), h] := L[y(t), h] = \sum_{l=0}^{k} \left[\alpha_l y(t + lh) - h\beta_l \dot{y}(t + lh)\right],$$

wobei L der *lineare Differenzenoperator* ist.

Definition 3.1.8 (Konsistenzordnung eines LMM)
Ein LMM ist von der *Konsistenzordnung* p, wenn für alle $y \in \mathcal{C}^{p+1}([t_0, t_f])$ gilt:

$$L[y(t), h] = \mathcal{O}(h^{p+1})$$

für $h \to 0$. Die Konsistenzordnung beschreibt, wie schnell der lokale Fehler für $h \to 0$ nach 0 geht.

Lemma 3.1.9
Ein LMM ist von der Konsistenzordnung p, wenn die folgenden Bedingungen erfüllt sind:

$$\sum_{l=0}^{k} \alpha_l = 0, \tag{3.5}$$

$$\sum_{l=0}^{k} (l\alpha_l - \beta_l) = 0, \tag{3.6}$$

$$\sum_{l=0}^{k} [\frac{1}{\nu!} l^\nu \alpha_l - \frac{1}{(\nu-1)!} l^{\nu-1} \beta_l] = 0, \quad \nu = 2, 3, \ldots, p. \tag{3.7}$$

Bemerkung 3.1.10 (Konsistenzbedingungen)
Die Bedingungen an ein LMM die Konsistenzordnung 1 zu besitzen, nennt man *Konsistenzbedingungen*. Sie können mit Hilfe der erzeugenden Polynome als

$$\rho(1) = 0, \qquad \rho'(1) = \sigma(1)$$

dargestellt werden.
Hat ein Verfahren mindestens die Konsistenzordnung 1, so ist $\xi_1 = 1$ eine Nullstelle von $\rho(\xi)$ und das Verfahren heißt *konsistent*.

3.1.2 Nullstabilität eines LMMs

Betrachtet man LMMs in der Praxis, so bemerkt man, dass ein LMM, trotz hoher Konsistenzordnung und somit kleinem lokalen Diskretisierungsfehler, nicht konvergieren muss. Dies beruht auf der Fehlerfortpflanzung aus fehlerhaften früheren Werten u_m und den Auswertungen der rechten Seite f an diesen Werten.
Vernachlässigt man nun die Fehlerfortpflanzung in den f_m-Termen, indem man den Fall $h \to 0$ betrachtet, so erhält man das Konzept der Nullstabilität:

Definition 3.1.11 (Nullstabilität)
Ein LMM heißt *nullstabil*, wenn das erzeugende Polynom $\rho(\xi)$ folgende Bedingungen erfüllt:

a) Die Nullstellen von $\rho(\xi)$ liegen auf dem Rand oder innerhalb des Einheitskreises und

b) die Nullstellen auf dem Rand des Einheitskreises sind einfach.

Bemerkung 3.1.12
Für ein konsistentes LMM definiert man noch folgende Begriffe:
Liegt mehr als eine Nullstelle auf dem Rand des Einheitskreises, so heißt das Verfahren *schwach stabil* (oder *schwach instabil*).
Sind alle von $\xi_1 = 1$ verschiedene Nullstellen innerhalb des Einheitskreises, heißt das Verfahren *stark stabil*.

3.1.3 Konvergenz eines LMMs

Dahlquist zeigte 1956, dass Konsistenz und Nullstabilität notwendige und hinreichende Bedingungen für die Konvergenz eines LMMs sind [Dah56].

Definition 3.1.13 (Konvergenz eines LMMs)
Ein LMM ist *konvergent*, wenn für alle IVPs, für die f Lipschitz-stetig auf $S := \{(t, y) : t_0 \leq t \leq t_f, y \in \mathbb{R}^{n_y}\}$ ist und für alle Startwerte $u_h(t_0 + mh)$, $m = 0, 1, \ldots, k-1$ mit

$$||y(t_0 + mh) - u_h(t_0 + mh)|| \to 0 \text{ für } h \to 0$$

gilt, dass

$$\epsilon(y(t), h) := ||y(t) - u_h(t)|| \to 0,\ h \to 0,\ t \in [t_0, t_f],$$

wobei ϵ der *globale Fehler* des LMMs ist.

Definition 3.1.14 (Konvergenzordnung)
Ein LMM ist *konvergent von der Ordnung* p, wenn für alle IVPs mit auf S genügend glattem f ein $h_0 > 0$ existiert, so dass für alle Startwerte $u_h(t_0 + mh)$, $m = 0, 1, \ldots, k-1$ mit

$$||y(t_0 + mh) - u_h(t_0 + mh)|| \leq C_0 h^p \quad \text{für } h \in (0, h_0]$$

dann

$$||y(t) - u_h(t)|| \leq Ch^p, \quad \text{für } h \in (0, h_0]$$

gilt.

Definition 3.1.15 (Stabilität ein LMMs)
Ein LMM ist *stabil*, wenn für alle genügend glatten Funktionen $y(t)$ von h unabhängige Konstanten $C_1, C_2 \in \mathbb{R}$ existieren, so dass für alle Gitter I_h gilt:

$$\max_{t \in I_h} ||\epsilon(y(t), t)|| \leq C_1 \max_{t \in I_h} ||\tau(y(t), t)|| + C_2 \, \epsilon_{start},$$

wobei ϵ_{start} das Maximum der Normen der Fehler in den Startwerten ist.

Folgerung 3.1.16
Wenn ein LMM konsistent und stabil ist und die Fehler in den Startwerten gleich 0 sind, so ist es konvergent und die Konvergenzordnung ist gleich der Konsistenzordnung.

Theorem 3.1.17 (Dahlquist)
Sei $y(t) \in \mathcal{C}^2(I, \mathbb{R}^{n_y})$ und f Lipschitz-stetig, dann ist ein nullstabiles und konsistentes LMM stabil.

Theorem 3.1.18 (Konvergenzkriterium)
Ein LMM ist genau dann konvergent, wenn es nullstabil und mindestens von Konsistenzordnung 1 ist.
Ein LMM ist genau dann konvergent und von der Konvergenzordnung p, wenn es nullstabil und von Konsistenzordnung p ist.

Dahlquist hat gezeigt, dass die Konsistenzordnung eines nullstabilen k-Schritt LMMs beschränkt ist [Dah56]:

Theorem 3.1.19 (Erste Dahlquist-Schranke)
Die Konsistenzordnung p eines nullstabilen k-Schritt LMMs erfüllt

$$\begin{aligned} p &\leq k+2 \quad \text{falls } k \text{ gerade,} \\ p &\leq k+1 \quad \text{falls } k \text{ ungerade,} \\ p &\leq k \quad \text{falls } \beta_k/\alpha_k \leq 0 \text{ (insbesondere also bei expliziten LMMs).} \end{aligned}$$

3.1.4 Absolute Stabilität eines LMMs

Bei dem Konzept der Nullstabilität haben wir die Fehlerfortpflanzung in den Termen $f(t_m, u_m)$ der rechten Seite des LMMs vernachlässigt. Dies kann dazu führen, dass wir, obwohl ein LMM nullstabil und konsistent - und damit konvergent - ist, Konvergenz nur für sehr kleine Schrittweiten h erhalten.
Die Untersuchung der Fehlerfortpflanzung in der rechten Seite führt auf das

Konzept der *absoluten Stabilität*.
Wir untersuchen das Verhalten eines LMM auf der repräsentativen Testgleichung von Dahlquist

$$\dot{y}(t) = \lambda y(t),\ \lambda \in \mathbb{C}. \tag{3.8}$$

Wenn wir auf diese Gleichung ein k-Schritt LMM anwenden, erhalten wir die lineare Differenzengleichung

$$(\alpha_k - h\lambda\beta_k)u_{m+k} + \ldots + (\alpha_0 - h\lambda\beta_0)u_m = 0. \tag{3.9}$$

Diese Gleichung hat genau dann stabile Lösungen u_l, wenn alle Nullstellen der *charakteristischen Gleichung*

$$\rho(\xi) - h\lambda\sigma(\xi) = 0$$

innerhalb oder auf dem Rand des Einheitskreises und mehrfache Nullstellen im Inneren des Einheitskreises liegen.
Dies führt zur folgenden Definition:

Definition 3.1.20 (Gebiet der absoluten Stabilität)
Das *Gebiet der absoluten Stabilität* eines LMMs ist definiert als

$$\mathcal{D} := \Big\{ h\lambda \in \mathbb{C} : \text{Alle Nullstellen von (3.9) erfüllen } |\xi_i(h\lambda)| \leq 1, \\ \text{und mehrfache Nullstellen erfüllen } |\xi_i(h\lambda)| < 1 \Big\}.$$

Bemerkung 3.1.21
Für $h = 0$ erhalten wir die Definition der Nullstabilität. Daher ist Nullstabilität äquivalent zu $0 \in \mathcal{D}$.

Beispiel 3.1.22 (Stabilitätsgebiete der Eulerverfahren)
Für das explizite Eulerverfahren $u_{m+1} = u_m + hf(t_m, u_m)$ haben wir die charakteristische Gleichung

$$-1 + \xi - h\lambda = 0,$$

und daher $|1 + h\lambda| \leq 1$ für $h\lambda \in \mathcal{D}$.
Für das implizite Eulerverfahren $u_{m+1} = u_m + hf(t_{m+1}, u_{m+1})$ haben wir

$$-1 + \xi - h\lambda\xi = 0.$$

Und damit $|1 - h\lambda| \geq 1$ für $h\lambda \in \mathcal{D}$ (vgl. Abb. 3.1).

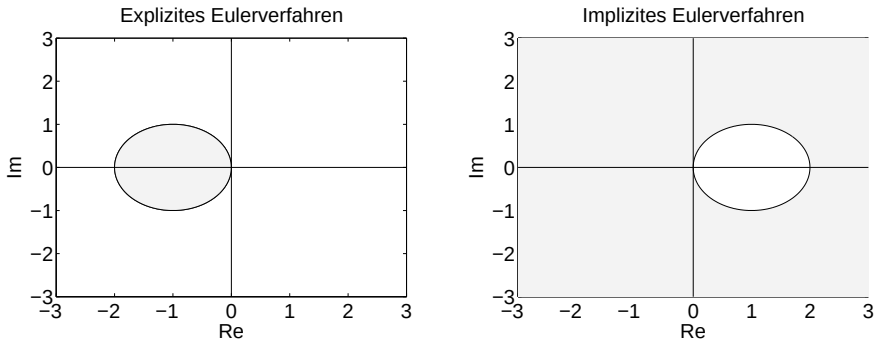


Abbildung 3.1: Stabilitätsgebiete des expliziten und impliziten Eulerverfahrens

Bemerkung 3.1.23
Das explizite Eulerverfahren ist ein repräsentatives Beispiel für die Ineffizienz von expliziten Verfahren bei der Behandlung von steifen Problemen, die, auf die Testgleichung übertragen, gewöhnlich dem Fall $\Re(\lambda) \ll 0$ entsprechen. Aufgrund seines beschränkten Stabilitätsgebietes kann man allein aus Stabilitätsgründen gezwungen sein, sehr kleine Schrittweiten zu wählen, unabhängig vom lokalen Diskretisierungsfehler. Zum Beispiel folgt für $\Re(\lambda) = -1000$ notwendigerweise $h < 10^{-3}$.

Die bei der Behandlung von steifen Problemen wünschenswerte Eigenschaft eines LMMs, dass $\mathbb{C}^- = \{z \in \mathbb{C} : \Re(z) \leq 0\}$ eine Teilmenge von $\mathcal{D}$ ist, hat Dahlquist definiert als:

Definition 3.1.24 (A-Stabilität eines LMMs)
Ein LMM ist *A-stabil* wenn $\mathbb{C}^- \subset \mathcal{D}$.

Dahlquist hat 1963 gezeigt, dass es eine Beschränkung für die Ordnung eines A-stabilen LMMs gibt [Dah63].

Theorem 3.1.25 (Zweite Dahlquist-Schranke)
Ein A-stabiles LMM besitzt eine Konsistenzordnung von $p \leq 2$.

Da eine maximale Konsistenzordnung von 2 in der Praxis i.A. nicht ausreichend ist um Probleme effizient zu lösen, führte Widlund eine leicht abgeschwächte Definition ein, um fast A-stabile Verfahren zu beschreiben.

Definition 3.1.26 (A(α)-Stabilität eines LMM)
Ein konvergentes LMM ist *A(α)-stabil* mit $0 < \alpha < \pi/2$ wenn

$$\mathcal{D}_\alpha := \{h\lambda : |\arg(-h\lambda)| < \alpha,\ h\lambda \neq 0\} \subset \mathcal{D},$$

wobei arg das komplexe Argument bedeutet.

3.2 Lineare Mehrschrittverfahren auf variablen Gittern

Wir gehen nun zu variablen Gittern über und transferieren einige unserer Definitionen und Ergebnisse vom äquidistanten Fall. Dabei werden wir beobachten, dass, mit einigen kleinen Anpassungen, die meisten Resultate ohne Probleme übertragen werden können.

Definition 3.2.1 (Allgemeines LMM auf variablen Gittern)
Ein *allgemeines LMM mit k Schritten auf einem variablen Gitter*

$$I_h = \{t \in [t_0, t_f] : t = t_m, m = 0, 1, \ldots, N,\ t_m = t_{m-1} + h_{m-1} \text{ für } m \neq 0\}$$

ist definiert durch

$$\sum_{l=0}^{k} \alpha_{lm} u_m = h_{m+k-1} \sum_{l=0}^{k} \beta_{lm} f(t_{m+l}, u_{m+l}),\ m = 0, \ldots, N-k, \qquad (3.10)$$

$$u_h(t_m) = u_m, \qquad m = 0, \ldots, k-1,$$

wobei $\alpha_{lm}, \beta_{lm} \in \mathbb{R}$, $|\alpha_{0m}| + |\beta_{0m}| \neq 0$ und α_{lm}, β_{lm} von den Schrittweitenänderungen $\omega_i := h_i/h_{i-1}$, $i = m+1, \ldots, m+k-1$, abhängen. Wir normieren im Folgenden so, dass $\alpha_{km} = 1$.

3.2.1 Konsistenzordnung von LMMs auf variablen Gittern

Ähnlich zu 3.1.8 definieren wir

Definition 3.2.2 (Konsistenzordnung auf variablem Gitter)
Ein LMM (3.10) besitzt *Konsistenzordnung p*, wenn für alle Polynome $P(t)$ mit $\deg(P) \leq p$ und alle Gitter I_h gilt:

$$\sum_{l=0}^{k} \alpha_{lm} P(t_{m+l}) = h_{m+k-1} \sum_{l=0}^{k} \beta_{lm} \dot{P}(t_{m+l}).$$

Theorem 3.2.3
Sei das LMM (3.10) von Konsistenzordnung p und sei $f \in \mathcal{C}^p(S)$. Zusätzlich gelte:

1. Die Schrittweitenänderungen $\omega_i = h_i/h_{i-1}$, $i = m+1, \ldots, m+k-1$, sind beschränkt für alle m und
2. die Koeffizienten α_{lm}, β_{lm} sind beschränkt.

Dann ist der lokale Diskretisierungsfehler, der analog zum äquidistanten Fall definiert ist, von der Ordnung $\mathcal{O}(h_m^{p+1})$.

3.2.2 Stabilität

Definition 3.2.4
Für ein LMM (3.10) mit Koeffizienten α_{lm} definieren wir zur einfacheren Notation die Matrizen

$$A_m := \begin{pmatrix} -\alpha_{k-1,m} & -\alpha_{k-2,m} & \ldots & -\alpha_{1,m} & -\alpha_{0,m} \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{pmatrix}.$$

Definition 3.2.5 (Stabilität auf variablem Gitter)
Ein LMM auf einem variablen Gitter ist *stabil*, wenn es ein $M \in \mathbb{R}$ gibt mit

$$||A_{m+j} A_{m+j-1} \ldots A_{m+1} A_m|| \leq M$$

für alle m und $j \geq 0$.

Wir erhalten den folgenden Zusammenhang zwischen Stabilität auf variablen Gittern und starker Stabilität auf äquidistanten Gittern, bewiesen von Crouzeix und Lisbona [CL84].

Theorem 3.2.6 (Stabilität auf variablen Gittern)
Für das LMM (3.10) gelte

1. $\sum_{l=0}^{k-1} \alpha_{lm} = -1$,
2. die Koeffizienten $\alpha_{lm} = \alpha_{lm}(\omega_{m+1}, \ldots, \omega_{m+k-1})$ sind stetig in einer Umgebung von $(1, \ldots, 1)$ und

3. das LMM ist stark stabil auf allen äquidistanten Gittern.

Dann existieren $\omega, \Omega \in \mathbb{R}$, mit $\omega < 1 < \Omega$, so dass das LMM stabil ist, falls

$$\omega \leq \omega_m \leq \Omega$$

für alle m gilt.

3.2.3 Konvergenz

Nun können wir die Konvergenzaussage für variable Gitter formulieren.

Theorem 3.2.7 (Konvergenz eines LMMs auf variablem Gitter)
Das LMM (3.10) sei stabil, von Konsistenzordnung p und die Koeffizienten α_{lm}, β_{lm} seien beschränkt. Für die Startwerte $u(t_m),\ m = 0, 1, \ldots, k-1$ gelte

$$||y(t_m) - u(t_m)|| = \mathcal{O}(h_0^p)$$

und die Schrittweitenänderungen ω_m seien beschränkt für alle $m \geq 1$.
Dann ist das LMM konvergent von der Ordnung p, d.h. es existiert ein $C \in \mathbb{R}$, so dass

$$||y(t_m) - u(t_m)|| \leq Ch^p, \quad t_m \in [t_0, t_f], \quad h = \max_m h_m.$$

3.3 BDF-Verfahren

Jetzt stellen wir die Backward-Differentiation-Formulas (BDF) vor. Diese wurden eingeführt von Curtiss und Hirschfelder [CH52] und wurden weiter bekannt durch die Untersuchungen von Gear [Gea71]. Wir beginnen mit ihrer Anwendung auf ODEs, erwähnen einige ihrer Eigenschaften und zeigen, wie man sie auf linear implizite DAEs anwendet.

3.3.1 BDF-Verfahren für ODEs

Die zugrunde liegende Idee bei BDF-Verfahren ist es, ein Interpolationspolynom durch die $(k+1)$ Werte $u_m, \ldots, u_{m+k}$ zu legen, und zu verlangen, dass dieses der ODE an der Stelle t_{m+k} genügt.

Definition 3.3.1 (BDF-Verfahren)
Das *k-Schritt BDF-Verfahren* ist definiert durch Festlegung der k Startwerte und die Differenzengleichung

$$\sum_{l=0}^{k} \alpha_{lm} u_{m+l} = h_{m+k-1} f(t_{m+k}, u_{m+k}), \quad m = 0, \ldots, N-k,$$

mit $\alpha_{lm} \in \mathbb{R}$, $\alpha_0, \alpha_k \neq 0$. Dabei werden die α_{lm} als Koeffizienten der Ableitung des Interpolationspolynoms, geteilt durch h_{m+k-1}, gewonnen. Da $\beta_k = 1 \neq 0$, sind die BDF-Verfahren implizite Verfahren.

Theorem 3.3.2 (Ordnung der BDF-Verfahren)
Ein k-Schritt BDF-Verfahren besitzt Konsistenzordnung k.

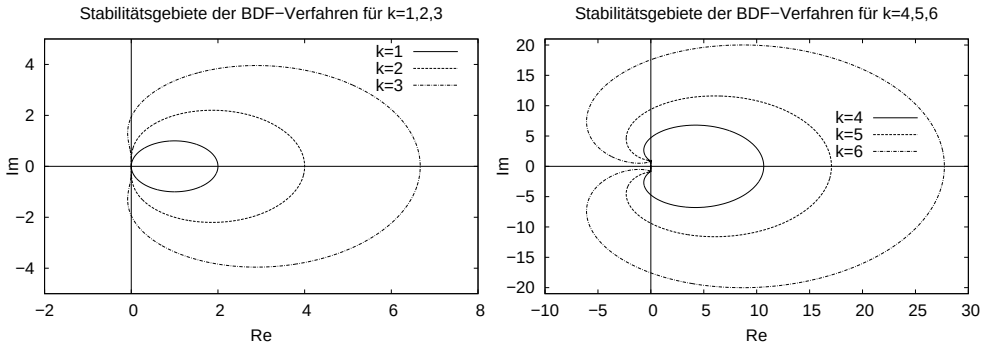
Im Gegensatz zu anderen LMMs sind die BDF-Verfahren nicht automatisch nach Konstruktion nullstabil, so dass es nötig ist, dies nachzurechnen. Cryer zeigte in diesem Zusammenhang folgenden Satz [Cry72]:

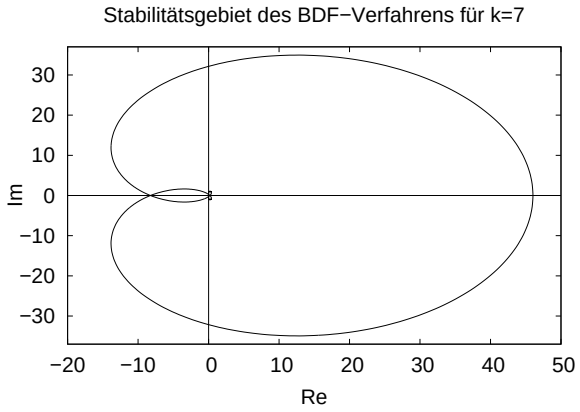
Theorem 3.3.3 (Nullstabilität der BDF-Verfahren)
Die BDF-Verfahren (auf äquidistanten Gittern) sind nullstabil für $k \leq 6$ und instabil für $k \geq 7$.

Daher sind nur die Verfahren bis zu $k = 6$ in der Praxis von Bedeutung (für lokale Fehlerschätzung ggf. bis $k = 7$).

Theorem 3.3.4 (Stabilitätsgebiete der BDF-Verfahren)
Die BDF-Verfahren sind A-stabil für $k \leq 2$ und A(α)-stabil für $k \leq 6$ mit folgenden Werten für α:

k	1	2	3	4	5	6
α	90°	90°	86.03°	73.35°	51.84°	17.84°





Dabei sind die Stabilitätsgebiete in obigen Abbildungen stets die Bereiche außerhalb der geschlossenen Kurven.

Griegorieff untersuchte das Verhalten der BDF-Verfahren auf variablen Gittern [Gri83]:

Theorem 3.3.5 (Stabilität von BDF-Verfahren auf variablen Gittern) Die BDF-Verfahren sind stabil auf variablen Gittern, wenn folgende Schranken an die Schrittweitenänderungen eingehalten werden:

k	2	3	4	5	6
ω	0	0.836	0.979	0.997	$1+\delta_1$
Ω	2.414	1.127	1.019	1.003	$1-\delta_2$

wobei δ_1, δ_2 gewisse Werte mit $0 < \delta_1, \delta_2 < 0.001$ sind. Das BDF-Verfahren mit $k = 1$ ist ein Einschrittverfahren, welches auf jedem Gitter stabil ist.

Bemerkung 3.3.6

Die obigen Schranken an die Schrittweitenänderungen sind sehr restriktiv, da sie alle möglichen Schrittweitenänderungen berücksichtigen. In der Anwendung sind diese Schranken zur effizienten Schrittweitensteuerung praktisch nicht zu gebrauchen. Wir kommen in Abschnitt 5.4 auf dieses Thema zurück.

3.3.2 BDF-Verfahren für DAEs vom Index 1

Wir betrachten nun das IVP für die linear implizite DAE (2.3)

$$\begin{aligned} A(t,x(t),z(t))\,\dot{x}(t) &= f(t,x(t),z(t)), \quad x(t_0)=x_0\in\mathbb{R}^{n_x}, \\ 0 &= g(t,x(t),z(t)), \quad z(t_0)=z_0\in\mathbb{R}^{n_z}, \\ t &\in [t_0,t_f]\subset\mathbb{R}, \\ x(t) &\in \mathbb{R}^{n_x}, \\ z(t) &\in \mathbb{R}^{n_z}, \\ A(t,x(t),z(t)) &\in \mathbb{R}^{n_x\times n_x} \text{ regulär} \\ g_z(t,x(t),z(t)) &\in \mathbb{R}^{n_z\times(n_x+n_z)} \text{ regulär} \end{aligned} \tag{3.11}$$

und wählen hier den sogenannten *indirekten Ansatz*, um ein Diskretisierungsschema zu erhalten: Da sowohl g_z als auch A regulär sind, existiert nach dem Satz der impliziten Funktion in einer gewissen Umgebung einer Lösung für alle $t\in[t_0,t_f]$ eine glatte Funktion G und eine lokal eindeutige Lösung der algebraischen Gleichungen $z(t)=G(x(t))$.
Eingesetzt in das ursprüngliche Problem führt dies auf das IVP

$$A(t,x(t),G(x(t)))\,\dot{x}(t)=f(t,x(t),G(x(t))), \quad x(t_0)=x_0\in\mathbb{R}^{n_x}.$$

Wenden wir nun ein BDF-Verfahren auf dieses IVP an, so erhalten wir auf variablem Gitter das Diskretisierungsschema

$$\begin{aligned} A(t_{m+k},x_{m+k},z_{m+k})\sum_{l=0}^{k}\alpha_{lm}x_{m+l} &= h_{m+k-1}f(t_{m+k},x_{m+k},z_{m+k}) \\ 0 &= g(t_{m+k},x_{m+k},z_{m+k}), \end{aligned} \tag{3.12}$$

mit $m=0,\dots,N-k$.
Die Konvergenz der BDF-Verfahren angewendet auf die DAE (3.11) folgt aus folgendem Theorem:

Theorem 3.3.7
Sei das LMM (3.10) von der Konsistenzordnung p, liege 0 im Stabilitätsgebiet, seien die Anfangswerte der DAE konsistent, d.h. $g(t_0,x_0,z_0)=0$, und seien die Fehler in den Startwerten des BDF-Verfahrens von der Ordnung $\mathcal{O}(h^p)$, dann ist das LMM konvergent von der Ordnung p.

Kapitel 4

Ableitungserzeugung bei DAEs vom Index 1

In diesem Kapitel stellen wir Methoden zur Generierung von Ableitungen der Lösungen von DAEs nach Anfangswerten, Parametern und Kontrollen, den sogenannten Sensitivitätsinformationen oder kurz Sensitivitäten, vor, die in vielen Anwendungen im Optimierungskontext benötigt werden.
Zuerst erläutern wir die benötigte zugrunde liegende Theorie und gehen im Anschluss daran auf zwei Ansätze zur numerischen Berechnung der Ableitungsinformationen mittels eines adaptiven numerischen Verfahrens ein: Die *Externe Numerische Differentiation (END)* und die *Interne Numerische Differentiation (IND).*

4.1 Die Variationsdifferentialgleichungen

Wir betrachten in den folgenden Kapiteln ODE-IVPs und DAE-IVPs, bei denen die Modellfunktionen f, g und A außer von der unabhängigen Variablen t und den Zustandsvariablen x bzw. z zusätzlich von Parametern p und Kontrollen q abhängen können.

Definition 4.1.1 (ODE-IVP mit Parametern und Kontrollen)
Mit Parametern und Kontrollen lautet das IVP für ODEs

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), p, q) \in \mathbb{R}^{n_y} \\ y(t_0) &= y_0 \in \mathbb{R}^{n_y}, \\ t &\in [t_0, t_f] \subset \mathbb{R}, \\ p &\in \mathbb{R}^{n_p}, \\ q &\in \mathbb{R}^{n_q}. \end{aligned} \tag{4.1}$$

Ist f stetig, so können wir dieses Problem in differentieller Form äquivalent in der sogenannten Integralform formulieren als

$$y(t) = y(t_0) + \int_{t_0}^{t_f} f(\tau, y(\tau), p, q)\, d\tau. \tag{4.2}$$

Offensichtlich hängt die Lösung $y(t)$ hierbei sowohl von den Anfangswerten y_0, als auch von den Parametern p und den Kontrollen q ab. Wir drücken dies mit der Schreibweise $y(t; y_0, p, q)$ aus.
Es gelten folgende Aussagen über die Differenzierbarkeit der Lösungen der ODE nach diesen Größen, die man prinzipiell durch Induktion und Anwendung der grundlegenden Differentiations- und Integrationsregeln beweist (Beweis, siehe z.B. [CL55]).

Satz 4.1.2 (Differenzierbare Abhängigkeit)
Sei $k \in \mathbb{N}^+$. Wenn $f \in \mathcal{C}^k(S, \mathbb{R}^{n_y})$, $S = [t_0, t_f] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_q}$ und die Ableitungen von f auf S beschränkt sind, so ist die eindeutige Lösung $y(t; y_0, p, q)$ von (4.1)

- k-fach stetig differenzierbar in y_0, p und q und
- $(k+1)$-fach stetig differenzierbar in t.

Wir erhalten zur Berechnung der Ableitungen formal durch Differenzieren von (4.2) nach y_0, p bzw. q und Umwandlung in die differentielle Form Anfangswertprobleme für die Sensitivitäten, die sogenannten *Variationsdifferentialgleichungen (VDE)*.

Satz 4.1.3 (Variationsdifferentialgleichungen)
Man erhält die Ableitungen der Lösung $y(t; y_0, p, q)$ des zugrunde liegenden IVPs als Lösung der *Variationsdifferentialgleichungen*

$$\begin{aligned} \frac{d}{dt}\frac{\partial y(t; y_0, p, q)}{\partial y_0} &= f_y(t, y(t; y_0, p, q), p, q)\ \frac{\partial y(t; y_0, p, q)}{\partial y_0}, \\ \frac{\partial y(t_0; y_0, p, q)}{\partial y_0} &= Id \in \mathbb{R}^{n_y \times n_y} \end{aligned}$$

für die Ableitung nach y_0, bzw.

$$\begin{aligned} \frac{d}{dt}\frac{\partial y(t; y_0, p, q)}{\partial \zeta} &= f_y(t, y(t; y_0, p, q), p, q)\ \frac{\partial y(t; y_0, p, q)}{\partial \zeta} \\ &\quad + f_\zeta(t, y(t; y_0, p, q), p, q) \\ \frac{\partial y(t_0; y_0, p, q)}{\partial \zeta} &= 0 \in \mathbb{R}^{n_\zeta \times n_\zeta} \end{aligned}$$

für $\zeta \in \{p, q\}$.

Bemerkung 4.1.4
Da in der Variationsdifferentialgleichung auch $y(t; y_0, p, q)$ auftritt, erfordert ihre Lösung i.a. stets auch die Lösung des zugrunde liegenden IVPs.
Die Lösung einer VDE existiert bzw. ist eindeutig, wenn die Lösung des zugrunde liegenden IVP existiert bzw. eindeutig ist.
Falls die Anfangswerte y_0 von p oder q abhängen, müssen die Anfangswerte der entsprechenden VDE zu $\frac{\partial y(t_0; y_0, p, q)}{\partial \zeta} = \frac{\partial y_0}{\partial \zeta}$ modifiziert werden.

Beispiel 4.1.5
Betrachten wir das einfache IVP mit einem skalaren Parameter $p \in \mathbb{R}$

$$\dot{y}(t; y_0, p) = -p\ y(t; y_0, p),\ y(t_0) = y_0 \in \mathbb{R}.$$

Dieses besitzt die allgemeine Lösung

$$y(t; y_0, p) = y_0\ e^{-p\,(t - t_0)}.$$

Wir untersuchen den Fall $p = 1$ und $t_0 = 0$ sowie $y_0 = 1$, bei dem man die VDEs leicht analytisch lösen kann.

Für die VDE der Ableitung nach y_0 erhalten wir dann

$$\begin{aligned}
\frac{d}{dt}\frac{\partial y(t;y_0,p)}{\partial y_0} &= f_y(t,y(t;y_0,p),p)\,\frac{\partial y(t;y_0,p)}{\partial y_0} \\
&= -p\frac{\partial y(t;y_0,p)}{\partial y_0} \\
\Longrightarrow \frac{d}{dt}\frac{\partial y(t;1,1)}{\partial y_0} &= -1\frac{\partial y(t;1,1)}{\partial y_0}, \\
\text{mit } \frac{\partial y(0;1,1)}{\partial y_0} &= 1.
\end{aligned}$$

Und damit in diesem einfachen Fall

$$\frac{\partial y(t;1,1)}{\partial y_0} = e^{-t}.$$

Für die VDE der Ableitung nach p ergibt sich:

$$\begin{aligned}
\frac{d}{dt}\frac{\partial y(t;y_0,p)}{\partial p} &= f_y(t,y(t;y_0,p),p)\,\frac{\partial y(t;y_0,p)}{\partial p} + f_p(t,y(t;y_0,p),p) \\
&= -p\frac{\partial y(t;y_0,p)}{\partial p} + (-1)\,y(t;y_0,p) \\
&= -1\frac{\partial y(t;1,1)}{\partial p} - y(t;1,1) \\
\Longrightarrow \frac{d}{dt}\frac{\partial y(t;1,1)}{\partial p} &= -1\frac{\partial y(t;1,1)}{\partial p} - e^{-t}, \\
\text{mit } \frac{\partial y(0;1,1)}{\partial p} &= 0.
\end{aligned}$$

Damit erhält man dann

$$\frac{\partial y(t;1,1)}{\partial p} = -te^{-t}.$$

Wir untersuchen nun den Fall der linear impliziten DAE mit Index 1 und werden für die Berechnung der Sensitivitäten der DAE dann *Variations-DAEs (VDAEs)* erhalten.

Definition 4.1.6 (DAE-IVP mit Parametern und Kontrollen)
Das *IVP für eine linear implizite DAE mit Parameter und Kontrollen* lautet

$$\begin{aligned} A(t,x(t),z(t),p,q)\,\dot{x}(t) &= f(t,x(t),z(t),p,q) \in \mathbb{R}^{n_x} \\ 0 &= g(t,x(t),z(t),p,q) \in \mathbb{R}^{n_z} \\ x(t_0) &= x_0 \in \mathbb{R}^{n_x} \\ z(t_0) &= z_0 \in \mathbb{R}^{n_z} \\ t &\in [t_0,t_f] \subset \mathbb{R} \\ x(t) \in \mathbb{R}^{n_x},\ z(t) \in \mathbb{R}^{n_z} &, \quad p \in \mathbb{R}^{n_p},\ q \in \mathbb{R}^{n_q} \\ A(t,x(t),z(t),p,q) &\in \mathbb{R}^{n_x \times n_x} \text{ regulär} \\ g_z(t,x(t),z(t),p,q) &\in \mathbb{R}^{n_z \times (n_x n_z)} \text{ regulär,} \end{aligned} \tag{4.3}$$

wobei hier die Abhängigkeit von x bzw. z von den Anfangswerten, Parametern und Kontrollen der Übersichtlichkeit halber nicht ausgeschrieben wurde.

Definition 4.1.7 (Wronski-Matrizen)
Wir definieren die *Wronski-Matrizen* $\mathcal{W}_\zeta$ (hier ohne Argumente) als

$$\mathcal{W}_\zeta := \begin{pmatrix} \mathcal{W}_\zeta^x \\ \mathcal{W}_\zeta^z \end{pmatrix} := \begin{pmatrix} \frac{\partial x}{\partial \zeta} \\ \frac{\partial z}{\partial \zeta} \end{pmatrix},$$

wobei $\zeta \in \{x_0, z_0, p, q\}$ oder auch eine beliebige Richtung aus dem $\mathbb{R}^{n_x+n_z+n_p+n_q}$ sein kann. Mit $\mathcal{W}$ bezeichnen wir

$$\mathcal{W} := \begin{pmatrix} \mathcal{W}^x \\ \mathcal{W}^z \end{pmatrix} := \begin{pmatrix} \mathcal{W}_{x_0}^x & \mathcal{W}_{z_0}^x & \mathcal{W}_p^x & \mathcal{W}_q^x \\ \mathcal{W}_{x_0}^z & \mathcal{W}_{z_0}^z & \mathcal{W}_p^z & \mathcal{W}_q^z \end{pmatrix}. \tag{4.4}$$

Mit diesen Bezeichnungen ergibt sich dann analog zu Satz 4.1.3 die VDAE für z.B. $\mathcal{W}_q$ als

$$\begin{aligned} (A_y\mathcal{W}_q + A_q)\dot{x} + A\dot{\mathcal{W}}_p^x &= f_y\mathcal{W}_q + f_q \\ 0 &= g_y\mathcal{W}_q + g_q, \end{aligned} \tag{4.5}$$

mit $y = (x^T, z^T)^T$. Allgemeiner können wir die VDAEs für die Ableitungen $\mathcal{W}$ nach den Anfangswerten, Parametern und Kontrollen zusammenfassen zu

$$\begin{aligned} A\dot{\mathcal{W}}^x &= \begin{pmatrix} (f_x - A_x\dot{x})^T \\ (f_z - A_z\dot{x})^T \\ (f_p - A_p\dot{x})^T \\ (f_q - A_q\dot{x})^T \end{pmatrix}^T \begin{pmatrix} \mathcal{W}^x_{x_0} & \mathcal{W}^x_{z_0} & \mathcal{W}^x_p & \mathcal{W}^x_q \\ \mathcal{W}^z_{x_0} & \mathcal{W}^z_{z_0} & \mathcal{W}^z_p & \mathcal{W}^z_q \\ 0 & 0 & Id_{n_p} & 0 \\ 0 & 0 & 0 & Id_{n_q} \end{pmatrix} \\ 0 &= \begin{pmatrix} g_x & g_z & g_p & g_q \end{pmatrix} \begin{pmatrix} \mathcal{W}^x_{x_0} & \mathcal{W}^x_{z_0} & \mathcal{W}^x_p & \mathcal{W}^x_q \\ \mathcal{W}^z_{x_0} & \mathcal{W}^z_{z_0} & \mathcal{W}^z_p & \mathcal{W}^z_q \\ 0 & 0 & Id_{n_p} & 0 \\ 0 & 0 & 0 & Id_{n_q} \end{pmatrix}. \end{aligned} \tag{4.6}$$

In vielen Anwendungen, so z.B. dem Optimalsteuerungs-Code MUSCOD-II [Lei99, BBLS99], oder dem Versuchsplanungs-Tool VPLAN [KÖ2], wird im allgemeinen nicht die komplette Ableitungsmatrix $\mathcal{W}$, also die Ableitungen nach allen Parametrisierungsvariablen, benötigt, sondern nur Ableitungen nach n_{ddir} bestimmten Richtungen $\zeta_i \in \mathbb{R}^{n_x+n_z+n_p+n_q}$, $i = 1, \ldots, n_{ddir}$. Fassen wir diese in eine Matrix

$$\mathcal{W}_{ddir} := \begin{pmatrix} \zeta_1 & \cdots & \zeta_{n_{ddir}} \end{pmatrix} \in \mathbb{R}^{(n_x+n_z+n_p+n_q)\times n_{ddir}}$$

zusammen, so können wir direkt die VDAE zur Berechnung der Richtungsableitungen $\overline{\mathcal{W}} = \mathcal{W} \cdot \mathcal{W}_{ddir}$ formulieren:

$$\begin{aligned} A\dot{\overline{\mathcal{W}}}^x &= \begin{pmatrix} (f_x - A_x\dot{x})^T \\ (f_z - A_z\dot{x})^T \\ (f_p - A_p\dot{x})^T \\ (f_q - A_q\dot{x})^T \end{pmatrix}^T \begin{pmatrix} \overline{\mathcal{W}}^x \\ \overline{\mathcal{W}}^z \\ (\mathcal{W}_{ddir})_p \\ (\mathcal{W}_{ddir})_q \end{pmatrix} \\ 0 &= \begin{pmatrix} g_x & g_z & g_p & g_q \end{pmatrix} \begin{pmatrix} \overline{\mathcal{W}}^x \\ \overline{\mathcal{W}}^z \\ (\mathcal{W}_{ddir})_p \\ (\mathcal{W}_{ddir})_q \end{pmatrix}, \end{aligned} \tag{4.7}$$

mit

$$\mathcal{W}_{ddir} = \begin{pmatrix} (\mathcal{W}_{ddir})_x \\ (\mathcal{W}_{ddir})_z \\ (\mathcal{W}_{ddir})_p \\ (\mathcal{W}_{ddir})_q \end{pmatrix}.$$

Bemerkung 4.1.8
Diese Formulierung erlaubt eine effiziente numerische Behandlung, da hierbei auch für die Modellfunktionen f, g und A nicht unbedingt die kompletten Jakobimatrizen berechnet werden müssen, sondern nur die entsprechenden Richtungsableitungen.

Bemerkung 4.1.9
Die obige Herleitung der VDEs bzw. VDAEs läßt sich bei genügend glatten Modellfunktionen f, g und A iterieren. Auf diese Weise können VDEs bzw. VDAEs auch für zweite und höhere Ableitungen ganz analog gewonnen werden. Beispielsweise erhält man für die gemischte zweite Ableitung W_{pq} nach Parametern und Kontrollen die VDAE:

$$\begin{aligned} &[W_q^T A_{yy} W_p + A_{yq} W_p + A_y W_{pq} + A_{py} W_q + A_{pq}]\dot{x} \\ &+(A_y W_p + A_p)\dot{W}_q^x + (A_y W_q + A_q)\dot{W}_p^x + A\dot{W}_{pq}^x &= \\ &W_q^T f_{yy} W_p + f_{yq} W_p + f_y W_{pq} + f_{py} W_q + f_{pq} & \\ &W_q^T g_{yy} W_p + g_{yq} W_p + g_y W_{p,q} + g_{py} W_q + g_{pq} &= 0. \end{aligned} \tag{4.8}$$

4.2 Numerische Berechnung der Ableitungsinformationen

Wir präsentieren nun die beiden grundlegenden Herangehensweisen zur numerischen Berechnung der Ableitungsinformationen, die *Externe Numerische Differentiation (END)* und die *Interne Numerische Differentiation (IND)*. Die Terminologie beruht darauf, dass bei der END die Differentiation außerhalb des Diskretisierungsschemas des numerischen Verfahrens erfolgt, bei der IND innerhalb.

4.2.1 Externe numerische Differentiation

Bei der END wird das numerische Verfahren prinzipiell als ein "Black-box"-Operator betrachtet, der zu gegebenen Eingabewerten eine Lösung liefert, ohne dass man den Vorgang im "Inneren" näher analysiert. Man approximiert die Ableitung der Lösung $y(t;\xi)$ mit $\xi = (y_0, p, q)$ in Richtung $\zeta \in \mathbb{R}^{n_y+n_p+n_q}$ durch finite Differenzen

$$\frac{\partial y(t;\xi)}{\partial \zeta} = \frac{y(t;\xi+\epsilon_\zeta\zeta) - y(t;\xi)}{\epsilon_\zeta} + \mathcal{O}(\epsilon_\zeta). \tag{4.9}$$

Dabei würde man zunächst die Lösung $y(t;\xi)$ des ursprünglichen DAE-IVPs, die sogenannte *Nominaltrajektorie*, berechnen, um dann die sogenannten *variierten Trajektorien* mit entsprechend der Richtung ζ gestörten Anfangswerten, Parametern und Kontrollen zu berechnen.
Werden dabei die Lösungen wie in unserem Fall mit einem adaptiven numerischen Verfahren berechnet, so führen i.a. schon kleine Änderungen an Anfangswerten, Parametern oder Kontrollen zu einer anderen Schrittweiten- oder Ordnungsfolge, also zu einem anderen Diskretisierungsschema. Dies kann zu Problemen führen, da die Ausgabe des Verfahrens dann i.a. nicht mehr differenzierbar von den Eingabegrößen abhängt, und damit auch bei minimalen Änderungen der Eingabewerte Sprünge in der Größenordnung der Integratorgenauigkeit zu erwarten sind, worauf Orthega und Rheinboldt [OR66], sowie Gear und Vu [GV83] hinwiesen.
Obiger Ansatz zur Approximation der Ableitungen führt im Anwendungskontext zu einem Effizienzproblem, wenn hohe Genauigkeiten der Ableitungen gefordert sind. Als Faustregel gilt, dass man für die Genauigkeit der Ableitungen maximal die Hälfte der Genauigkeit der Nominaltrajektorie und der variierten Trajektorien erhält [SB92]. Daher müssen für hohe Genauigkeiten der Ableitungen die Nominaltrajektorie und die variierten Trajektorien mit sehr hoher Genauigkeit berechnet werden, was zu einem drastischen Anstieg des Rechenaufwands führen kann.

4.2.2 Interne Numerische Differentiation

Bei der IND, deren Prinzip von Bock [Boc81] entwickelt wurde, wird das numerische Verfahren nicht mehr als "Black-box"-Operator behandelt, sondern vielmehr als eine Folge von Abbildungen angesehen. Da moderne "state-of-the-art" Methoden zur Erlangung ihrer Effizienz die Schrittweiten- und Ordnungsfolge, aber auch z.B. die iterative Lösung von Gleichungssystemen an das jeweilige Problem anpassen müssen, ist die Folge der Abbildungen nicht konstant, sondern sie wird adaptiv erzeugt. Diese adaptiv erzeugte Folge hängt i.a. nicht mehr stetig von den Eingabewerten - also den Anfangswerten, Parametern und Kontrollen - der Methode ab. Die Idee der IND besteht nun darin, die Folge der Abbildungen, die bei der Berechnung der Nominaltrajektorie adaptiv erzeugt wurde, zu differenzieren, nicht aber die adaptiven Komponenten selbst, die zur Erzeugung der Abbildungsfolge dienen.

Praktisch bedeutet dies, dass nach der Berechnung der Nominaltrajektorie alle adaptiven Komponenten eingefroren werden, unter anderem die Schrittweiten- und Ordnungssteuerung, die Fehlerschätzung sowie die Iterationsmatrizen und die Iterationsanzahl etwaiger iterativer Gleichungssystemlöser.
Wir unterscheiden im Folgenden zwischen zwei grundsätzlichen Ansätzen zur Generierung von Ableitungsinfomationen mittels der IND.

Die Methode der variierten Trajektorien

Hierbei werden die Wronski-Matrizen (4.4), ähnlich zur END, mittels finiten Differenzen approximiert, d.h. es wird sowohl das IVP für die Nominaltrajektorie $y(t;\xi)$ zu den ursprünglichen Anfangswerten, Parametern und Kontrollen gelöst, als auch die IVPs für die variierten Trajektorien $y(t;\xi+\epsilon_\zeta\zeta)$. Im Anschluß wird der Differenzenquotient gebildet. Im Unterschied zur END wird hierbei aber nur ein Diskretisierungsschema verwendet, also insbesondere in jedem Schritt jeweils dieselbe Schrittweite, Ordnung und dieselben Iterationsmatrizen sowohl für die Nominaltrajektorie wie auch für die variierten Trajektorien.
Ein Problem bei diesem Ansatz stellt die Wahl von ϵ_ζ dar. Als Faustregel gilt hier $\epsilon_\zeta = \sqrt{\epsilon_{mech}}$, mit ϵ_{mech} der Maschinengenauigkeit, falls $||\xi|| \approx ||\zeta|| \approx 1$ und die Komponenten ungefähr die gleiche Größenordnung haben.
Der Vorteil ist, dass jetzt die Trajektorien nicht mehr mit der bei der END benötigten extremen Genauigkeit berechnet werden müssen, um eine brauchbare Genauigkeit der Ableitungen zu erhalten. Beispielsweise genügt bei obiger Wahl von ϵ_ζ und gleichen Voraussetzungen eine Integrationsgenauigkeit von $\mathcal{O}(\sqrt{\epsilon_{mech}})$ um eine Genauigkeit der Ableitungen von $\mathcal{O}(\sqrt{\epsilon_{mech}})$ zu erhalten, im Vergleich zu $\mathcal{O}(\epsilon_{mech})$ bei der END [Lei95].

Lösung der Variations-DAE

Der andere Ansatz zur Generierung der Ableitungen besteht in der Berechnung der Lösungen der VDAEs. Betrachten wir das BDF-Diskretisierungsschema (3.12) der Nominaltrajektorie im $(n+1)$-ten Schritt, ergänzt um die Abhängigkeit von Parametern und Kontrollen, so erhalten wir

mit $y_{n+1} = (x_{n+1}^T, z_{n+1}^T)^T$

$$\begin{pmatrix} A(t_{n+1}, y_{n+1}, p, q) \sum_{i=0}^{k} \alpha_{k-i,n} x_{n+1-i} + h_n f(t_{n+1}, y_{n+1}, p, q) \\ g(t_{n+1}, x_{n+1}, z_{n+1}, p, q) \end{pmatrix} = 0. \tag{4.10}$$

Der Idee der IND entsprechend wird dies nach einer Ableitungsrichtung ζ differenziert. Wir führen dies am Beispiel der Ableitung nach den Kontrollen q durch und erhalten (mit eingefrorenen adaptiven Komponenten)

$$\begin{aligned} A(t_{n+1}, y_{n+1}, p, q) \sum_{i=0}^{k} \alpha_{k-i,n} \frac{\partial x_{n+1-i}}{\partial q} \quad + \\ \left(A_y(t_{n+1}, y_{n+1}, p, q) \frac{\partial y_{n+1}}{\partial q} + A_q(t_{n+1}, y_{n+1}, p, q) \right) \sum_{i=0}^{k} \alpha_{k-i,n} x_{n+1-i} \quad + \\ h_n f_y(t_{n+1}, y_{n+1}, p, q) \frac{\partial y_{n+1}}{\partial q} + h_n f_q(t_{n+1}, y_{n+1}, p, q) \quad = \quad 0 \\ g_y(t_{n+1}, y_{n+1}, p, q) \frac{\partial y_{n+1}}{\partial q} + g_q(t_{n+1}, y_{n+1}, p, q) = 0. \end{aligned} \tag{4.11}$$

Vergleichen wir nun das BDF-Diskretisierungsschema, das wir erhalten, wenn wir die VDAE (4.5) für $\mathcal{W}_q$ diskretisieren, mit dem Schema (4.11), so stellen wir fest, dass sie übereinstimmen. Leiten wir nun auch noch das iterative Verfahren zur Lösung der auftretenden Gleichungssysteme (siehe Abschnitt 5.3) ab, so ist die mit obigem Schema berechnete Lösung die exakte Ableitung der diskretisierten Nominaltrajektorie. Der lokale Diskretisierungsfehler der Lösung der VDE ist dabei von derselben Ordnung wie derjenige der Nominaltrajektorie [Boc87].

Bei diesem Ansatz werden neben den Ableitungen der Modellfunktionen nach den Zustandsvariablen y auch die Ableitungen derselbigen nach den Parametern und Kontrollen, bzw. nach kombinierten Richtungen all dieser benötigt.

Kapitel 5

DAESOL-II

In diesem Kapitel stellen wir den im Zuge dieser Diplomarbeit implementierten Integrator DAESOL-II vor. Wir beschreiben die bei der Implementierung verwendeten Strategien, gehen auf einige technische Aspekte ein und beschreiben den grundlegenden Aufbau des Integrators.

Ziel der Entwicklung von DAESOL-II war es, einen "state-of-the-art" Integrator zu implementieren, der auf modernen Theorien und Strategien zur Behandlung von DAE-IVPs beruht, wie sie z.B. in [BSS95] beschrieben sind. Desweiteren sollte er eine solide und einfach erweiterbare Grundlage für Weiterentwicklungen wie die Behandlung von Unstetigkeitsstellen in den rechten Seiten, Retardierungen und nicht zuletzt einer Parallelisierung darstellen. Dabei sollten moderne Software-Bibliotheken verwendet werden, welche die vorhandene Hardware optimal ausnutzen können.
DAESOL-II leistet in seiner ersten Version unter anderem folgendes:

- schnelle und exakte Lösung von IVPs für steife linear implizite DAEs vom Index 1,
- effiziente Erzeugung von Ableitungen dieser Lösungen nach Anfangswerten, Parametern und Kontrollen,
- einfache Erweiterbarkeit ohne Expertenwissen über den gesamten Integrator durch modularen Aufbau mit präzise definierten Schnittstellen,
- klare und benutzerfreundliche Bedienung für den Anwender bei umfangreichen Konfigurationsmöglichkeiten und

- Einsetzbarkeit auf vielen Systemen mit effizienter Ausnutzung der vorhandenen Hardware.

5.1 Darstellung der Interpolationspolynome im BDF-Verfahren

Das Kernstück von DAESOL-II ist ein BDF-Verfahren zur Lösung von linear impliziten DAE-IVPs auf einem variablen Gitter, wie es in 3.3.2 beschrieben wurde.
Zur Darstellung der innerhalb des BDF-Verfahrens benötigten Interpolationspolynome verwenden wir die Newton-Darstellung. Diese ermöglicht eine effiziente Speicherung sowie Aktualisierung der Interpolationspolynome von einem Schritt zum nächsten.

Definition 5.1.1 (Newton-Darstellung)
In der *Newton-Darstellung* ist das Interpolationspolynom $\mathcal{P}$ durch $(k+1)$ Punkte $p_i = p(t_i), i = 0, \ldots, k$ an der Stelle t gegeben durch

$$\mathcal{P}(t; p_0, \ldots, p_k) = \sum_{i=0}^{k} N_i(t) p[t_k, \ldots, t_{k-i}], \tag{5.1}$$

wobei die sogenannten *dividierten Differenzen* $p[\ldots]$ rekursiv definiert sind durch

$$\begin{aligned} p[t_i] &:= p(t_i) \\ p[t_{i+j}, \ldots, t_i] &:= \frac{p[t_{i+j}, \ldots, t_{i+1}] - p[t_{i+j-1}, \ldots, t_i]}{t_{i+j} - t_i} \end{aligned} \tag{5.2}$$

und die *Newtonpolynome* N_i durch

$$N_i(t) = \begin{cases} \prod_{l=1}^{i} (t - t_{k-l}) & \text{für } i = 1, \ldots, k \\ 1 & \text{für } i = 0. \end{cases} \tag{5.3}$$

Bei der Implementierung des BDF-Verfahrens wird in jedem Schritt das dem Verfahren zugrunde liegende Interpolationspolynom $\mathcal{P}^C$ benötigt. Dieses ist bei einem k-Schritt BDF-Verfahren vom Grad k und interpoliert die $(k+1)$ Werte $x_{n+1-i}, i = 0, \ldots, k$ der differentiellen Variablen und genügt in t_{n+1} der DAE.
Mit diesen Bedingungen ist sowohl das Interpolationspolynom $\mathcal{P}^C$, wie auch

sein Wert $y_{n+1}^C = \mathcal{P}^C(t_{n+1})$ eindeutig festgelegt. Wir bezeichnen dieses Interpolationspolynom als *Korrektorpolynom* und seinen Wert an der Stelle t_{n+1} als *Korrektor*.
Da das resultierende Gleichungssystem im Diskretisierungsschema (3.12) nichtlinear ist, benötigen wir zur iterativen Lösung zudem einen Startwert für die differenziellen und algebraischen Variablen, den wir als *Prädiktor* bezeichnen.
Wir gewinnen ihn mit Hilfe eines zweiten Interpolationspolynoms $\mathcal{P}^P$ vom Grad k durch die letzten $k+1$ Werte $y_{n+1-i}, i = 1, \ldots, k+1$, dem *Prädiktorpolynom*. Da Prädiktor- und Korrektorpolynom bei der Behandlung von ODEs und DAEs jeweils von der gleichen Gestalt sind, entwickeln wir sie im folgenden der Übersichtlichkeit halber nur für ODEs.

Wir definieren die folgenden Bezeichnungen, die es uns ermöglichen, die Berechnung und Speicherung der Interpolationspolynome sowie den Übergang von einem Schritt zum nächsten effizient zu beschreiben.

Definition 5.1.2
Wir definieren die einzelnen Faktoren der Newtonpolynome (5.3) (für den Wert $t = t_{n+1}$)

$$\psi_i(n+1) := t_{n+1} - t_{n+1-i} = h_n + \ldots + h_{n+1-i} = \psi_{i-1}(n) + h_n, \tag{5.4}$$

die Quotienten aus den neuen und alten N_i's

$$\beta_i(n+1) := \begin{cases} 1 & \text{für } i = 1 \\ \frac{\psi_1(n+1)\cdot\ldots\cdot\psi_{i-1}(n+1)}{\psi_1(n)\cdot\ldots\cdot\psi_{i-1}(n)} & \text{für } i > 1, \end{cases} \tag{5.5}$$

die *modifizierten dividierten Differenzen*

$$\Phi_i(n) := \begin{cases} y_n & \text{für } i = 1 \\ \psi_1(n)\cdot\ldots\cdot\psi_{i-1}(n)\; y[t_n, \ldots, t_{n-i+1}] & \text{für } i > 1, \end{cases} \tag{5.6}$$

sowie

$$\Phi_i^*(n) := \beta_i(n+1)\; \Phi_i(n) \tag{5.7}$$

$$\gamma_i(n+1) := \sum_{j=1}^{i-1} \frac{1}{\psi_j(n+1)} \tag{5.8}$$

$$\delta_i(n+1) := \sum_{j=1}^{i} \Phi_j^*(n). \tag{5.9}$$

Dabei vereinbaren wir leere Produkte als 1 und leere Summen als 0.

Als nächstes zeigen wir, wie Prädiktor und Korrektor mittels dieser Grössen dargestellt werden.

Lemma 5.1.3 (Darstellung des Prädiktors)
Für das Prädiktorpolynom im $(n+1)$-ten Schritt gilt

$$\mathcal{P}^P(t_{n+1-i}) = y_{n+1-i},\ i = 1, \ldots, k+1. \tag{5.10}$$

Mit der Newtondarstellung (5.1) und obigen Bezeichnungen erhalten wir dann die Darstellung

$$\begin{aligned} y^P_{n+1} &= \mathcal{P}^P(t_{n+1}) \\ &= \sum_{j=0}^{k} \prod_{i=0}^{j-1} (t_{n+1} - t_{n-i})\ y[t_n, \ldots, t_{n-j}] \\ &= \sum_{j=1}^{k+1} \psi_1(n+1) \ldots \psi_{j-1}(n+1)\ y[t_n, \ldots, t_{n-j+1}] \\ &= \sum_{j=1}^{k+1} \Phi^*_j(n) \qquad (5.11) \\ &= \delta_{k+1}(n+1). \end{aligned}$$

Damit sind die $\delta_i(n+1)$ die Partialsummen des Korrektors im Schritt $(n+1)$. Für die Ableitung $\dot{\mathcal{P}}^P(t_{n+1})$ an der Stelle t_{n+1} ergibt sich:

$$\begin{aligned} \dot{\mathcal{P}}^P(t_{n+1}) &= \sum_{j=0}^{k} \frac{d}{dt} \prod_{i=0}^{j-1} (t - t_{n-i})\ y[t_n, \ldots, t_{n-j}] \Bigg|_{t=t_{n+1}} \\ &= \sum_{j=0}^{k} \sum_{l=0}^{j-1} \prod_{i=0, i\neq j}^{j-1} (t - t_{n-i})\ y[t_n, \ldots, t_{n-j}] \Bigg|_{t=t_{n+1}} \\ &= \sum_{j=0}^{k} \sum_{l=0}^{j-1} \frac{1}{\psi_{l+1}(n+1)} \prod_{i=0}^{j-1} \psi_{i+1}(n+1)\ y[t_n, \ldots, t_{n-j}] \\ &= \sum_{j=0}^{k} \gamma_{j+1}(n+1) \Phi^*_{j+1}(n) \\ &= \sum_{j=1}^{k+1} \gamma_j(n+1) \Phi^*_j(n). \qquad (5.12) \end{aligned}$$

Lemma 5.1.4 (Darstellung des Korrektors)
Sowohl Prädiktor- als auch Korrektorpolynom interpolieren die k Werte $y_n, \ldots, y_{n-k+1}$ korrekt. Damit können wir ihre Differenz schreiben als

$$\mathcal{P}^C(t) - \mathcal{P}^P(t) = \Delta(t)(y^C_{n+1} - y^P_{n+1}). \tag{5.13}$$

Dabei ist $\Delta(t)$ als Differenz zweier Polynome vom Grad k selbst ein Polynom vom Grad $\leq k$, das durch die $(k+1)$ Bedingungen

$$\Delta(t_{n+1-i}) = \begin{cases} 1 & \text{für } i = 0 \\ 0 & \text{für } i = 1, \ldots, k \end{cases} \tag{5.14}$$

eindeutig festgelegt ist. Wir erhalten also

$$\Delta(t) = \prod_{j=1}^{k} \frac{t - t_{n+1-j}}{t_{n+1} - t_{n+1-j}} \tag{5.15}$$

und durch Differenzieren für die Ableitung an der Stelle t_{n+1}

$$\begin{aligned} \dot{\Delta}(t) &= \frac{d}{dt} \prod_{j=1}^{k} \frac{t - t_{n+1-j}}{t_{n+1} - t_{n+1-j}} \Bigg|_{t=t_{n+1}} \\ &= \sum_{j=1}^{k} \frac{1}{t_{n+1} - t_{n+1-j}} \prod_{i=1, i\neq j}^{k} \frac{t - t_{n+1-j}}{t_{n+1} - t_{n+1-j}} \Bigg|_{t=t_{n+1}} \\ &= \sum_{j=1}^{k} \frac{1}{t_{n+1} - t_{n+1-j}} \\ &= \gamma_{k+1}(n+1). \end{aligned} \tag{5.16}$$

Durch Differentiation von (5.13) erhalten wir schließlich bei t_{n+1} mit (5.12)

und (5.11)

$$\begin{aligned}
\dot{y}^C_{n+1} &= \dot{\mathcal{P}}^C(t_{n+1}) \\
&= \dot{\mathcal{P}}^P(t_{n+1}) + \dot{\Delta}(t_{n+1})(y^C_{n+1} - y^P_{n+1}) \\
&= \sum_{j=1}^{k+1} \gamma_j(n+1)\Phi^*_j(n) + \gamma_{k+1}(n+1)(y^C_{n+1} - \sum_{j=1}^{k+1} \Phi^*_j(n)) \\
&= \sum_{j=1}^{k+1} (\gamma_j(n+1) - \gamma_{k+1}(n+1))\Phi^*_j(n) + \gamma_{k+1}(n+1)y^C_{n+1} \\
&= -\sum_{j=1}^{k} \frac{1}{\psi_j(n+1)} \sum_{i=1}^{j} \Phi^*_i(n) + \gamma_{k+1}(n+1)y^C_{n+1} \\
&= -\sum_{j=1}^{k} \frac{1}{\psi_j(n+1)} \delta_j(n+1) + \gamma_{k+1}(n+1)y^C_{n+1}. \qquad (5.17)
\end{aligned}$$

Dabei bezeichnen wir den Term

$$cc(y)_{n+1} := -\sum_{j=1}^{k} \frac{1}{\psi_j(n+1)} \delta_j(n+1) \qquad (5.18)$$

als *Korrektor-Konstante.*

Lemma 5.1.5 (Update der modifizierten dividierten Differenzen)
Seien y_n und $\delta_i(n)$ gegeben, dann erhält man $\Phi_{i+1}(n)$ als

$$\begin{aligned}
\Phi_{i+1}(n) &= \psi_1(n) \dots \psi_i(n) y[t_n, \dots, t_{n-i}] \\
&= \psi_1(n) \dots \psi_{i-1}(n) \frac{\psi_i(n)}{t_n - t_{n-i}} (y[t_n, \dots, t_{n-i+1}] - y[t_{n-1}, \dots, t_{n-i}]) \\
&= \psi_1(n) \dots \psi_{i-1}(n) (y[t_n, \dots, t_{n-i+1}] - y[t_{n-1}, \dots, t_{n-i}]) \\
&= \Phi_i(n) - \Phi^*_i(n-1) \\
&= y_n - \delta_i(n). \qquad (5.19)
\end{aligned}$$

Damit erhalten wir für $\Phi^*_i(n)$

$$\Phi^*_i(n) = \beta_i(n+1)(y_n - \delta_{i-1}(n)), \qquad (5.20)$$

und können die im Schritt n benötigten Terme $\Phi^*_i(n)$ mit Hilfe der $\delta_{i-1}(n)$ aus dem vorherigen Schritt $(n-1)$ und des letzten Wertes y_n berechnen.

Implementierung in DAESOL-II

Die Berechnung von Prädiktor und Korrektor im Schritt $(n+1)$, ausgehend von Schritt n, ist in DAESOL-II folgendermaßen implementiert.
Seien aus Schritt n die Größen $\psi_i(n)$, $\delta_i(n)$ und y_n gegeben. Dann berechnen wir in folgender Reihenfolge:

a) Die $\psi_i(n+1)$ nach (5.4)

$$\psi_i(n+1) = \psi_{i-1}(n) + h_n,\ i = k, \ldots, 2, \qquad \psi_1(n+1) = h_n,$$

b) die $\beta_i(n+1)$ nach (5.5)

$$\beta_1(n+1) = 1, \qquad \beta_i(n+1) = \beta_{i-1}(n+1)\frac{\psi_{i-1}(n+1)}{\psi_{i-1}(n)},\ i = 2, \ldots, k,$$

c) $\gamma_{k+1}(n+1)$ nach (5.8)

$$\gamma_{k+1}(n+1) = \sum_{i=1}^{k} \frac{1}{\psi_i(n+1)},$$

d) den neuen Prädiktor y^P_{n+1} nach (5.11)

$$\begin{aligned} y^P_{n+1} &= \sum_{j=1}^{k+1} \Phi^*_j(n) \\ &= y_n + \sum_{j=1}^{k} \Phi^*_{j+1}(n) \\ &= y_n + \sum_{j=1}^{k} \beta_{j+1}(n+1)[y_n - \delta_j(n)] \ \text{ nach (5.20)}, \end{aligned}$$

e) sowie die neue Korrektor-Konstante

$$cc(y)_{n+1} = -\sum_{j=1}^{k} \frac{1}{\psi_j(n+1)}\delta_j(n+1).$$

Damit kann der Korrektor dargestellt werden als

$$\dot{y}^C_{n+1} = \gamma_{k+1}(n+1)y^C_{n+1} + cc(y)_{n+1}$$

und für den Koeffizienten α_k des BDF-Verfahrens gilt

$$\alpha_k = h_n\gamma_{k+1}(n+1). \tag{5.21}$$

Bemerkung 5.1.6
Während der Integration brauchen nur die jeweils letzten Werte von $\psi_i(n)$, $\delta_i(n)$ und y_n gespeichert zu werden.
Für die δ_i geschieht dies jeweils während der Summation in Punkt d), so dass diese nicht noch einmal gesondert berechnet werden müssen.

5.2 Fehlerschätzung

Ziel der Schrittweiten- und Ordnungsstrategie eines effizienten adaptiven numerischen Verfahrens ist es, die Schrittweite und Ordnung während der Integration so an das Problem anzupassen, dass die Integration mit minimalem Aufwand durchgeführt wird, wobei der Fehler jeweils unter einer meist vom Benutzer vorgegebenen Toleranz TOL bleiben soll. Dazu ist es offensichtlich nötig, in jedem Schritt den Fehler des Verfahrens zu schätzen und dann dementsprechend den Schritt zu akzeptieren oder zurückzuweisen und mit angepaßter Schrittweite zu wiederholen.
Wünschenswert wäre dazu eine Möglichkeit zur Berechnung des globalen Fehlers des Verfahrens. Leider läßt sich dies in der Praxis nicht einfach bewerkstelligen: Der globale Fehler akkumuliert sich aus den in den einzelnen Schritten gemachten lokalen Fehlern und der Fehlerfortpflanzung, wobei insbesondere die Fehlerfortpflanzung quantitativ nicht zugänglich ist. Eine Approximation des globalen Fehlers ϵ im Schritt $(n+1)$ findet man beispielsweise bei Bauer [Bau99] mit der Formel

$$\begin{pmatrix} \alpha_k A + A_x \dot{x}^C_{n+1} + h f_x & A_z \dot{x}^C_{n+1} + h f_z \\ g_x & g_z \end{pmatrix} \begin{pmatrix} \epsilon^x_{n+1} \\ \epsilon^z_{n+1} \end{pmatrix} = \begin{pmatrix} h\delta_1 - A(h\sigma_{n+1} - hcc(\epsilon^x)) \\ \delta_2 \end{pmatrix}, \tag{5.22}$$

wobei in δ_1 und δ_2 der Fehler durch vorzeitigen Abbruch des iterativen Verfahrens zur Lösung der nichtlinearen Gleichungssysteme innerhalb des BDF-Verfahrens und höhere Ordnungen der Fehler ϵ^x, ϵ^z und σ zusammengefasst sind. Diese Größen sind allerdings ebenso wie die Korrektor-Konstante $cc(\epsilon^x)$ nicht direkt berechenbar.
Daher begnügen wir uns für die Fehlerkontrolle mit einer Schätzung des lokalen Fehlers, basierend auf dem lokalen Diskretisierungsfehler, wie man sie beispielsweise bei [Bau99] oder [Eic91] findet.

Schätzung des lokalen Fehlers

Nach Definition 3.1.7 und der Korrektorformel (5.17) erhalten wir im Schritt $(n+1)$ den lokalen Diskretisierungsfehler für ein k-Schritt BDF-Verfahren durch

$$\begin{aligned}\sigma_{n+1} := \sigma(y(t_{n+1}), h_n) &= \dot{y}(t_{n+1}) - \dot{y}^C_{n+1} \\ &= \dot{y}(t_{n+1}) - \dot{\mathcal{P}}^{P,ex}(t_{n+1}) \\ &\quad -\gamma_{k+1}(n+1)\left(y(t_{n+1}) - \mathcal{P}^{P,ex}(t_{n+1})\right),\end{aligned}$$

wobei $y(t)$ hier die exakte Lösung der DAE beschreibt und $\mathcal{P}^{P,ex}$ das mittels der exakten Werte $y(t_n), \ldots, y(t_{n-k})$ berechnete Prädiktorpolynom im Schritt $(n+1)$ bezeichnet. Analog werden wir die Bezeichungen $y^{ex}[\ldots]$ und Φ^{ex} verwenden.
Aufgrund der Interpolationseigenschaften von $\mathcal{P}^{P,ex}$ ergibt sich:

$$y(t) - \mathcal{P}^{P,ex}(t_{n+1}) = \prod_{i=0}^{k}(t - t_{n-i})\; y^{ex}[t, t_n, \ldots, t_{n-k}]. \tag{5.23}$$

Daraus folgt durch Differenzieren:

$$\begin{aligned}\dot{y}(t) - \dot{\mathcal{P}}^{P,ex}(t_{n+1}) &= \frac{d}{dt}\left(\prod_{i=0}^{k}(t - t_{n-i})\right)\; y^{ex}[t, t_n, \ldots, t_{n-k}] \\ &\quad + \prod_{i=0}^{k}(t - t_{n-i})\; \frac{d}{dt} y^{ex}[t, t_n, \ldots, t_{n-k}].\end{aligned}$$

Es gilt

$$\frac{d}{dt} y^{ex}[t, t_n, \ldots, t_{n-k}] = y^{ex}[t, t, t_n, \ldots, t_{n-k}],$$

wobei $y^{ex}[t,t] := \dot{y}^{ex}(t)$ definiert wird.
Ferner ist

$$\begin{aligned}\prod_{i=0}^{k}(t_{n+1} - t_{n-i}) &= \prod_{i=1}^{k+1}\psi_i(n+1) \text{ und} \\ \frac{d}{dt}\prod_{i=0}^{k}(t_{n+1} - t_{n-i}) &= \sum_{j=1}^{k+1}\frac{1}{\psi_j(n+1)}\prod_{i=0}^{k}(t_{n+1} - t_{n-i}) \\ &= \gamma_{k+2}\prod_{i=1}^{k+1}\psi_i(n+1).\end{aligned}$$

Und damit erhalten wir für σ_{n+1}

$$\begin{aligned}
\sigma_{n+1} &= \gamma_{k+2}(n+1)\prod_{i=1}^{k+1}\psi_i(n+1)\; y^{ex}[t_{n+1},\ldots,t_{n-k}] \\
&\quad + \prod_{i=1}^{k+1}\psi_i(n+1)\; y^{ex}[t_{n+1},t_{n+1},\ldots,t_{n-k}] \\
&\quad -\gamma_{k+1}(n+1)\prod_{i=1}^{k+1}\psi_i(n+1)\; y^{ex}[t_{n+1},\ldots,t_{n-k}] \\
&= (\gamma_{k+2}(n+1)-\gamma_{k+1}(n+1))\prod_{i=1}^{k+1}\psi_i(n+1)\; y^{ex}[t_{n+1},\ldots,t_{n-k}] \\
&\quad + \prod_{i=1}^{k+1}\psi_i(n+1)\; y^{ex}[t_{n+1},t_{n+1},\ldots,t_{n-k}] \qquad (5.24) \\
&= \left(\sum_{i=1}^{k+1}\frac{1}{\psi_i(n+1)} - \sum_{i=1}^{k}\frac{1}{\psi_i(n+1)}\right)\Phi^{ex}_{k+2}(n+1) \\
&\quad + \prod_{i=1}^{k+1}\psi_i(n+1)\; y^{ex}[t_{n+1},t_{n+1},\ldots,t_{n-k}] \\
&= \frac{1}{\psi_{k+1}(n+1)}\Phi^{ex}_{k+2}(n+1) + \prod_{i=1}^{k+1}\psi_i(n+1)\; y^{ex}[t_{n+1},t_{n+1},\ldots,t_{n-k}].
\end{aligned}$$

Die mit ex indizierten Terme wie Φ^{ex}, die hier eigentlich mit der exakten Lösung berechnet wurden, werden in der Praxis durch die entsprechenden Terme aus der numerischen Berechnung der Lösung ersetzt, da die exakte Lösung ja nicht zugänglich ist. Gear hat gezeigt, dass diese Schätzung für äquidistante Gitter asymptotisch korrekt ist [Gea74].
Analog zur Approximation für den globalen Fehler (5.22) kann man nun folgende Approximation des lokalen Fehlers μ, basierend auf dem lokalen Diskretisierungsfehler, herleiten

$$\begin{pmatrix} \alpha_k A + A_x\dot{x}^C_{n+1} + hf_x & A_z\dot{x}^C_{n+1} + hf_z \\ g_x & g_z \end{pmatrix}\begin{pmatrix} \mu^x_{n+1} \\ \mu^z_{n+1}\end{pmatrix} = \begin{pmatrix} -hA\sigma^x_{n+1} \\ 0 \end{pmatrix}. \qquad (5.25)$$

Implementierung der Fehlerschätzung in DAESOL-II

Eine Berechnung des lokalen Fehlers (und im Zuge der Schrittweitensteuerung einer neuen Schrittweite) aus (5.25) würde aber sehr aufwendig sein, da

wir in jedem Schritt das Gleichungssystem lösen müßten. Daher verwenden wir in DAESOL-II die folgende vereinfachte Fehlerschätzung

$$\begin{aligned} E_k(n+1,h_n) &= h_n||\sigma_{n+1}|| \\ &= h_n\Big|\Big|\frac{1}{\psi_{k+1}(n+1)}\Phi_{k+2}(n+1) \\ &\quad + \prod_{i=1}^{k+1}\psi_i(n+1)\; y[t_{n+1},t_{n+1},\ldots,t_{n-k}]\Big|\Big| \\ &\doteq h_n\frac{1}{\psi_{k+1}(n+1)}\Big|\Big|\prod_{i=1}^{k+1}\psi_i(n+1)\; y[t_{n+1},\ldots,t_{n-k}]\Big|\Big| \\ &= h_n\prod_{i=1}^{k}\psi_i(n+1)\;\Big|\Big|y[t_{n+1},\ldots,t_{n-k}]\Big|\Big|. \end{aligned} \tag{5.26}$$

Nach jedem Schritt wird getestet, ob diese Schätzung für $E_k(n+1,h_n)$ unter der vom Benutzer vorgegeben relativen Toleranz TOL liegt. Ist dies der Fall, so wird der Schritt akzeptiert. Andernfalls wird der Schritt zurückgewiesen und mit verringerter Schrittweite wiederholt. Die Details zur Reduktion der Schrittweite werden im Abschnitt 5.4 näher diskutiert.

5.3 Lösung der nichtlinearen Gleichungssysteme

Da die BDF-Verfahren implizite Verfahren sind, müssen wir, wie oben festgestellt, in jedem Schritt ein nichtlineares Gleichungssystem lösen. Dies geschieht aufgrund der Steifheit der behandelten Probleme in der ersten Version von DAESOL-II mittels eines Newton-ähnlichen Verfahrens unter Anwendung einer *Monitor-Strategie*, welche im Folgenden näher erläutert wird. Aufgrund des modularen Aufbaus von DAESOL-II ist es allerdings später auch möglich, andere, mehr an das jeweils behandelte Problem adaptierte, Verfahren zur Lösung der Gleichungssysteme mit begrenztem Aufwand einzubinden.
Wir schreiben das Diskretisierungsschema (3.12) des BDF-Verfahrens für eine linear implizite DAE als

$$F(y_{n+1}) = 0, \quad \text{wobei wiederum } y_{n+1} = (x_{n+1}^T, z_{n+1}^T)^T. \tag{5.27}$$

Das Newton-Verfahren zur iterativen Lösung des Gleichungssystems im Schritt $(n+1)$ ist dann gegeben durch einen Startwert $y_{n+1}^{(0)}$ und die Itera-

tionsvorschrift

$$y_{n+1}^{(i+1)} = y_{n+1}^{(i)} + \Delta y_{n+1}^{(i)},$$

wobei das Inkrement $\Delta y_{n+1}^{(i)}$ berechnet wird als Lösung des linearen Gleichungssystems

$$J(y_{n+1}^{(i)}) \cdot \Delta y_{n+1}^{(i)} = -F(y_{n+1}^{(i)}), \tag{5.28}$$

mit $J(y_{n+1}^{(i)}) := \frac{\partial F(y_{n+1}^{(i)})}{\partial y}$ der Jakobi-Matrix von F. Sie hat die Gestalt

$$J(y_{n+1}^{(i)}) = \begin{pmatrix} \alpha_k A + A_x(\alpha_k x_{n+1}^{(i)} + h\,cc) + h\,f_x & A_z(\alpha_k x_{n+1}^{(i)} + h\,cc) + h\,f_z \\ g_x & g_z \end{pmatrix}, \tag{5.29}$$

wobei $cc = cc(x)_{n+1}$ hier die Korrektor-Konstante, also den von y_{n+1} unabhängigen Teil der Ableitung des Korrektorpolynoms $\dot{\mathcal{P}}^C(t_{n+1})$ an der Stelle t_{n+1}, beschreibt. Als Startwert verwenden wir den Wert $\mathcal{P}^P(t_{n+1})$ des Prädiktorpolynoms an der Stelle t_{n+1}.

Bei der Lösung der Gleichungssysteme mit dem Newton-Verfahren muß also in jedem Iterationsschritt die Jakobi-Matrix aufgestellt und zerlegt werden. Normalerweise wird hierzu der größte Teil der Rechenzeit während der Integration benötigt. Dies ist insbesondere der Fall für große DAE-Systeme bzw. wenn die Auswertungen der Ableitungen der Modellfunktionen f, g und A sehr kostspielig zu berechnen sind.

Oft ändert sich jedoch die Jakobi-Matrix von einer Iteration zur nächsten bzw. sogar über mehrere Integrationsschritte des BDF-Verfahrens nur sehr wenig. Daher bietet es sich an, die Jakobi-Matrix so lange wie möglich konstant zu halten und wiederzuverwenden, um den Rechenaufwand zu verringern. Das Newton-ähnliche Verfahren besitzt leicht schlechtere Konvergenzeigenschaften als das reine Newton-Verfahren, was einige zusätzliche Iterationen notwendig macht. Dieser zusätzliche Aufwand fällt aber i.A. deutlich geringer aus als die Einsparungen durch weniger Auswertungen und Zerlegungen der Jakobi-Matrix.

Wir stellen nun eine Monitor-Strategie für die Iterationsmatrizen vor, die gewährleistet, dass das Newton-ähnliche Verfahren mit wenigen Iterationsschritten konvergiert, und dabei die Jakobi-Matrix und die Ableitungen der Modellfunktionen so lange wie möglich wiederverwendet.

Die Monitor-Strategie geht zurück auf Bock und Eich und findet sich beispielsweise in [Eic87].

Als Grundlage der Strategie betrachten wir zuerst das Konvergenzverhalten von Newton-ähnlichen Verfahren, für das Bock [Boc87] folgenden Satz bewiesen hat.

Satz 5.3.1 (Lokaler Kontraktionssatz)
Sei $D \subseteq \mathbb{R}^{n_y}$ offen und $F : D \longrightarrow \mathbb{R}^{n_y}$ eine C^1-Funktion. Wir bezeichnen mit $J = \frac{\partial F}{\partial y}$ die Jakobi-Matrix von F und mit M eine Approximation der Inversen von J.
Für alle $\tau \in [0,1]$, für alle $y, y + \Delta y \in D$ mit $\Delta y = -MF(y)$ und für alle m existieren ω und κ, so dass:

1. Eine verallgemeinerte Lipschitz-Bedingung gelte:
$$\frac{\|M(J(y^m + \tau\Delta y^m) - J(y^m))\Delta y^m\|}{\tau\|\Delta y^m\|^2} \leq \omega^m, \;\; \omega^m \leq \omega \leq \infty, \quad (5.30)$$

2. die Güte der Näherungsinversen M in Richtung der Newton-Inkremente ausreichend ist:
$$\frac{\|M(F(y^m) - J(y^m)\Delta y^m)\|}{\|\Delta y^m\|} \leq \kappa^m, \;\; \kappa^m \leq \kappa < 1, \quad (5.31)$$

3. der Startwert y^0 der Iteration die folgende Bedingung erfülle
$$\delta_0 := \frac{\omega^0}{2}\|\Delta y^0\| + \kappa^0 < 1, \quad (5.32)$$

4. und die Kugel $D_0 := S_r(y^0)$ um y^0 mit Radius $r = \frac{\|\Delta y^0\|}{1-\delta_0}$ in D liege.

Dann gilt:

1. Die Iteration $y^{m+1} = y^m + \Delta y^m$, $\Delta y^m = -MF(y^m)$ ist wohldefiniert und bleibt in D_0,

2. es existiert eine Nullstelle $y^* \in D_0$, gegen die y^m konvergiert,

3. die Folge y^m konvergiert mindestens linear mit
$$\|\Delta y^m\| \leq \left(\frac{\omega^{m-1}}{2}\|\Delta y^{m-1}\| + \kappa^{m-1}\right)\|\Delta y^{m-1}\| \leq \|\Delta y^{m-1}\|$$

4. und es gilt die apriori-Abschätzung
$$\|y^m - y^*\| \leq \frac{\delta_0^m}{1-\delta_0}\|\Delta y^0\|.$$

Mittels einer Schätzung der Konvergenzrate δ_0 können wir dann die Konvergenz des Newton-ähnlichen Verfahrens kontrollieren. Wir erhalten diese nach der Durchführung von zwei Newton-Iterationen durch

$$\tilde{\delta}_0 := \frac{\|\Delta y^{(1)}\|}{\|\Delta y^{(0)}\|} \leq \delta_0.$$

Als Abbruchkriterium für das Newton-ähnliche Verfahren verwenden wir die Norm des Inkrementes $\|\Delta y^i\|$. Wenn $\|\Delta y^i\| \leq c_{newton} \cdot TOL$, mit $c_{newton} < 1$ und TOL der vom Benutzer vorgegebenen relativen Toleranz, so brechen wir das Verfahren ab.
Da der Startwert $y^{(0)} = \mathcal{P}^P(t_{n+1})$ normalerweise in der Nähe des Lösungspunktes y^* liegt, sollte er sich im lokalen Konvergenzbereich des Newton-ähnlichen Verfahrens befinden. Außer beispielsweise bei sehr nichtlinearen Problemen ist es hier also weniger das Ziel, überhaupt einen Lösungspunkt zu finden, sondern vielmehr ihn mit möglichst geringem Aufwand pro Integrationsschritt zu berechnen. Der Aufwand setzt sich hierbei aus der Berechnung und Zerlegung der Jakobi-Matrizen und dem Lösen der linearen Gleichungssysteme im Newton-ähnlichen Verfahren mit bereits zerlegter Matrix zusammen.
Um letzteren Anteil zu begrenzen, fordern wir eine so gute Konvergenzrate, dass nach maximal 3 Iterationen das Abbruchkriterium erfüllt wird. (Wir benötigen auf jeden Fall 2 Iterationen zur Schätzung der Konvergenzrate und lassen noch eine dritte zu, wenn die Schätzung der Konvergenzrate eine Erfüllung des Abbruchkriteriums nach 3 Schritten voraussagt.)
Eine Übersicht zur Wahl der Bedingungen an $\tilde{\delta}_0$ in Abhängigkeit der Anzahl der Iterationen m und dem geforderten Verbesserungsfaktor c_{red} nach m Iterationen in der Abschätzung $\|x^m - x^*\| \leq c_{red}\|x^0 - x^*\|$ gibt von Schwerin [vS97]. Wir verwenden eine Wahl von $m = 3$ Iterationen und $c_{red} = \frac{1}{12}$ und erhalten damit $\delta_0 \leq 0.3$.
Wir testen also nach 2 Iterationen neben dem Abbruchkriterium, ob für die geschätzte Konvergenzrate $\tilde{\delta}_0 < 0.3$ gilt. Falls $\tilde{\delta}_0 \geq 0.3$ ist, d.h. die Konvergenzrate des Newton-ähnlichen Verfahrens zu schlecht ist, kann dies verschiedene Ursachen haben:

a) Die Güte der Iterationsmatrix ist nicht gut genug, d.h. κ im lokalen Kontraktionssatz ist zu groß:

- Der Koeffizient α_k des BDF-Verfahrens oder die Schrittweite h hat sich stark verändert, z.B. bei Änderung der Ordnung oder der Schrittweite.
- Die Ableitungen der Modellfunktionen f, g oder A haben sich wesentlich verändert.

b) Der Startwert $y^{(0)}$ ist zu weit vom Lösungspunkt entfernt, das Problem ist "zu nichtlinear", d.h. ω im lokalen Kontraktionssatz ist zu groß und der Startwert liegt nicht mehr im lokalen Konvergenzbereich des Verfahrens.

Implementierung der Monitor-Strategie in DAESOL-II

Von obigen Überlegungen geleitet verwenden wir in DAESOL-II folgende Strategie zur Steuerung der Newton-Iterationen mittels der Schätzung der Konvergenzrate.

a) In jedem Iterationsschritt wird das Abbruchkriterium getestet. Falls es erfüllt ist, wird die Iteration als erfolgreich abgebrochen.

b) Nach zwei Iterationen wird die Konvergenzrate geschätzt. Liegt diese unterhalb der geforderten Schranke von 0.3, so wird eine dritte Iteration zugelassen, andernfalls die Iteration als fehlgeschlagen abgebrochen.

c) Falls nach 3 Iterationen das Abbruchkriterium nicht erfüllt ist, wird die Iteration ebenfalls als fehlgeschlagen abgebrochen.

Die Wiederverwendung bzw. Neuberechnung und Zerlegung von Ableitungen und der Iterationsmatrix erfolgt nach folgendem Schema:

1. Solange für die Schätzung der Konvergenzrate $\tilde{\delta}_0 < 0.3$ gilt, wird die Jakobi-Matrix konstant gehalten und ihre Zerlegung weiterverwendet.

2. Falls keine Konvergenz erzielt wird, so wird die Iterationsmatrix mit den aktuellen Werten für α_k und h neu zerlegt, dabei aber die alten Ableitungen f_y, g_y und A_y verwendet. Dann werden die Iterationen wiederholt.

3. Falls wiederum keine Konvergenz erzielt wird, werden auch die Ableitungen f_y, g_y und A_y neu berechnet und die Iterationsmatrix erneut zerlegt. Die Iterationen werden wiederholt.

4. Falls dennoch keine Konvergenz erzielt wird, so wird der komplette BDF-Schritt mit verkleinerter Schrittweite wiederholt. Die Details der Schrittweitenreduktion werden in Abschnitt 5.4 näher erläutert.

Dabei ist insbesondere Punkt 2 eine Maßnahme, die in vielen anderen Integratoren nicht verwendet wird. Wenn sich allerdings die Ableitungen der Modellfunktion nur langsam ändern, und sie dazu noch aufwendig auszuwerten sind, bietet sich hier viel Einsparpotential [Eic87].

5.4 Schrittweiten- und Ordnungssteuerung

In diesem Abschnitt beschreiben wir die Strategie zur Schrittweiten- und Ordnungssteuerung, die wir in DAESOL-II verwenden.
Ziel dieser Strategie ist es, den nächsten Schritt so zu planen, dass die neue Schrittweite möglichst groß ist, aber der entstehende lokale Fehler unter der Toleranz TOL bleibt, so dass der Schritt akzeptiert wird.

Schätzung von h_{n+1}

Nehmen wir im Folgenden an, der Schritt $(n+1)$ sei akzeptiert worden. Um eine neue Schrittweite h_{n+1} für den nächsten Schritt zu bestimmen, benötigen wir zunächst eine Schätzung für den lokalen Fehler im Schritt $(n+2)$ für die Ordnung k, in Abhängigkeit der neuen Schrittweite h_{n+1}.
Ausgehend von der Formel (5.24) für den lokalen Diskretisierungsfehler σ_{n+1} erhalten wir für σ_{n+2}

$$\begin{aligned}\sigma_{n+2} &= \prod_{i=1}^{k} \psi_i(n+2)\; y^{ex}[t_{n+2}, \ldots, t_{n-k+1}] \\ &\quad + \prod_{i=1}^{k+1} \psi_i(n+2)\; y^{ex}[t_{n+2}, t_{n+2}, \ldots, t_{n-k+1}],\end{aligned}$$

und daraus analog die Fehlerschätzung für den lokalen Fehler

$$\begin{aligned} E_k(n+2,h_{n+1}) &= h_{n+1}\|\sigma_{n+1}\| \\ &\doteq h_{n+1}\prod_{i=1}^{k}\psi_i(n+2)\Big\|y[t_{n+2},\ldots,t_{n-k+1}]\Big\|. \end{aligned} \tag{5.33}$$

Hierbei läßt sich die noch unbekannte dividierte Differenz (konservativ) schätzen durch [BSS95]:

$$\begin{aligned} \Big\|y[t_{n+2},\ldots,t_{n-k+1}]\Big\| &\leq \Big\|y[t_{n+1},\ldots,t_{n-k}]\Big\| \\ &\quad +\psi_{k+2}(n+2)\Big\|y[t_{n+1},\ldots,t_{n-k-1}]\Big\|. \end{aligned} \tag{5.34}$$

Damit erhalten wir die folgende Bedingung für h_{n+1} :

$$\begin{aligned} E_k(n+2,h_{n+1}) &\doteq h_{n+1}\prod_{i=1}^{k}\psi_i(n+2)\cdot\Big(\Big\|y[t_{n+1},\ldots,t_{n-k}]\Big\| \\ &\qquad +\psi_{k+2}(n+2)\Big\|y[t_{n+1},\ldots,t_{n-k-1}]\Big\|\Big) \\ &\leq TOL' := c\,TOL, \end{aligned} \tag{5.35}$$

wobei hier $c \leq 1$ ein Sicherheitsfaktor ist, z.B. $c = 0.5$.
Aus dieser in h_{n+1} polynomiellen Ungleichung läßt sich nur schwer ein passendes h_{n+1} ermitteln. Daher berechnen wir zunächst als Näherung die maximale zulässige Schrittweite auf einem äquidistanten Gitter und reduzieren sie gegebenenfalls mittels obiger Ungleichung.
Auf äquidistantem Gitter erhalten wir für die Ordnung k die Schätzung

$$\hat{E}_k(n+2,h) = k!\;h^{k+1}\|y[t_{n+1},\ldots,t_{n-k}]\|,$$

und damit aus der Forderung $\hat{E}_k(n+2,h) \leq TOL'$ die sogenannte *maximale gleichmäßige Schrittweite* [Ble86]:

$$\hat{h}_{\hat{k}} = \sqrt[\hat{k}+1]{\frac{TOL'}{\hat{k}!\;\|y[t_{n+1},\ldots,t_{n-\hat{k}}]\|}}. \tag{5.36}$$

Wir berechnen $\hat{h}_{\hat{k}}$ für die aktuelle Ordnung $\hat{k} = k$ der Integration, sowie für $\hat{k} = k-1$ und $\hat{k} = k+1$. Wir vergleichen diese drei Werte und ändern die Ordnung im nächsten Schritt, wenn sich für die neue Ordnung eine signifikant größere Schrittweite ergibt. Auf diese Weise erhalten wir eine neue Ordnung

k^* für den nächsten Schritt.
Schließlich testen wir noch, ob $\hat{h}_{k^*}$ die Formel für die Fehlerschätzung

$$E_k(n+2, \hat{h}_{k^*}) \leq TOL'$$

auf variablem Gitter erfüllt. Ist dies der Fall, so wird die Schrittweite akzeptiert und der nächste Schritt mit $h_{n+1} = \hat{h}_{\hat{k}}$ geplant. Andernfalls wird die Schrittweite mit Hilfe von (5.35) angepaßt. Dazu benutzen wir die Formel

$$h_{k^*}^2 = \frac{TOL'}{q(\hat{h}_{k^*}) \, (\|y[t_{n+1},\ldots,t_{n-k^*}]\| + (\hat{h}_{k^*} + \psi_{k+1}(n+1))\|y[t_{n+1},\ldots,t_{n-k-1}]\|)}, \qquad (5.37)$$

wobei

$$q(h) := \prod_{i=1}^{k^*-1} (h + \psi_i(n+1))$$

monoton wachsend in h ist.
Da $h_{k^*} < \hat{h}_{k^*}$ ist, folgt $h_{k^*}^2 q(h_{k^*}) < h_{k^*}^2 q(\hat{h}_{k^*})$ und h_{k^*} erfüllt dann die Ungleichung

$$E_k(n+2, h_{k^*}) \leq TOL'$$

und wird daher als Schrittweite für den neuen Schritt verwendet.
Die vorgestellte Strategie, die die Ordnungs- und Schrittweitensteuerung auf Basis variabler Gitter durchführt, erlaubt eine flexible Anpassung von Schrittweiten und Ordnungen. Sie steigert, wie Bleser [Ble86] und später Eich [Eic91] demonstriert haben, im Vergleich zu konventionellen Schrittweitenstrategien, die auf äquidistanten Gittern basieren, sowohl die Effizienz durch weniger Schrittzurückweisungen und ein relativ schnelles Hochschalten der Ordnung, als auch die Stabilität.

Bemerkung 5.4.1 (Schrittweitenänderungen und Stabilität)
Wie in Abschnitt 3.2.2 gezeigt ist die Stabilität von BDF-Verfahren auf variablen Gittern ohne weitere Annahmen nur für sehr begrenzte Schrittweitenänderungen bewiesen, die besonders bei höheren Ordnungen eine effiziente Schrittweitensteuerung nicht zulassen. Es gibt eine Reihe von Untersuchungen, die unter bestimmten Annahmen großzügigere Schranken zulassen. Calvo et. al. [CLM87] berechneten beispielsweise für BDF-Verfahren auf pseudo-äquidistantem Gitter (Nordsieck-Verfahren, [Nor62]) Schranken für die Schrittweitenänderungen, die die Stabiliät garantieren. Dabei wird im

Anschluß an eine Schrittweitenänderung die Schrittweite konstant gehalten. Die Schranken werden dabei umso großzügiger, je länger die Schrittweite konstant gehalten wird.
Gear und Tu [GT74] bewiesen unter der Annahme stetiger Schrittweitenänderungen weniger strenge Schranken an die Schrittweitenänderungen als Griegorieff in [Gri83].
Die in DAESOL-II verwendete Schrittweitensteuerung zielt auf stetige Schrittweitenänderungen ab und vermeidet allzu starke Sprünge in den Schrittweiten. Dabei werden Schrittweitenänderungen in jeden Schritt gestattet. Die engen Schranken aus Theorem 3.3.5 müssen daher nicht eingehalten werden.

Bemerkung 5.4.2 (Ordnungsänderungen und Stabilität)
Analog zu Schrittweitenänderungen können auch Änderungen der Ordnung, was ja streng genommen ein Wechsel der verwendeten LMM bedeutet, Probleme bei der Stabilität mit sich bringen. In DAESOL-II wird dem vorgebeugt, indem das in den Untersuchungen von Gear und Wanatabe [GW74] geforderte Einfrieren der Ordnung nach einem Ordnungswechsel eingehalten wird. Dieses Konstanthalten der Ordnung dauert je nach Ausgangsordnung und Erhöhung bzw. Reduzierung der Ordnung zwischen einem und vier Schritten.

Verhalten bei Schrittzurückweisungen

Die obigen Überlegungen dienten der Schätzung einer neuen Schrittweite für den nächsten Schritt für den Fall, dass der aktuelle Schritt akzeptiert wurde. Ist dies nicht der Fall, stellt sich die Frage, mit welcher Schrittweite der aktuelle Schritt $(n+1)$ wiederholt werden soll.
Dabei unterscheiden wir zwischen den beiden möglichen Ursachen der Schrittzurückweisung in DAESOL-II:

a) Der geschätzte Fehler $E_k(n+1, h_n)$ ist zu groß.

b) Das Newton-ähnliche Verfahren konvergiert trotz Neuberechnung und Zerlegung der Jakobi-Matrix nicht innerhalb von drei Schritten.

Im Fall a), wenn das Newton-ähnliche Verfahren konvergiert, aber die Schätzung des lokalen Fehlers zu groß ist, berechnen wir eine neue Schrittweite zur

Wiederholung des Schrittes aus der Fehlerformel (5.35) auf variablem Gitter mittels

$$h_n^{(neu)} = h_n^{(alt)} \frac{TOL'}{E_k(n+1, h_n^{(alt)})}, \tag{5.38}$$

mit TOL' wie in (5.35).

Im Fall b), wenn das Newton-ähnliche Verfahren trotz Neuberechnung und Zerlegung der Jakobi-Matrix nicht konvergiert, reduzieren wir die Schrittweite implizit über eine Verschärfung der Toleranz.
Dabei soll die neue Schrittweite so gewählt werden, dass für die neue Konvergenzrate $\delta_0^{(neu)} \leq \frac{1}{4}$ gilt. Das entspricht dem Ziel einer Konvergenz des Newton-ähnlichen Verfahrens innerhalb von zwei Schritten [Enk84].
Da die Schrittzurückweisung am Ende der Monitor-Strategie liegt, wurde zuvor die Jakobi-Matrix mit den aktuellen Ableitungen der Modellfunktionen neu berechnet und zerlegt. Daher gilt hier im Gütekriterium für die Iterationsmatrix (5.31) $\kappa = 0$.
Eine Schätzung von ω erhalten wir dann mittels (5.32) als:

$$\omega \approx 2 \frac{\delta_0^{(alt)}}{\|\Delta y_{n+1,alt}^{(0)}\|}. \tag{5.39}$$

Damit läßt sich die Forderung

$$\delta_0^{(neu)} = \frac{\omega}{2} \|\Delta y_{n+1,neu}^{(0)}\| \leq \frac{1}{4}$$

umformen zu

$$\|\Delta y_{n+1,neu}^{(0)}\| \leq \frac{\|\Delta y_{n+1,alt}^{(0)}\|}{4\ \delta_0^{(alt)}}. \tag{5.40}$$

Betrachten wir nun die Schätzung des lokalen Fehlers, so erhalten wir aus

$$\|y_{n+1}^C - y_{n+1}^P\| = \prod_{i=1}^{k} \psi_i(n+1) \|y[t_{n+1}, \ldots, y_{n-k}]\|$$

und der Näherung

$$\|\Delta y_{n+1}^{(0)}\| \approx \|y_{n+1}^C - y_{n+1}^P\|$$

schließlich die Approximation

$$E_k(n+1, h) = \frac{h}{\psi_{k+1}(n+1)} \|\Delta y^{(0)}\|. \tag{5.41}$$

Verschärfen wir also nun TOL zu

$$\overline{TOL} = \frac{h}{\psi_{k+1}(n+1)} \frac{\|\Delta y^{(0)}_{n+1,alt}\|}{4\delta_0^{(alt)}} \tag{5.42}$$

und fordern dementsprechend in der Schrittweitenneubestimmung

$$E_k(n+1, h_n^{(neu)}) \leq \overline{TOL}'$$

so erreichen wir damit

$$\delta_0^{(neu)} \leq \frac{1}{4}.$$

5.5 Start des BDF-Verfahrens

Ein k-Schritt BDF-Verfahren erfordert die Vorgabe von k-Startwerten. Zu Beginn des Integrationsvorgangs zur Lösung eines IVPs steht lediglich der Anfangswert $y_0 = (x_0^T, z_0^T)^T$ zur Verfügung. Nun gibt es mehrere Möglichkeiten, die Integration zu starten.

Bei einem sogenannten *selbst startenden Verfahren* beginnen wir mit einem Ein-Schritt BDF-Verfahren und erhöhen die Ordnung sukzessive gemäß der Schrittweiten und Ordnungssteuerung. Der Nachteil hierbei ist, dass wegen der niedrigen Ordnung zu Beginn meist sehr kleine Schritte gemacht werden müssen, und die Lösung evtl. ungenauer ist. Fehler, die dabei zu Beginn der Integration gemacht werden, können u.U. noch eine ganze Zeit lang den Gesamtfehler des Verfahrens dominieren (siehe dazu z.B. [Bau99]). Außerdem sind im Zuge der sukzessiven Schrittweiten- und Ordnungserhöhung mehrere Neuberechnungen und Zerlegungen der Jakobi-Matrix zu erwarten.

Eine weitere Möglichkeit besteht darin, einen kleinen Rückwärtsschritt mit einem Einschrittverfahren zu machen, um dann mit einem Zwei-Schritt-Verfahren zu starten.

Schließlich gibt es noch die Möglichkeit, ein Einschrittverfahren höherer Ordnung zu benutzen, um einige Startwerte zu berechnen und dann gleich mit einem LMM höherer Ordnung zu starten. Gear [Gea80] verwendete als erster ein Runge-Kutta-Verfahren zum Start eines LMMs. Von Schwerin und Bock [vSB95] verwendeten ein explizites Runge-Kutta-Verfahren (RK) als Starter für ein Adams-Verfahren. Im Unterschied zu Gear sind bei diesem Verfahren mehrere innere Stufen von höherer Ordnung, so dass mit einem Schritt des

RK-Verfahrens die benötigten Startwerte bereitgestellt werden können.
Für BDF-Verfahren entwickelte Bauer [Bau99] ein implizites RK-Verfahren, bei dem analog zu dem Ansatz von Bock und von Schwerin mehrere innere Stufen von höherer Ordnung sind. Damit können mit einem Schritt des RK-Verfahrens die Startwerte für ein BDF-Verfahren 4. Ordnung bereitgestellt werden. Weiterhin ist dieses so konstruiert, dass die Iterationsmatrizen für das RK-Verfahren im BDF-Verfahren wiederverwendet werden können.

In der ersten Version von DAESOL-II ist ein selbst startendes Verfahren implementiert. Da hierbei im ersten Schritt auch nicht genügend Vergangenheitswerte für das Prädiktorpolynom zur Verfügung stehen, verwenden wir zur Prädiktorgewinnung im ersten Schritt ein explizites Euler-Verfahren. Auch hier ist durch den modularen Aufbau eine spätere Erweiterung beispielsweise mit einem der obigen Ansätze möglich.
Dies ist insbesondere im Hinblick auf die effiziente Behandlung von Problemen mit Unstetigkeiten in den Modellfunktionen oder deren niedrigeren Ableitungen von Interesse, da bei jeder Unstetigkeitsstelle die Integration gestoppt und neu gestartet werden muß.

5.6 Generierung von konsistenten Anfangswerten

Um ein Anfangswertproblem für eine DAE zu lösen, werden konsistente Anfangswerte benötigt. Im Anwendungsfall sind diese unter Umständen nicht genau bekannt, weil sie sich beispielsweise einer Messung entziehen. Im vollimpliziten Fall bedeuten konsistente Anfangswerte, dass $F(t_0, y_0, \dot{y}_0) = 0$ gelten muss. Im linear-impliziten Fall mit Index 1 sind die differentiellen Variablen x frei, und die algebraischen Variablen z sind wegen der Regularitätsbedingung an g_z eindeutig durch die differentiellen Variablen bestimmt. Die Konsistenz der Anfangswerte bedeutet hier, dass $g(t_0, x_0, z_0) = 0$ gilt.
Falls die Integration mit inkonsistenten Anfangswerten gestartet würde, würde man nach einem Schritt, falls das iterative Verfahren zur Lösung der nichtlinearen Gleichungssysteme eine Lösung überhaupt findet, einen konsistenten Punkt erhalten. Allerdings würden sich Probleme in der Fehlerschätzung und natürlich der Qualität der berechneten "Lösung" ergeben. Denn im ersten Schritt geht mit $h_0 \to 0$ der Fehler nicht gegen 0, da stets ein nicht

verschwindender Beitrag der Distanz zwischen konsistenten und inkonsistenten Anfangswerten verbleibt.
Bei einem linear impliziten DAE-IVP vom Index 1 bedeutet die Generierung von konsistenten Anfangswerten somit die Suche nach einer Nullstelle der algebraischen Gleichungen $g(t_0, x_0, z)$ in Abhängigkeit von lediglich den algebraischen Variablen z.
Ein einfacher Ansatz dazu, der auch in DAESOL-II implementiert ist, ist ein Vollschritt-Newton-Verfahren. Dieses hat die Schwäche, dass es nur konvergiert, wenn die Startschätzung für die algebraischen Variablen im lokalen Konvergenzbereich (siehe auch Satz 5.3.1) des Newton-Verfahrens liegt.
Bei hochgradig nichtlinearen Problemen ist es daher möglich, dass keine geeigneten Startschätzungen für die algebraischen Variablen gefunden werden können. In diesem Fall würden dann global konvergente Verfahren zur Nullstellensuche benötigt. Dies könnten beispielsweise gedämpfte Newton-Verfahren oder Homotopie-Methoden sein, wie sie in [BSS95] oder [Bau99] beschrieben werden, welche leicht in DAESOL-II integriert werden können.

5.7 Skalierung

Bei der numerischen Lösung von Problemen ist es notwendig, die unterschiedlichen Größenordnungen der einzelnen Komponenten der Variablen zu berücksichtigen. Daher verwenden wir bei den beschriebenen Strategien zur Fehlerschätzung und Schrittweitensteuerung statt der gewöhnlichen l_2-Norm $\|y\|_2$ eine gewichtete Norm, die berechnet wird durch:

$$\|y_{n+1}\| = \frac{1}{n_y}\sqrt{\sum_{i=1}^{n_y}\left(\frac{(y_{n+1})_i}{(v_{scal_{n+1}})_i}\right)^2}. \tag{5.43}$$

Dabei bezeichnet der Index i jeweils die i-te Komponente des Vektors und v_{scal} ist der Skalierungsvektor, der die einzelnen Skalierungsfaktoren zusammenfaßt.

In DAESOL-II sind bisher vier Möglichkeiten zur Skalierung implementiert, die sich in der Wahl und Aktualisierung der Skalierungsvektoren unterscheiden:

1. Die DASSL-Skalierung [BCP96], die in vielen Integratoren verwendet

wird:

$$(v_{scal_{n+1}})_i = |(y_{n+1})_i| + (atol)_i/TOL,\ i = 1, \ldots, n_y.$$

2. Gear-Skalierung:

$$(v_{scal_{n+1}})_i = \max\{|(y_{n+1})_i|, (v_{scal_n})_i\},\ i = 1, \ldots, n_y.$$

3. Alte Deuflhard-Skalierung:

$$(v_{scal_{n+1}})_i = \max\{|(y_{n+1})_i|, (v_{scal_n})_i, CMIN\},\ i = 1, \ldots, n_y,$$

mit $CMIN = \max_{i=1}^{n_y}(y_{n+1})_i \cdot \epsilon_{mech} \cdot \frac{100}{TOL}$.

4. Neue Deuflhard-Skalierung:

$$(v_{scal_{n+1}})_i = \max\{|(y_{n+1})_i|, (v_{scal_n})_i, (atol)_i\},\ i = 1, \ldots, n_y.$$

Dabei gilt $v_{scal_0} = 0$ und ϵ_{mech} steht für die Maschinengenauigkeit, TOL für die von Benutzer vorzugebende relative Toleranz und *atol* einer von Benutzer vorzugebenden absoluten Toleranz, die sowohl als Skalar, wie auch als Vektor angegeben werden kann.
Außer in der DASSL-Skalierung ist der Skalierungsvektor im aktuellen Schritt von den vorherigen Werten abhängig, um den maximalen Wert der einzelnen Komponenten während der Integration zu berücksichtigen. Falls eine Komponente im Laufe der Integration stark abfällt, aber dennoch weiterhin für die Berechnung signifikant ist, ist die DASSL-Skalierung den anderen vorzuziehen.
Verschiedene Größenordnungen der einzelnen Komponenten können durch entsprechende Wahl der Komponenten von *atol* berücksichtigt werden. In der DASSL-Skalierung spielt *atol* die Rolle einer absoluten Fehlertoleranz, in der neuen Deuflhard-Skalierung die eines Skalierungsfaktors.

5.8 Berechnung der Sensitivitäten

Zur Generierung von Ableitungen der Lösung des DAE-IVPs verwendet DAESOL-II die in Kapitel 4 beschriebene Technik der Internen Numerischen Differentiation. Dabei sind in der ersten Version die folgenden drei Möglichkeiten zur Sensitivitätsberechnung implementiert, zwischen denen der Benutzer zur Laufzeit wählen kann:

a) Berechnung mittels variierter Trajektorien,

b) Berechnung durch Lösen der Variations-DAE mittels Newton-Verfahren,

c) Berechnung durch direktes Lösen der Variations-DAE.

Variierte Trajektorien

Hierbei erfolgt die Berechnung ganz analog dazu, wie es in Abschnitt 4.2.2 beschrieben wurde. Beim Aufruf von DAESOL-II gibt der Benutzer eine Matrix $\mathcal{W}_{ddir}$ mit den Richtungen der zu berechnenden Ableitungen vor. Dann werden entsprechend dieser Matrix die Anfangswerte der gestörten variierten Trajektorien gebildet. Schließlich werden parallel die Nominaltrajektorie und die variierten Trajektorien mit demselben Diskretisierungsschema integriert und an jedem Ausgabepunkt die finiten Differenzen zur Approximation der Wronski-Matrizen gebildet.

Lösung der Variations-DAE mit Newton-Verfahren

Wir beschreiben hier aus Gründen der Übersichtlichkeit nur den Fall einer Ableitungsrichtung ζ, der allgemeine Fall wird völlig analog behandelt. Zur Berechnung der Sensitivitäten durch Lösen der VDAE nach dem Prinzip der IND diskretisieren wir wie in Abschnitt 4.2.2 skizziert die VDAE (4.7) und erhalten dann folgendes Gleichungssystem (ohne Argumente) für die Wronski-Matrix $\mathcal{W}_\zeta$ nach einer beliebigen Richtung $\zeta \in \mathbb{R}^{n_x+n_z+n_p+n_q}$ im Schritt $(n+1)$:

$$\begin{aligned} &\begin{pmatrix} \alpha_k A + h\, f_x - hA_x\dot{x} & h\, f_z - hA_z\dot{x} \\ g_x & g_z \end{pmatrix} \cdot \mathcal{W}_{\zeta,n+1} \\ &= \begin{pmatrix} -h((f_p - A_p\dot{x})(\zeta)_p + (f_q - A_q\dot{x})(\zeta)_q - A\; cc(\mathcal{W}_\zeta)_{n+1}) \\ -g_\zeta \end{pmatrix}, \end{aligned} \tag{5.44}$$

wobei $cc(W_\zeta)_{n+1}$ hier analog zur Berechnung der Nominaltrajektorie den konstanten Anteil des Korrektorpolynoms der Wronski-Matrix beschreibt.

Bemerkung 5.8.1

Die Matrix auf der linken Seite von (5.44) ist identisch zur Jakobi-Matrix (5.29) bei der Lösung des nichtlinearen Gleichungssystems für die Nominal-

trajektorie im aktuellen Punkt y_{n+1}.

Folgen wir strikt dem Prinzip der IND, so leiten wir zur Berechnung der Wronski-Matrix $\mathcal{W}_{\zeta,n+1}$ an der Stelle t_{n+1} auch die Berechnungsschritte des Iterationsverfahrens zur Gewinnung von y_{n+1} bei der Berechnung der Nominaltrajektorie ab.
Leiten wir also das Newton-ähnliche Verfahren

$$\begin{aligned} y_{n+1}^{(i+1)} &= y_{n+1}^{(i)} + \Delta y_{n+1}^{(i)}, \\ \Delta y_{n+1}^{(i)} &= -MF(y_{n+1}^{(i)}, p, q), \end{aligned}$$

nach ζ ab, wobei F hierbei wieder wie in (5.27) das Diskretisierungsschema des BDF-Verfahrens zusammenfaßt, M die Iterationsmatrix des Newton-ähnlichen Verfahrens beschreibt und sich i hier aufgrund der Monitor-Strategie zwischen 0 und maximal 2 bewegt.
Unter Anwendung der Kettenregel ergibt sich dann als Iterationsvorschrift für die Wronski-Matrizen

$$\begin{aligned} \frac{\partial y_{n+1}^{(i+1)}}{\partial \zeta} &= \frac{\partial y_{n+1}^{(i)}}{\partial \zeta} - M\left(\frac{\partial F(y_{n+1}^{(i)}, p, q)}{\partial y_{n+1}^{(i)}} \frac{\partial y_{n+1}^{(i)}}{\partial \zeta} + \frac{\partial F(y_{n+1}^{(i)}, p, q)}{\partial \zeta}\right) \\ \Leftrightarrow \mathcal{W}_{\zeta,n+1}^{(i+1)} &= \mathcal{W}_{\zeta,n+1}^{(i)} - M\left(J(x_{n+1}^{(i)}, p, q)\mathcal{W}_{\zeta,n+1}^{(i)} + F_\zeta(x_{n+1}^{(i)}, p, q)\right) \end{aligned} \quad (5.45)$$

wobei hier

$$J(x_{n+1}^{(i)}, p, q) = \frac{\partial F(y_{n+1}^{(i)}, p, q)}{\partial y_{n+1}^{(i)}} = \begin{pmatrix} \alpha_k A + h\, f_x - hA_x\dot{x} & h\, f_z - hA_z\dot{x} \\ g_x & g_z \end{pmatrix}$$

gerade die Matrix auf der linken Seite von (5.44) ist und

$$F_\zeta(x_{n+1}^{(i)}, p, q) = \begin{pmatrix} -h((f_p - A_p\dot{x})(\zeta)_p + (f_q - A_q\dot{x})(\zeta)_q - A\; cc(\mathcal{W}_\zeta)_{n+1}) \\ -g_\zeta \end{pmatrix} \quad (5.46)$$

die rechte Seite.
Dies bedeutet also, dass wir auf das Gleichungssystem (5.44) für die Wronski-Matrizen ein Newton-ähnliches Verfahren mit derselben Iterationsmatrix wie für die Nominaltrajektorien und derselben Iterationszahl anwenden, so dass wir die Iterationsmatrizen hier nicht neu berechnen müssen. Wir gewinnen dabei den Startwert analog zur Nominaltrajektorie aus dem entsprechenden Prädiktorpolynom. Während der Iterationen wird die linke Seite von (5.44) mittels der Richtungsableitungen der Modellfunktionen ausgewertet.

Direktes Lösen der Variations-DAE

Die dritte Möglichkeit zur Berechnung der Sensitivitäten, die in DAESOL-II implementiert ist, erhalten wir durch eine leichte Abweichung vom Prinzip der IND. Weil das Gleichungssystem (5.44) linear in der Wronski-Matrix $\mathcal{W}_{\zeta,n+1}$ ist, können wir es direkt lösen.
Dies erspart uns die Auswertungen der Ableitungen der Modellfunktionen während der Newton-ähnlichen Iterationen, dafür müssen wir die Matrix $J(x_{n+1}, p, q)$ aber in jedem Schritt zerlegen.

5.9 Ableitungen der Modellfunktionen

Während des Integrationsvorgangs werden Ableitungen der Modellfunktionen benötigt. Bei der Lösung der Nominaltrajektorie sind dies die Ableitungen nach den Zustandsvariablen x und z, in Fall der Sensitivitätsberechnung, außer bei der Benutzung der variierten Trajektorien, auch die nach Parametern und Kontrollen. In DAESOL-II können die benötigten Ableitungen entweder durch das Programm selbst mittels einer Approximation durch finite Differenzen oder durch den Benutzer in Form vom Funktionen übergeben werden. Dabei werden in DAESOL-II die Ableitungen soweit möglich gleich als Richtungsableitungen ausgewertet, was sowohl Speicherplatz wie auch Rechenzeit spart. Diese Möglichkeit wird sowohl bei der Berechnung durch DAESOL-II mittels finiter Differenzen, als auch bei der Übergabe der Ableitungsroutinen durch den Benutzer, sofern diese Richtungsableitungen unterstützen, ausgenutzt.
Im Folgenden wollen wir einige Möglichkeiten zur Ableitungsgenerierung kurz diskutieren.

Finite Differenzen

Hierbei werden, wie in Abschnitt 4.2.1 für die Approximation der Ableitungen der Nominaltrajektorie beschrieben, die Richtungsableitungen beispielsweise von f im Punkte ξ nach einer Richtung ζ approximiert durch den Differenzenquotienten

$$\frac{\partial f(t;\xi)}{\partial \zeta} \doteq \frac{f(t;\xi+\epsilon_\zeta \zeta) - f(t;\xi)}{\epsilon_\zeta}. \tag{5.47}$$

Wie bereits geschildert bereitet hier die Wahl von ϵ_ζ Probleme, insbesondere für unterschiedliche Größenordnungen der einzelnen Komponenten. Wählen wir es zu groß, so erhalten wir größere Fehler durch die Terme höherer Ordnung, die bei der Approximation vernachlässigt werden. Wählen wir es zu klein, so wächst durch Auslöschung der Fehleranteil, der auf den Rundungsfehlern beruht. Ein weiterer Nachteil ist, dass wir selbst im günstigsten Fall nur eine Genauigkeit von der halben Maschinengenauigkeit erwarten können. Dies kann bei hoher geforderter Genauigkeit an die berechneten Sensitivitäten zu einem Problem führen. Daher ist in diesem Fall oft eine Auswertung der Ableitungen auf eine der folgenden Weisen mehr zu empfehlen. Bei der Lösung der Nominaltrajektorie ist dies nicht so kritisch, da dort die Ableitungen der Modellfunktionen lediglich für die Iterationsmatrix des Newton-ähnlichen Verfahrens benötigt werden. Hierbei reicht meist, wie oben gezeigt, eine mäßig gute Approximation der Jakobi-Matrix des nichtlinearen Gleichungssystems aus.

Manuelle Berechnung

Hierbei werden die Ableitungen vom Benutzer analytisch von Hand berechnet und dann in einer Funktion programmiert und diese an DAESOL-II übergeben. Dies bietet den Vorteil, dass man die Berechnung der Ableitungen sehr effizient gestalten kann und eine hohe Genauigkeit erzielt wird. Allerdings ist das Ausrechnen der Ableitungen von Hand für große Modelle ersteinmal sehr aufwendig und auch fehleranfällig.

Symbolische Berechnung

Bei diesem Ansatz werden die Ableitungen aus einer Zeichenkette, welche die Funktionsauswertung beschreibt, auf symbolischen Wege mit Hilfe eines Computeralgebra-Systems wie beispielsweise Mathematica oder Maple wieder als Zeichenkette berechnet, und dann in eine Routine transformiert, die DAESOL-II übergeben wird. Der Vorteil ist die relativ komfortable Generierung von analytischen Ableitungen der Modellfunktionen. Allerdings sind die entstehenden Berechnungsvorschriften mitunter sehr komplex und aufwendig auszuwerten.

Automatische Differentiation

Eine effiziente Möglichkeit zur akkuraten numerischen Berechnung von (Richtungs)-Ableitungen der Modellfunktionen im Rahmen der Maschinengenauigkeit stellt die sogenannte Automatische Differentiation (AD) dar. Hierbei wird die Berechnungsvorschrift der Modellfunktion als Abfolge von sogenannten Elementaroperationen wie Addition, Multiplikation, Division, Exponentialfunktion, Wurzelfunktion usw. betrachtet, von denen die Ableitungen bekannt und leicht zu berechnen sind. Dies führt zu einer Graphen von vielen dieser Elementaroperationen zur Auswertung der Funktion, aus dem man dann eine lineare Sequenz von Auswertungen generieren kann. Die Ableitungen der Funktion erhält man dann durch Ableiten der einzelnen Elementaroperationen und durch sukzessive Anwendung der Kettenregel. Dies geschieht in engem Zusammenhang mit der Auswertung der Funktion selbst. Dabei treten lediglich Rundungsfehler auf, die auch bei der Funktionsauswertung unvermeidbar sind, aber keine Diskretisierungsfehler, so dass wir die Ableitungen faktisch mit Maschinengenauigkeit berechnen können. Die Automatische Differentiation erlaubt ebenfalls eine effiziente Ausnutzung der Dünnbesetztheit der Ableitungsmatrizen.
Es existieren zwei Möglichkeiten der Richtung, in der man sich beim Ableiten durch die Sequenz der Elementaroperationen bewegen kann, beide im Detail beschrieben in [Gri00].

Vorwärtsmodus Dieser wird zur Generierung von Richtungsableitungen nach einer Richtung ζ der Form

$$\dot{\xi} = \frac{df}{dx}\zeta,$$

eingesetzt, wie wir sie auch in der ersten Version von DAESOL-II benötigen. Dabei werden bei der Auswertung der Funktion parallel zur Berechnung der Zwischenwerte der Funktion die Ableitungen der Zwischenergebnisse nach der unabhängigen Variable x berechnet. Nach Abschluß der Auswertung der Funktion haben wir somit auch die Ableitung der Funktion vorliegen. Man kann zeigen, dass der Aufwand zur Berechnung einer Richtungsableitung im Vorwärtsmodus maximal $\frac{5}{2}$-mal so hoch ist wie zur Auswertung der Funktion selbst und praktisch kein zusätzlicher Speicherplatz benötigt wird. Bei der

Berechnung von n_{ddir} Richtungsableitungen beträgt der Aufwand maximal $(1 + 1.5\, n_{ddir})$-mal den der Funktionsauswertung [Gri00].

Rückwärtsmodus Dieser wird dazu eingesetzt, Linearkombinationen $\overline{\zeta}$ von Ableitungen der Form

$$\overline{\xi} = \overline{\zeta}\frac{df}{dx}$$

zu berechnen. Dabei werden schrittweise die Ableitungen des Endergebnisses der Funktionsauswertung nach den Zwischenergebnissen erzeugt. Da man sich dazu in umgekehrter Reihenfolge durch die Auswertungssequenz der Funktion bewegt, muß man zunächst eine komplette Auswertung der Funktion machen und die Zwischenergebnisse dabei abspeichern. Danach werden in einem sogenannten Rückwärtslauf die Ableitungen berechnet. Hierbei wird für die Abspeicherung der Zwischenergebnisse zusätzlicher Speicherplatz benötigt. Der Aufwand für die Berechnung einer Linearkombination ist hierbei maximal 4-mal so hoch wie der der Funktionsauswertung, bei $n_{linkomb}$ Linearkombinationen dann $(1.5 + 2.5\, n_{linkomb})$-mal so hoch [Gri00].

Bemerkung 5.9.1
Neben den in der ersten Version von DAESOL-II erforderlichen Richtungsableitungen, die bei Anwendung der Automatischen Differentiation am besten mit dem Vorwärtsmodus berechnet werden, werden ferner auch komplette Jakobi-Matrizen der Modellfunktionen benötigt. Nehmen wir an, dass $f : \mathbb{R}^n \to \mathbb{R}^m$, so ist die Jakobi-Matrix $\frac{df}{dx} \in \mathbb{R}^{m \times n}$. Dann kann die Jakobi-Matrix im Zuge der Automatischen Differentiation durch n Richtungsableitungen im Vorwärtsmodus oder m Linearkombinationen von Ableitungen im Rückwärtsmodus mit zusätzlichem Speicherbedarf gewonnen werden, so dass hier die beste Wahl von den konkreten Problemdimensionen abhängt.

Bemerkung 5.9.2
Der Rückwärtsmodus der Automatischen Differentiation ist außerdem im Hinblick auf die zukünftige Erweiterung von DAESOL-II mit einem Rückwärtsmodus der Sensitivitätsberechnung mittels der Lösung der adjungierten Variations-DAE von Interesse. Hierbei werden Linearkombinationen der Ableitungen der Modellfunktionen benötigt. Auch für die Generierung von Sensitivitäten höherer Ordnung ist der Rückwärtsmodus von Bedeutung. Wie aus (4.8) ersichtlich werden hierbei höhere Ableitungen der Modellfunktionen

benötigt, und durch eine Kombination von Vorwärts- und Rückwärtsmodus der AD können beispielsweise sehr effizient zweite Ableitungen erzeugt werden.

Bemerkung 5.9.3 (Implementierungen der AD)
Bei der Implementierung der AD finden sich vor allem zwei Ansätze.

a) Die Sourcecode-Transformation, bei der aus dem Quellcode der Funktion, nach dem Prinzip der AD, Quellcode für die Ableitungen generiert wird.
   Diese Möglichkeit wird beispielsweise in den Tools ADIFOR [BCC+92] für Fortran-Routinen bzw. ADIC für C-Routinen verwendet.

b) Das Operator-Overloading in objektorientierten Programmiersprachen. Hierbei werden die Elementaroperationen überladen mit der zusätzlichen Berechnung der Ableitungen. Dies ist beispielweise in dem Tool ADOL-C [GJM+99] in C++ realisiert.

5.10 Relaxierung der algebraischen Gleichungen

Bei der numerischen Lösung von Optimierungsproblemen, bei denen ein DAE-Modell auftritt, kann es mitunter wünschenwert oder sogar notwendig sein, DAE-IVPs mit inkonsistenten Anfangwerten für die algebraischen Variablen starten zu können. Dies ist beispielsweise der Fall, wenn das Optimierungsproblem mit Newton-Typ-Verfahren gelöst wird und wenn die algebraischen Variablen von den zu optimierenden Variablen abhängen, da dann meist nach jeder Iteration die Konsistenzbedingungen verletzt sind.
In diese Fällen ist eine Relaxierung der algebraischen Gleichungen empfehlenswert (Bock et. al in [BES88]). Der Optimierungsalgorithmus ist in diesem Fall dafür verantwortlich, dass im Lösungspunkt des Optimierungsproblems die Konsistenz gewährleistet ist, beispielsweise durch Hinzunahme der algebraischen Gleichungen im Startpunkt als Nebenbedingungen, wie dies im Code MUSCOD-II [Lei99] realisiert ist.

Die Relaxierung führt auf ein DAE-IVP der Form

$$\begin{aligned} A(t,x,z)\dot{x} &= f(t,x,z) \\ 0 &= g(t,x,z) - \vartheta(t)g(t_0,x_0,z_0), \\ x(t_0) &= x_0, \\ z(t_0) &= z_0. \end{aligned} \tag{5.48}$$

Dabei wird die Relaxierungsfunktion $\vartheta : \mathbb{R} \to \mathbb{R}$ als genügend glatt angenommen und es gilt $\vartheta(t_0) = 1$, $\vartheta(t) \geq 0$ und i.A. ist $\vartheta(t)$ monoton fallend. Falls die Relaxierungsfunktion stark genug fällt und man über einen hinreichend langen Zeitraum integriert, erhält man schließlich konsistente Werte.
Diese relaxierte Formulierung bietet im Optimierungskontext den Vorteil, dass nun alle Anfangswerte frei wählbar sind.
Der hinzugefügte Relaxierungsterm erscheint bei dieser Formulierung identisch im Diskretisierungsschema für die Nominaltrajektorie.

In DAESOL-II ist die Möglichkeit der Relaxierung mittels der Standard-Relaxierungsfunktion

$$\vartheta(t) = e^{-p_{relax}\left(\frac{t-t_0}{t_{end}-t_0}\right)}$$

implementiert, mit einem durch den Benutzer frei wählbaren Relaxierungsparameter p_{relax}. Außerdem ist eine Möglichkeit der Implementierung weiterer Relaxierungsfunktionen vorgesehen.

Bemerkung 5.10.1 (Relaxierung bei der Sensitivitätsberechnung)
Wenn die relaxierte Formulierung bei der Lösung der Nominaltrajektorie angewendet wird, so verändern sich entsprechend auch die zu lösenden VDAEs und die entsprecheden Diskretisierungsschemata. Dort erscheint ein zusätzlicher Term, der sich aus der Relaxierungsfunktion und den entsprechenden Ableitungen von g nach Anfangswerten, Parametern bzw. Kontrollen im Startpunkt (t_0, x_0, z_0) zusammensetzt. So muß beispielsweise der algebraische Teil der VDAE (4.7) modifiziert werden zu

$$\begin{aligned} 0 = &\begin{pmatrix} g_x & g_z & g_p & g_q \end{pmatrix} \begin{pmatrix} \overline{W}^x \\ \overline{W}^z \\ (\mathcal{W}_{ddir})_p \\ (\mathcal{W}_{ddir})_q \end{pmatrix} \\ &-\vartheta(t) \begin{pmatrix} g_{x,0} & g_{z,0} & g_{p,0} & g_{q,0} \end{pmatrix} \mathcal{W}_{ddir}, \end{aligned}$$

wobei die Indizierung der Ableitungen von g mit 0 die Auswertung an der Stelle (t_0, x_0, z_0, p, q) kennzeichnet. Der Ableitungsterm der Relaxierung muß nur einmal zu Beginn der Integration ausgewertet und abgespeichert werden.

5.11 Kontinuierliche Darstellung der Lösungen

Durch die im BDF-Verfahren stattfindende Polynominterpolation der letzten Trajektorienwerte bietet sich eine einfache und effiziente Möglichkeit zur Generierung einer kontinuierlichen Darstellung der Lösungen. Diese ermöglicht es, die Ausgabe der Lösungen vom Integrationsgitter zu entkoppeln und somit die Werte der Lösung an beliebigen Stellen, die nicht auf dem Integrationsgitter liegen, fehlerkontrolliert zu berechnen. Diese Darstellung kann bei späteren Erweiterungen wie beispielsweise der Behandlung von implizit definierten Modelländerungen auch dazu genutzt werden, Unstetigkeitsstellen bzw. Schaltpunkte zu suchen (siehe auch [BP04], [Eic91]).
Zur kontinuierlichen Darstellung der Lösung im Intervall $[t_n, t_{n+1}]$ verwenden wir das Interpolationspolynom $\mathcal{P}^I_{n+1}(t)$, welches die Werte $y_{n+1}, \ldots, y_{n-k}$ interpoliert, d.h. mit dem Korrektorpolynom des gerade durchgeführten Schrittes identisch ist. Falls wir die Ordnung im nächsten Schritt beibehalten, so ist es auch identisch mit dem Prädiktorpolynom in nächsten Schritt. Wir erhalten also

$$\begin{aligned} \mathcal{P}^I_{n+1}(t) &= \sum_{j=0}^{k} \prod_{i=0}^{j-1} (t - t_{n+1-i})\; y[t_{n+1}, \ldots, t_{n+1-j}] \qquad (5.49) \\ &= \mathcal{P}^C_{n+1}(t) \\ &= \mathcal{P}^P_{n+2}(t) \text{ (falls die Ordnung konstant bleibt).} \end{aligned}$$

Man kann zeigen, dass diese kontinuierliche Darstellung der Lösungen zumindest bei äquidistantem Integrationsgitter die Bedingungen der *natürlichen Interpolation* [BS81] erfüllt. Das bedeutet, dass der Interpolationsfehler bei der Auswertung von $\mathcal{P}^I$ asymptotisch kleiner ist als der Diskretisierungsfehler des BDF-Verfahrens an den Gitterpunkten.

Die Ausgabe von Lösungen in DAESOL-II beruht auf der soeben beschriebenen kontinuierlichen Lösungsdarstellung. Der Benutzer kann dabei sowohl einen Vektor mit beliebigen Ausgabestellen, als auch ein äquidistan-

tes Gitter definieren, auf dem Lösungen ausgegeben werden sollen. Ferner kann DAESOL-II an das Problem angepaßte, vom Nutzer konfigurierbare Gnuplot-Dateien erstellen. Schließlich kann der Benutzer noch eine eigene Ausgaberoutine übergeben, die dann an den Ausgabestellen aufgerufen wird.

5.12 Softwarebibliotheken

DAESOL-II bedient sich zur Erzielung eines hohen Maßes an Effizienz bei der Ausführung von Standard Matrix-Vektor- und Matrix-Matrix-Operationen, sowie zur Lösung linearer Gleichungssysteme der Standard-Schnittstellen von BLAS (Basic Linear Algebra Subprograms) und LAPACK (Linear Algebra PACKage) im Falle der Behandlung von dichtbesetzten Matrizen. Ferner besitzt es eine variable Darstellung von dünnbesetzten Matrizen zur Möglichkeit der Verwendung verschiedener Software-Bibliotheken, da es für den dünnbesetzten (engl.: sparse) Fall keine einheitliche verbreitete Schnittstelle gibt. Dies ermöglicht die Verwendung von sowohl für die einzelnen Probleme als auch für die verwendete Hardware-Plattform optimierten Bibliotheken. In der ersten Version von DAESOL-II werden standardmäßig die folgenden beiden frei verfügbaren Bibliotheken verwendet.

ATLAS

Die Automatically Tuned Linear Algebra Sofware (ATLAS) [ATL] ist eine frei verfügbare Bibliothek, die die komplette Funktionalität des BLAS-Standards [BLA02, BDD+02], sowie eine Teilmenge der Routinen von LAPACK [ABB+99] zur Verfügung stellt. Sie wurde von Whaley und Petitet [WPD01] entwickelt. Das besondere an ihr ist, dass die bereitgestellten Routinen automatisch ohne Zutun des Benutzers bei der Kompilierung für den Aufbau der vorhandene Hardware wie z.B. die Cache-Hierarchie und den Befehlssatz des verwendeten Prozessors optimiert werden. Dies ermöglicht eine sehr hohe Effizienz der Linearen-Algebra-Operationen auch auf Systemen, für die keine von Hersteller optimierten Bibliotheken verfügbar sind. Selbst wenn diese vorhanden sind, so ist ihr Performancegewinn im Vergleich zu ATLAS meist nur marginal.
Die Geschwindigkeitsgewinne insbesondere bei Matrix-Operationen mit dichtbesetzten Matrizen auf den Systemen der DAESOL-II-Entwickler lagen beim

Faktor 10-15 gegenüber der Standard-BLAS bzw. LAPACK.

UMFPACK

UMFPACK ist eine frei verfügbare Bibliothek zur Lösung von dünnbesetzten Gleichungssystemen mit unsymmetrischen Matrizen, die ihrerseits zur Erlangung ihrer Effizienz auch vollen Gebrauch der ATLAS-BLAS macht. UMFPACK steht für Unsymmetric-pattern MultiFrontal PACKage und wurde von Davis et al. entwickelt [Dav04, Dav]. Neben der Lösung von Gleichungssystemen und der Möglichkeit, mit variablen Sparsity-Strukturen zu arbeiten, bietet sie Hilfsroutinen zur effizienten Umwandlung zwischen verschiedenen Darstellungen von Sparse-Matrizen, so dass sie sich auch im Hinblick auf die spätere Verwendung anderer Sparse-Bibliotheken gut als Grundlage eignet.

5.13 Das Programmpaket DAESOL-II

In diesem Abschnitt beschreiben wir kurz die Schnittstelle von DAESOL-II, geben einen kurzen Überblick über den modularen Aufbau des Programms in Form eines Ablaufdiagramms und beschreiben einige weitere Routinen, die im Zuge der Entwicklung von DAESOL-II entstanden.

Die Schnittstelle

Die Schnittstelle von DAESOL-II spiegelt die Ziele der Erweiterbarkeit und Modularität wieder. Der Aufruf von DAESOL-II erfolgt mittels des Programmcodes

```
int DAESOL(
    double* xz,
    double* p,
    double* q,
    double* time,
    int (*EVAL_F)(double, double *, ... , double *),
    int (*EVAL_G)(double, double *, ... , double *),
    int (*EVAL_A)(double, double *, ... , struct TDAESOL_DMat ),
    int (*EVAL_S)(double, double *, ... , double * ),
    struct TIntegratorControls *IntegratorControls,
    struct TIntegratorConstants *IntegratorConstants,
```

```
    struct TSensitivityControls *SensitivityControls,
    struct TIntegratorStatus *IntegratorStatus
).
```

Dabei sind beim Aufruf `xz`, `p` und `q` die Zeiger auf die (Start-)Werte der entsprechenden Variablen, `time` auf die Anfangszeit, bei der Rückkehr enthalten sie die entsprechenden Werte am Ende der Integration.
Die mit EVAL_ beginnenden Zeiger sind Zeiger auf die vom Benutzer bereitzustellenden Auswertungsroutinen für die Modellfunktionen f, g und A, sowie für eventuelle Schaltfunktionen S. Diese letzte Möglichkeit wird von der ersten Version nicht benutzt und ist für spätere Erweiterungen gedacht. Falls eine der Funktionen nicht benötigt wird, kann sie einfach durch einen NULL-Zeiger ersetzt werden.
Die vier Strukturen am Ende enthalten alle anderen von DAESOL-II verwendeten Informationen und Konfigurationsmöglichkeiten, thematisch in einzelne Strukturen aufgeteilt. Die Subsummierung der Parameter und Konfigurationen auf diese Strukturen hat den Vorteil, dass bei späteren Erweiterungen die Schnittstellen nicht geändert werden müssen.
In der Struktur `IntegratorControls` sind die vom Benutzer normalerweise zur Lösung eines Anfangswertproblems vorzugebenden Informationen zusammengefaßt. Dazu gehören die Problemdimensionen, die Endzeit, die Skalierungen und Toleranzen und auch die Art der Systemmatrix A sowie die Methode der Ableitungsgenerierung der Modellfunktionen und ggf. die Ableitungsroutinen.
Die Struktur `IntegratorConstants` enthält Konfigurationen und Einstellungen, die vom Benutzer normalerweise nicht, oder nur selten geändert werden müssen. Diese beinhalten neben anderen die Maschinengenauigkeit, die maximale und die minimale Ordnung und Schrittweite des BDF-Verfahrens, Schranken an Ordnungs- und Schrittweitenänderungen und Einstellungen zum Start des Verfahrens und der relaxierten Formulierung.
Mit der Struktur `SensitivityControls` werden die Einstellungen zur Generierung der Sensitivitäten geregelt. Darunter fallen beispielsweise die Wahl des Modus der Sensitivitätsberechnung sowie die Ableitungsrichtungen für die Richtungsableitungen und ihre Anzahl.
Die Struktur `IntegratorStatus` hingegen muß nicht zwingend vor dem Aufruf von DAESOL-II vom Benutzer geändert werden. Sie enthält neben vielen

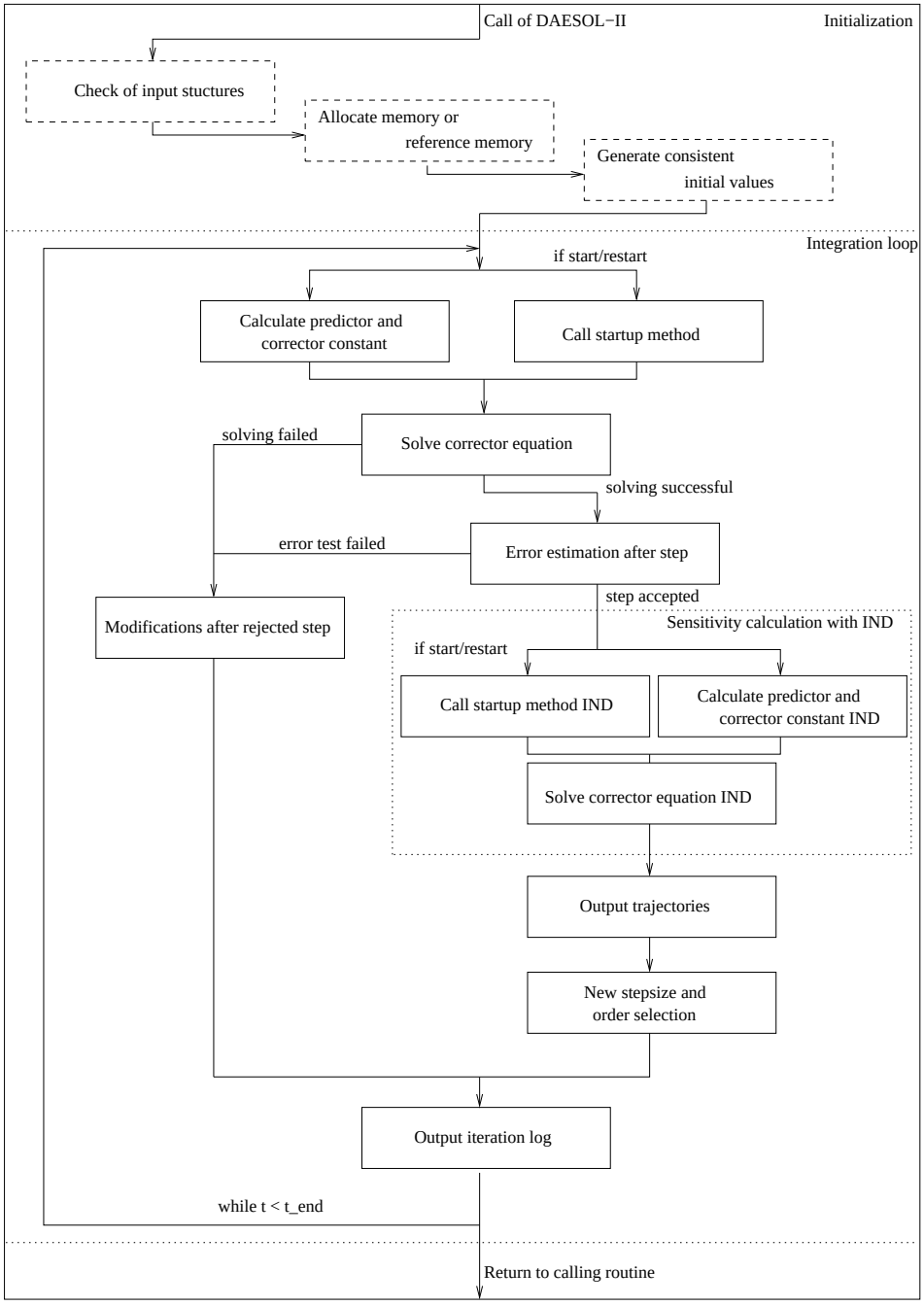
statistischen Informationen wie gesamter Schrittzahl, Anzahl der Funktions- und Ableitungsauswertungen sowie der Matrizenzerlegungen auch Zeiger auf Informationen, die den aktuellen Status des Integrators beschreiben, wie die aktuelle Zeit, die Werte der Trajektorien und dividierten Differenzen, den aktuellen Skalierungsvektor und die aktuelle Fehlerschätzung. Damit ist es möglich, den Integrator nach jedem Schritt anzuhalten, die Informationen zu verwerten, und die Integration durch Übergabe der unveränderten Struktur mittels eines sogenannten Warm-Starts nahtlos fortzusetzen.

Weitere Routinen

Neben der eigentlichen Integrationsroutine entstanden im Zuge der Entwicklung noch einige für den Benutzer nützliche Hilfsroutinen sowie ein Rahmenprogramm zur einfachen Implementierung von eigenen Problemen.
Die Hilfsroutinen, die für den Benutzer von Bedeutung sind, dienen der Initialisierung der oben genannten Kontrollstrukturen. Sie sind nach dem Schema `FORMAT_{name_der_struktur}` bezeichnet und erlauben eine einfache Initialisierung der jeweiligen Struktur mit definierten Werten, so dass DAESOL-II später bei Einstellungen, die der Benutzer nicht geändert hat, Standardeinstellungen verwenden kann.
Das Programm `daesol_test` bietet eine einfach zu bedienende Möglichkeit zur Einbindung von eigenen Anfangswertproblemen, die dann mittels des Rahmenprogramms aufgerufen können. Dabei sind die wichtigsten Einstellungsmöglichkeiten über Kommandozeilenparameter steuerbar.

Programmaufbau

Der Aufbau der Routine `daesol` gestaltet sich folgendermaßen:



5.14 Dokumentation

Eines der Ziele der Entwicklung von DAESOL-II ist die gute Erweiterbarkeit und einfache Bedienbarkeit des Integrators. Damit sich auch andere Entwickler und ggf. auch Anwender in dem doch umfangreichen und komplexen Programmcode, wie er mit einem modernen Integrator einhergeht, zurechtfinden und eine gute Wartbarkeit des Programmcodes gewährleistet ist, ist eine gute Dokumentation des Quelltextes und der Programmstruktur notwendig. Zu diesem Zweck verwenden wir in DAESOL-II das Programm *doxygen*, welches von van Heesch entwickelt wird [vH] und von ihm unter der freien GPL-Lizenz [GNU] zur Verfügung gestellt wird. Es extrahiert aus den C-Quelltexten mit Hilfe von speziellen Kommentaren Informationen und Beschreibungen von Funktionen, Parametern, Variablen, Aufrufhierarchien u.ä. Diese werden, zum Teil graphisch, aufbereitet und wahlweise im HTML-Format oder auch als Latex- oder PDF-Datei ausgegeben. Im HTML-Format kann dann beispielsweise der Quelltext interaktiv mit einem Browser betrachtet werden, und von einem Funktionsparameter oder einer `struct`-Komponente führt dann ein Link zu dessen Definition. Dies bedeutet eine erhebliche Vereinfachung einerseits bei der Dokumentation und Wartung des DAESOL-II-Projekts, andererseits auch ein einfacheres Zurechtfinden für den Anwender und den Entwickler von Erweiterungen.

Kapitel 6

Anwendungen

In diesem Kapitel demonstrieren wir anhand von Anwendungen die Zuverlässigkeit und die Effizienz der von uns vorgestellten Methoden und des von uns implementierten Integrators DAESOL-II. Hierzu beginnen wir aus Referenzzwecken mit zwei einfachen Beispielen, für die die Lösungen einfach analytisch zugänglich sind. Im Anschluß daran behandeln wir einige Aufgaben aus Beispielsammlungen für Anfangswertproblemlöser für steife Probleme. Zum Abschluß beschäftigen wir uns mit dem Modell eines Destillationsprozesses aus der Anwendung und der interessanten Peroxidase-Oxidase-Reaktion aus der Biochemie.
Soweit nicht anders angegeben verwenden wir dabei stets eine dichte Darstellung bei der Lösung der Linearen-Algebra Subprobleme und für die Toleranzen wählen wir $rel\_tol = a\_tol = TOL$. Die benötigten Ableitungen der Modellfunktionen werden i.a. mittels finiter Differenzen berechnet.

6.1 Einfache Sensitivitätstestprobleme

In diesem Abschnitt behandeln wir zu Testzwecken zwei relativ einfache Beispiele, bei denen wir die Lösungen und auch die Sensitivitäten analytisch als Referenz berechnen können.

6.1.1 Die Dahlquistgleichung mit Parameter

Wir betrachten die Dahlquistgleichung

$$\dot{y} = -\lambda y, \qquad \text{mit } \lambda > 0, \quad y, \lambda \in \mathbb{R}.$$

Sei $t_0 = 0$ und $y(t_0) = y_0$ dann erhalten wir die Lösung

$$y(t) = y_0 \, e^{-\lambda t}$$

und als Lösungen für die Sensitivitäten nach y_0 bzw. λ

$$\begin{aligned} \frac{\partial y(t)}{\partial y_0} &= e^{-\lambda t} \quad \text{und} \\ \frac{\partial y(t)}{\partial \lambda} &= -t \, y_0 \, e^{-\lambda t}. \end{aligned}$$

Wir verwenden die Werte $y_0 = 1$, $\lambda = 1$ und lösen das Problem auf dem Intervall $I = [0, 20]$. Wir vergleichen im Folgenden die Lösungen sowie die Schrittweiten und Fehlerschätzungen von DAESOL-II und dem Vorgängercode DAESOL.

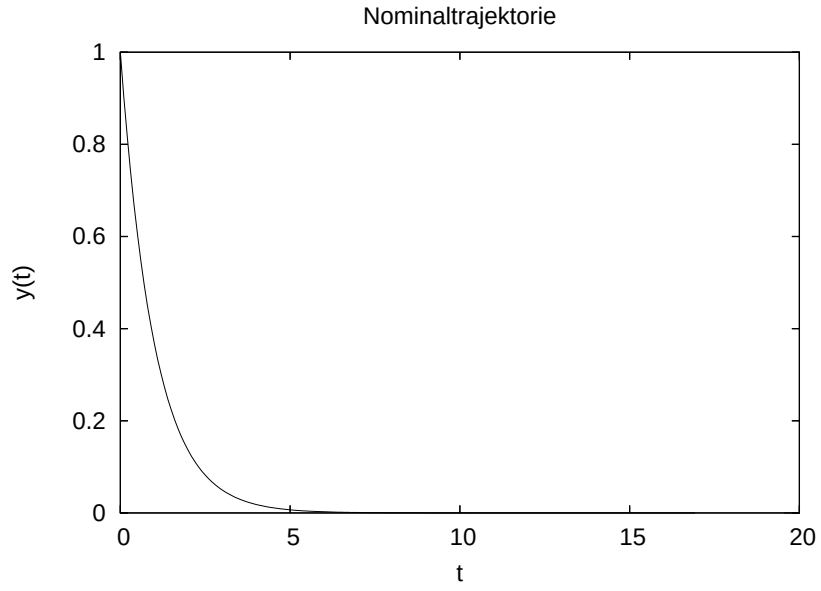


Abbildung 6.1: Die Nominaltrajektorie für die Dahlquistgleichung, berechnet mit den Werten $y_0 = 1, \lambda = 1$ und der Toleranz $TOL = 10^{-6}$.

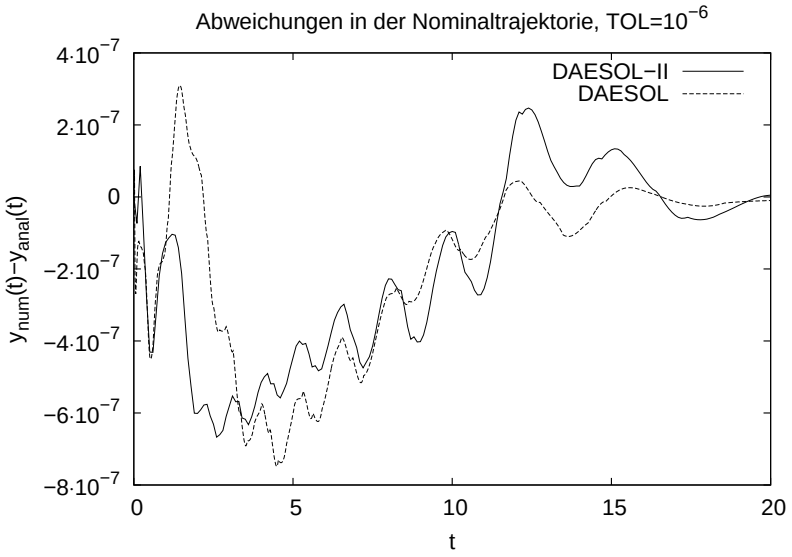


Abbildung 6.2: Abweichungen der mit DAESOL bzw. DAESOL-II berechneten Nominaltrajektorien für die Dahlquistgleichung mit den Werten $y_0 = 1$, $\lambda = 1$ und der Toleranz $TOL = 10^{-6}$ von der analytischen Lösung über dem Integrationsintervall.

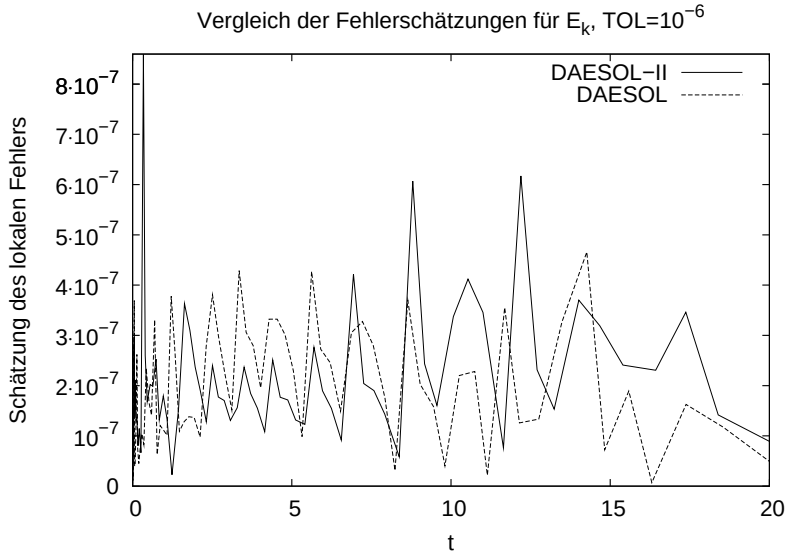


Abbildung 6.3: Schätzungen des lokalen Fehlers in den einzelnen Integrationsschritten von DAESOL bzw. DAESOL-II bei der Berechnung der Nominaltrajektorie der Dahlquistgleichung für $y_0 = 1$, $\lambda = 1$ und der Toleranz $TOL = 10^{-6}$.

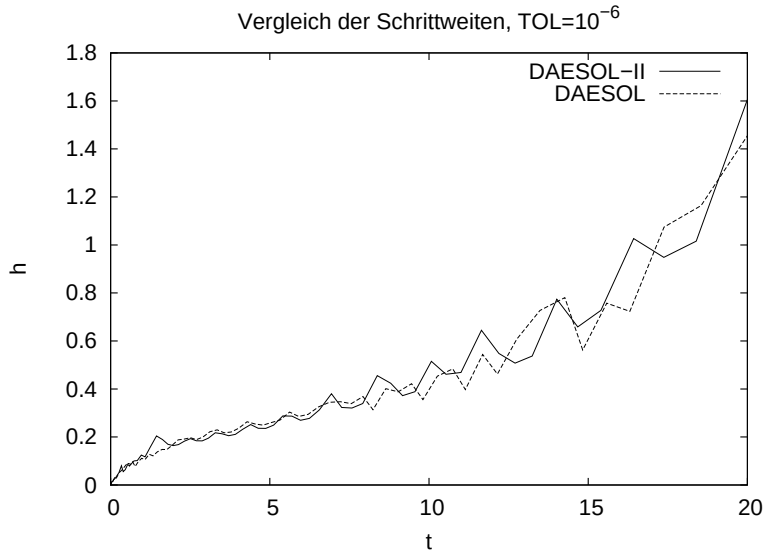


Abbildung 6.4: Verwendete Schrittweiten von DAESOL bzw. DAESOL-II bei der Berechnung der Nominaltrajektorie der Dahlquistgleichung für $y_0 = 1$, $\lambda = 1$ und der Toleranz $TOL = 10^{-6}$.

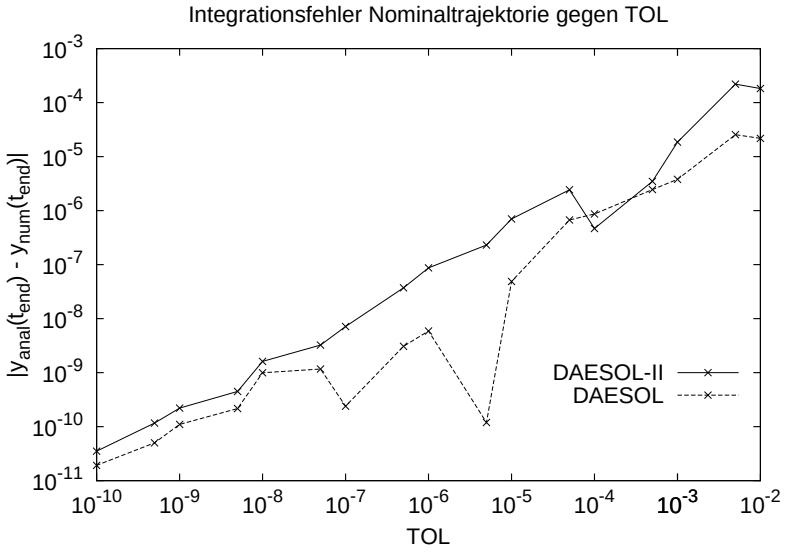


Abbildung 6.5: Die Relation zwischen dem Fehler am Ende des Integrationsintervalls bei der Berechnung der Nominaltrajektorie der Dahlquistgleichung mit DAESOL bzw. DAESOL-II für $y_0 = 1$, $\lambda = 1$ und der von Benutzer vorgegebenen Toleranz für Toleranzen von $TOL = 10^{-10}$ bis $TOL = 10^{-2}$.

	TOL	#Steps	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL	10^{-4}	53	104	8	1
DAESOL	10^{-6}	83	165	6	1
DAESOL	10^{-8}	126	292	7	1
DAESOL	10^{-10}	215	527	10	2
DAESOL-II	10^{-4}	51	105	9	1
DAESOL-II	10^{-6}	79	170	7	1
DAESOL-II	10^{-8}	124	308	8	2
DAESOL-II	10^{-10}	218	547	8	2

Tabelle 6.1: Der numerische Aufwand von DAESOL bzw. DAESOL-II bei der Berechung der Nominaltrajektorien der Dahlquistgleichung für $y_0 = 1$, $\lambda = 1$ und verschiedene Werte für TOL.

Wir beobachten also von einem Ausreißer abgesehen prinzipiell ein ähnliches Fehlerverhalten, und eine vergleichbaren Aufwand der beiden Verfahren.

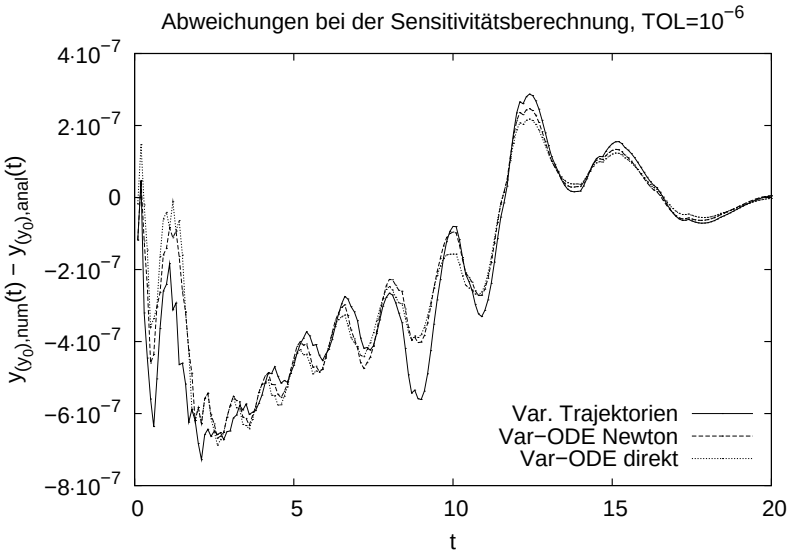


Abbildung 6.6: Abweichungen bei der Berechnung der Sensitivität $y_{(y_0)}(t) = \frac{\partial y(t)}{\partial y_0}$ mit DAESOL-II mittels der implementierten Varianten von der analytisch erhaltenen Sensitivität der Lösung der Dahlquistgleichung nach dem Anfangswert y_0 für $y_0 = 1$, $\lambda = 1$ und der Toleranz $TOL = 10^{-6}$.

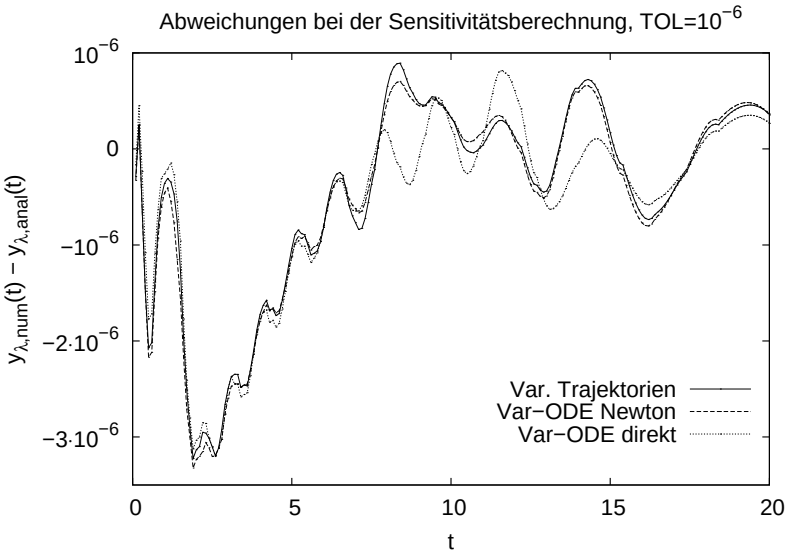


Abbildung 6.7: Abweichungen bei der Berechnung der Sensitivität $y_\lambda(t) = \frac{\partial y(t)}{\partial \lambda}$ mit DAESOL-II mittels der implementierten Varianten von der analytisch erhaltenen Sensitivität der Lösung der Dahlquistgleichung nach dem Parameter λ für $y_0 = 1$, $\lambda = 1$ und der Toleranz $TOL = 10^{-6}$.

	Methode	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL	V-ODE Direkt	467	83	80
DAESOL-II	Var. Traj.	488	7	1
DAESOL-II	V-ODE Newton	170	7	160
DAESOL-II	V-ODE Direkt	170	84	78

Tabelle 6.2: Numerischer Aufwand bei der Sensitivitätsberechnung nach y_0 und λ für die Dahlquistgleichung mittels DAESOL bzw. DAESOL-II für $y_0 = 1$, $\lambda = 1$ und TOL=10^{-6}.

Hierbei können wir gut die unterschiedlichen Auswirkungen der Art der Sensitivitätsberechnung im Hinblick auf die Anzahl der benötigten Funktions- und Ableitungsauswertungen begutachten, wie die Tabelle 6.2 demonstriert. Je nach Aufwand zur Berechnung von Funktionen bzw. Ableitungen ist daher in der Praxis zwischen der Anwendung der einzelnen Arten abzuwägen, um eine optimale Effizienz zu erlangen.

6.1.2 Freier gedämpfter harmonischer Oszillator

Der freie gedämpfte eindimensionale harmonische Oszillator wird beschrieben durch die Bewegungsgleichung

$$m\ddot{z} + c\dot{z} + Dz = 0.$$

Ein mechanisches Beispiel dafür ist ein Massepunkt an einer Feder, dessen Bewegung gedämpft wird. Wir definieren die Abkürzungen

$$\gamma := \frac{c}{2m} \qquad \text{und} \qquad \omega_0^2 := \frac{D}{m}$$

und erhalten die gewönliche Differentialgleichung zweiter Ordnung

$$\ddot{z} + 2\gamma\dot{z} + \omega_0^2 z = 0.$$

Seien die Anfangwerte als $z(t_0) = z_0 \in \mathbb{R}$ und $\dot{z}(t_0) = \dot{z}_0 \in \mathbb{R}$ gegeben. Abhängig von dem Verhältnis von γ und ω_0^2 können wir dann drei Fälle unterscheiden:

a) Für $\gamma^2 < \omega_0^2$ die gedämpfte harmonische Schwingung mit der Lösung

$$z(t) = e^{-\gamma t}\Big(z_0 \cos(\omega t) + \frac{\gamma z_0 + \dot{z_0}}{\omega} \sin(\omega t)\Big),$$

wobei $\omega := \sqrt{\omega_0^2 - \gamma^2}$.

b) Für $\gamma^2 > \omega_0^2$ den Kriechfall mit der Lösung

$$z(t) = e^{-\gamma t}\Big(z_0 \cosh(\sqrt{\gamma^2 - \omega_0^2}\, t) + \frac{\gamma z_0 + \dot{z_0}}{\sqrt{\gamma^2 - \omega_0^2}} \sinh(\sqrt{\gamma^2 - \omega_0^2}\, t)\Big),$$

c) Für $\gamma^2 = \omega^2$ den aperiodischen Grenzfall mit

$$z(t) = e^{-\gamma t}\Big(z_0 + (\dot{z_0} + \gamma z_0)\, t\Big).$$

Wir formulieren das Problem als System von zwei gewöhnlichen Differentialgleichungen folgendermaßen: Mit $y = (y_1, y_2) \in \mathbb{R}^2$ erhalten wir als rechte Seite

$$f(t, y) = \begin{pmatrix} y_2 \\ -2\gamma y_2 - \omega_0^2 y_1 \end{pmatrix}.$$

Für die folgenden Tests wählen wir die Anfangswerte $z_0 = (y_1)_0 = 2$ und $\dot{z}_0 = (y_2)_0 = 0$ und integrieren auf dem Intervall $I = [0, 100]$. Die drei Fälle werden getrennt betrachtet.

Gedämpfte harmonische Schwingung

Wir wählen $\gamma = 0.1$ und $\omega_0 = 1$ und erhalten folgende Resultate. Dabei vergleichen wir wieder mit dem Code DAESOL sowie bei der Berechnung der Nominaltrajektorie mit MATLAB mit dem Löser ode15s, der ebenfalls für steife Probleme konstruiert ist und teilweise auf BDF-Verfahren basiert. Aus Platzgründen beschränken wir uns bei der detaillierteren Fehleranalyse und der Sensitivitätsberechnung auf die Komponente $z(t) = y_1(t)$.

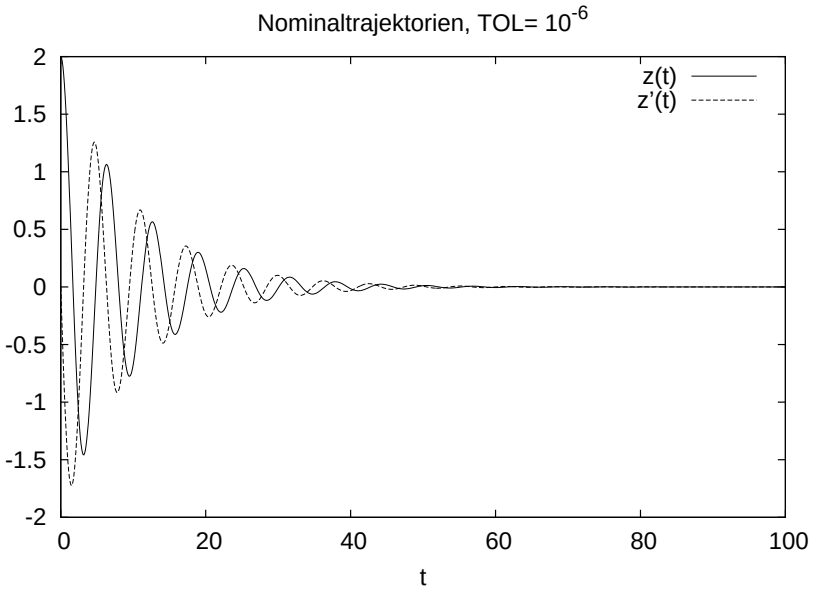


Abbildung 6.8: Numerisch mit DAESOL-II berechnete Nominaltrajektorien des harmonischen Oszillators für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 \equiv y_{1,0} = 2$, $z_0' \equiv \dot{z}_0 \equiv y_{2,0} = 0$ und die Toleranz $TOL = 10^{-6}$. Hierbei beobachten wir eine gedämpfte harmonische Schwingung, wobei in der Darstellung $z(t) \equiv y_1(t)$ den Ort des Oszillators beschreibt und $z'(t) \equiv \dot{z}(t) \equiv y_2(t)$ seine Geschwindigkeit.

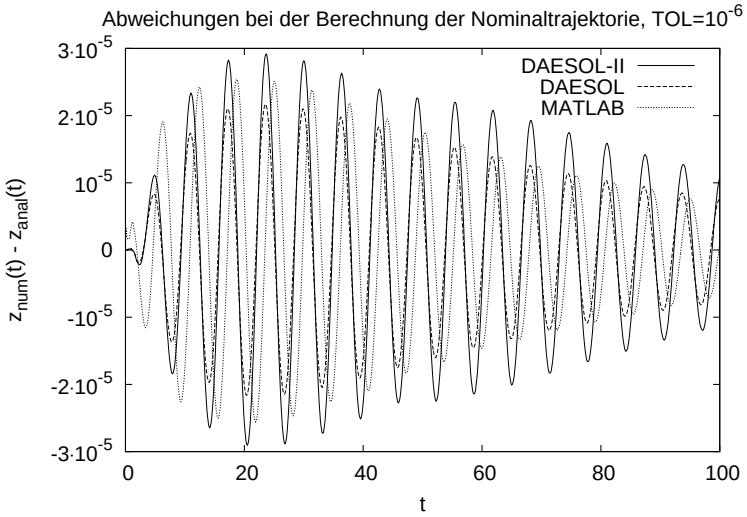


Abbildung 6.9: Abweichungen der mit DAESOL, DAESOL-II bzw. MATLAB berechneten Nominaltrajektorie für $z(t) \equiv y_1(t)$ für den harmonischen Oszillator mit den Werten $\gamma = 0.1$, $\omega_0 = 1$, $z_0 \equiv y_{1,0} = 2$, $\dot{z}_0 \equiv y_{2,0} = 0$ und die Toleranz $TOL = 10^{-6}$ von der analytischen Lösung über dem Integrationsintervall.

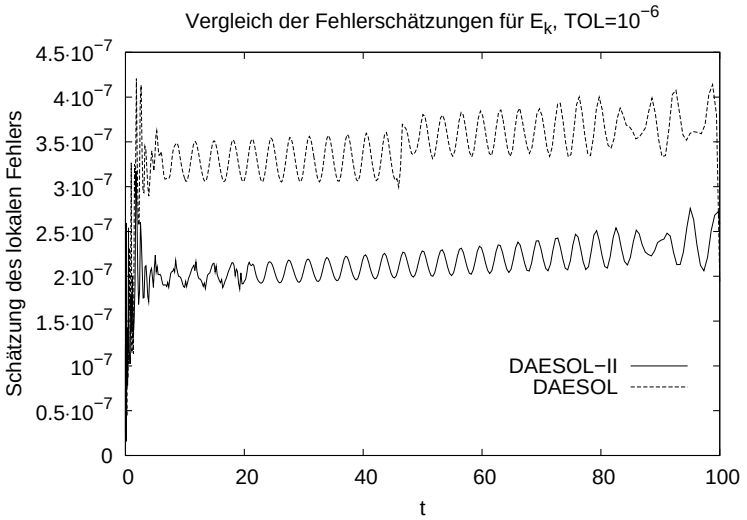


Abbildung 6.10: Schätzungen des lokalen Fehlers in den einzelnen Integrationsschritten von DAESOL bzw. DAESOL-II bei der Berechnung der Nominaltrajektorien des harmonischen Oszillators für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$. Das scheinbar unruhigere Verhalten gegen Ende des Integrationsintervalls ist rein darstellungsbedingt aufgrund der gestiegenden Schrittweiten.

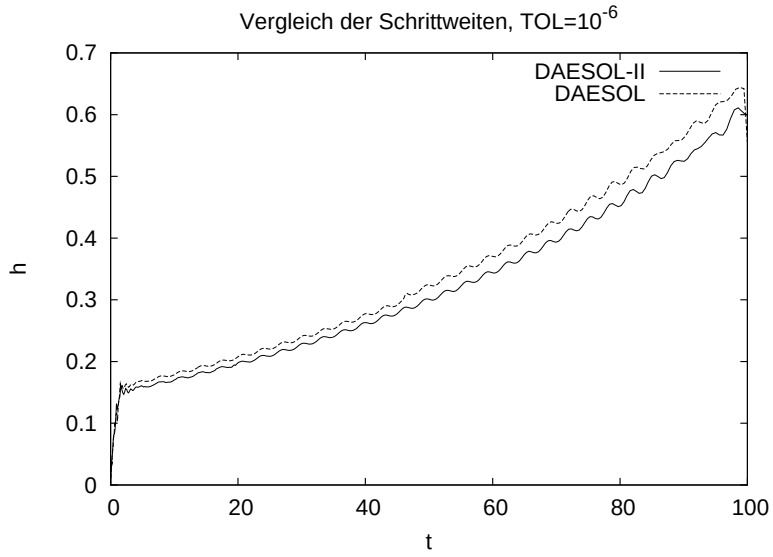


Abbildung 6.11: Verwendete Schrittweiten von DAESOL bzw. DAESOL-II bei der Berechnung der Nominaltrajektorien des harmonischen Oszillators für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$.

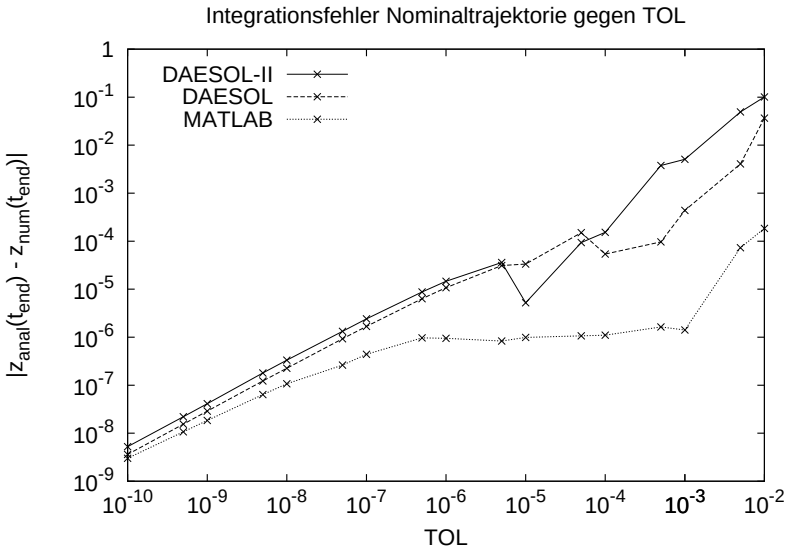


Abbildung 6.12: Die Relation zwischen dem Fehler am Ende des Integrationsintervalls bei der Berechnung der Nominaltrajektorie $z(t) \equiv y_1(t)$ des harmonischen Oszillators mit DAESOL, DAESOL-II bzw. MATLAB für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und vorgegebenen Toleranzen von $TOL = 10^{-10}$ bis $TOL = 10^{-2}$.

	TOL	#Steps	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL	10^{-3}	187	429	10	1
DAESOL	10^{-7}	513	1463	6	1
DAESOL-II	10^{-3}	54	96	7	1
DAESOL-II	10^{-7}	111	248	8	2
MATLAB	10^{-3}	290	319	22	1
MATLAB	10^{-7}	727	823	89	1

Tabelle 6.3: Der numerische Aufwand von DAESOL, DAESOL-II bzw. MATLAB bei der Berechung der Nominaltrajektorien des harmonischen Oszillators für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und verschiedene Werte für TOL.

An diesem Beispiel beobachten wir, wie positiv sich die Monitorstrategie für das Newton-ähnliche Verfaren, die in DAESOL und DAESOL-II implementiert ist, auswirkt. Beide Codes benötigen hier erheblich weniger Matrixzerlegungen als MATLAB.

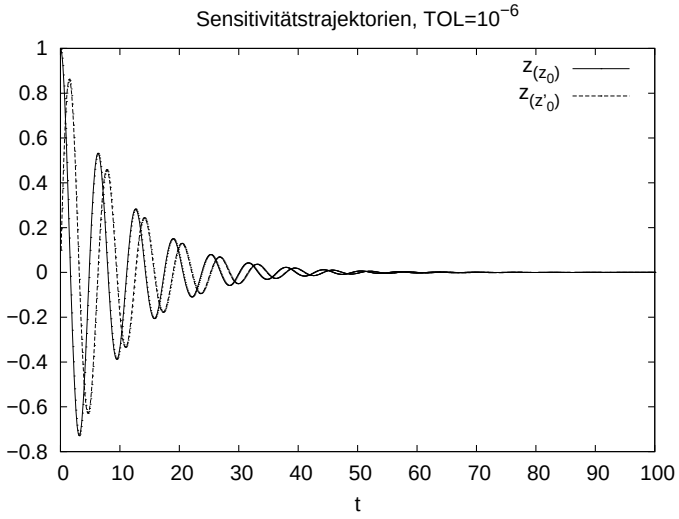


Abbildung 6.13: Numerisch von DAESOL-II mittels der Methode der variierten Trajektorien berechneten Sensitivitäten $z_{(z_0)}(t) \equiv \frac{\partial z(t)}{\partial z_0}$ und $z_{(z_0')}(t) \equiv \frac{\partial z(t)}{\partial \dot{z}_0}$ für die Lösung $z(t)$ des harmonischen Oszillators mit den Werten $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$.

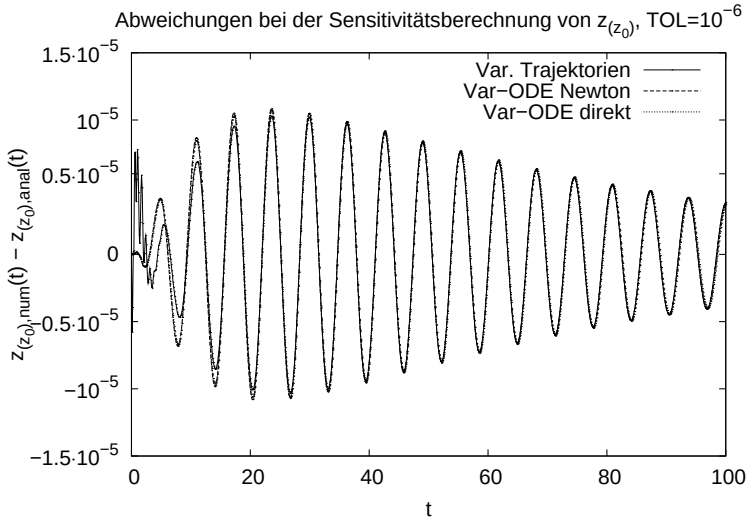


Abbildung 6.14: Abweichungen bei der Berechnung der Sensitivität $z_{(z_0)}(t) \equiv \frac{\partial z(t)}{\partial z_0}$ mit DAESOL-II mittels der implementierten Varianten von der analytisch erhaltenen Sensitivität der Lösung $z(t)$ des harmonischen Oszillators nach dem Anfangswert z_0 für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$.

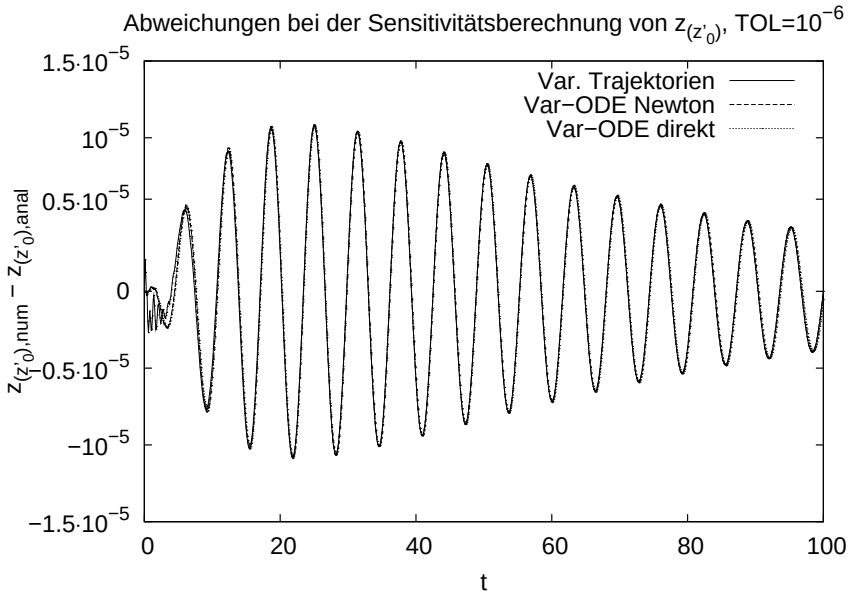


Abbildung 6.15: Abweichungen bei der Berechnung der Sensitivität $z_{(z_0')}(t) \equiv \frac{\partial z(t)}{\partial \dot{z}_0}$ mit DAESOL-II mittels der implementierten Varianten von der analytisch erhaltenen Sensitivität der Lösung $z(t)$ des harmonischen Oszillators nach dem Anfangswert $\dot{z}_0$ für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$.

	Methode	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL	V-ODE Direkt	753	390	382
DAESOL-II	Var. Traj.	530	8	2
DAESOL-II	V-ODE Newton	182	8	176
DAESOL-II	V-ODE Direkt	182	95	89

Tabelle 6.4: Numerischer Aufwand bei der Sensitivitätsberechnung nach z_0 und $\dot{z}_0$ für den harmonischen Oszillator mittels DAESOL bzw. DAESOL-II für $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$.

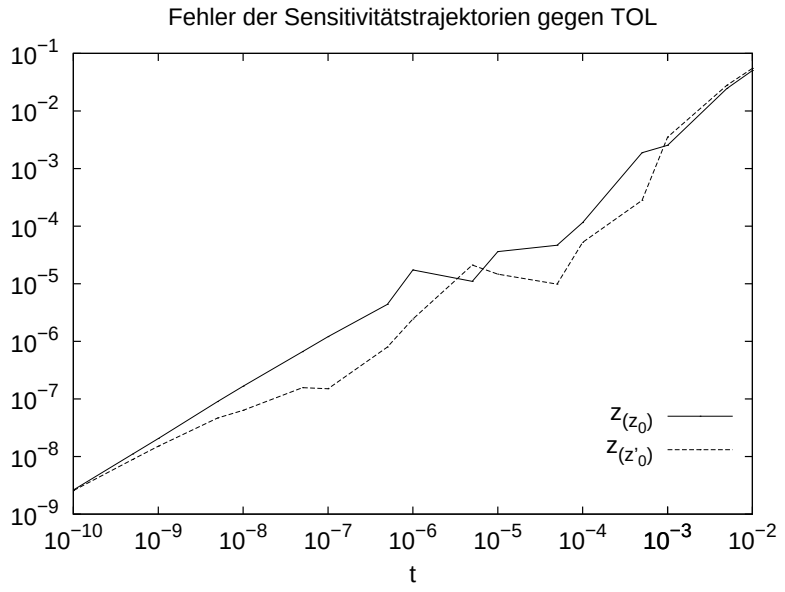


Abbildung 6.16: Die Relation zwischen dem Fehler am Ende des Integrationsintervalls bei der Berechnung der Sensitivitäten $z_{(z_0)}(t) \equiv \frac{\partial z(t)}{\partial z_0}$ bzw. $z_{(z'_0)}(t) \equiv \frac{\partial z(t)}{\partial \dot{z}_0}$ für die Lösung $z(t)$ des harmonischen Oszillators mittels DAESOL bzw. DAESOL-II für die Werte $\gamma = 0.1$, $\omega_0 = 1$, $z_0 = 2$, $\dot{z}_0 = 0$ und vorgegebenen Toleranzen von $TOL = 10^{-10}$ bis $TOL = 10^{-2}$.

Bei den beiden anderen nun folgenden Fällen beschränken wir uns auf die Angabe der von DAESOL-II berechneten Ergebnisse und deren Abweichungen von der jeweiligen analytischen Lösung.

Kriechfall

Hierbei wählen wir $\gamma = 0.5$ und $\omega_0 = 0.2$.

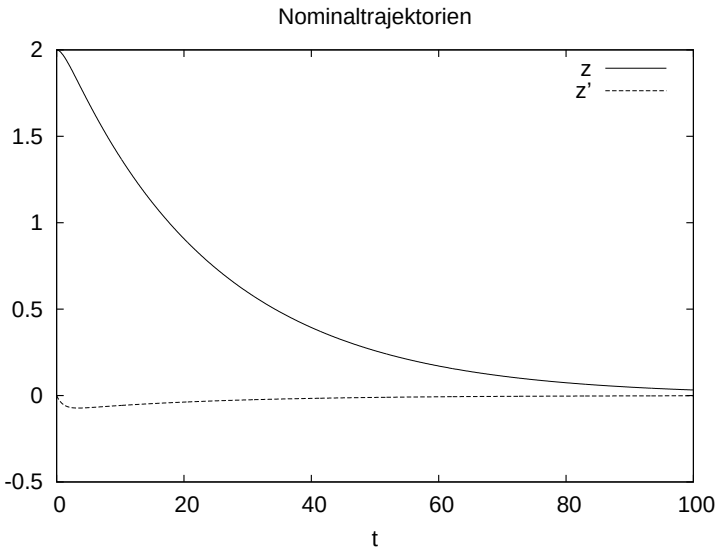


Abbildung 6.17: Numerisch mit DAESOL-II berechnete Nominaltrajektorien des harmonischen Oszillators für $\gamma = 0.5$, $\omega_0 = 0.2$, $z_0 \equiv y_{1,0} = 2$, $z_0' \equiv \dot{z}_0 \equiv y_{2,0} = 0$ und die Toleranz $TOL = 10^{-6}$. Hierbei bezeichnet in der Abbildung $z(t) \equiv y_1(t)$ den Ort des Oszillators beschreibt und $z'(t) \equiv \dot{z}(t) \equiv y_2(t)$ seine Geschwindigkeit.

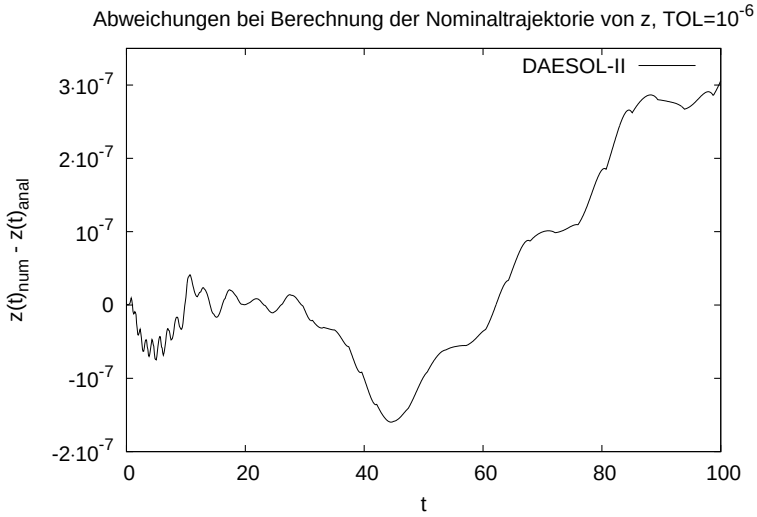


Abbildung 6.18: Abweichungen der mit DAESOL-II berechneten Nominaltrajektorie für $z(t) \equiv y_1(t)$ für den harmonischen Oszillator für die Werte $\gamma = 0.5$, $\omega_0 = 0.2$, $z_0 \equiv y_{1,0} = 2$, $\dot{z}_0 \equiv y_{2,0} = 0$ und die Toleranz $TOL = 10^{-6}$ von der analytischen Lösung über dem Integrationsintervall.

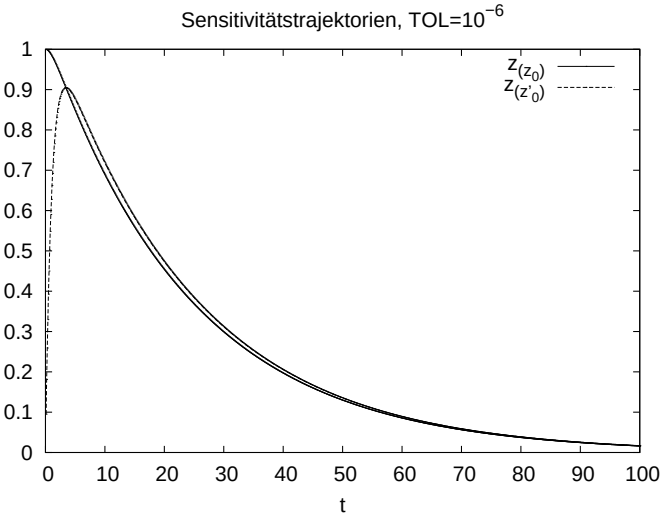


Abbildung 6.19: Numerisch von DAESOL-II mittels Lösung der Var-ODE mit Newton-Verfahren berechnete Sensitivitäten $z_{(z_0)}(t) \equiv \frac{\partial z(t)}{\partial z_0}$ und $z_{(z_0')}(t) \equiv \frac{\partial z(t)}{\partial \dot{z}_0}$ für die Lösung $z(t)$ des harmonischen Oszillators mit den Werten $\gamma = 0.5$, $\omega_0 = 0.2$, $z_0 = 2$, $\dot{z}_0 = 0$ und der Toleranz $TOL = 10^{-6}$.

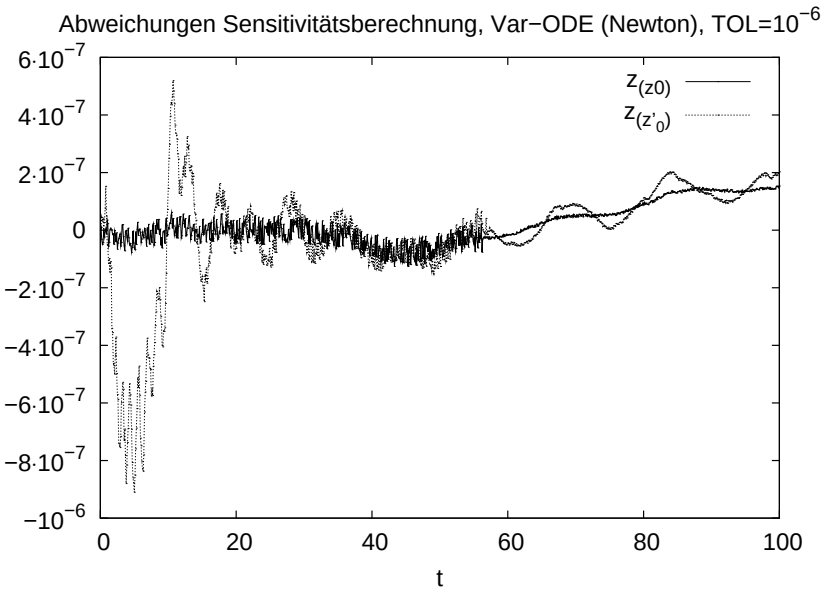


Abbildung 6.20: Abweichungen bei der Berechnung der Sensitivitäten $z_{(z_0)}(t) \equiv \frac{\partial z(t)}{\partial z_0}$ bzw. $z_{(z_0')}(t) \equiv \frac{\partial z(t)}{\partial \dot{z}_0}$ mit DAESOL-II mittels Lösung der Var-ODE mit Newton-Verfahren von der analytisch erhaltenen Sensitivität der Lösung $z(t)$ des harmonischen Oszillators nach dem Anfangswert z_0 bzw. $\dot{z}_0$ für $\gamma = 0.5$, $\omega_0 = 0.2$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$.

Aperiodischer Grenzfall

Hierbei wählen wir $\gamma = 0.2$ und $\omega_0 = 0.2$.

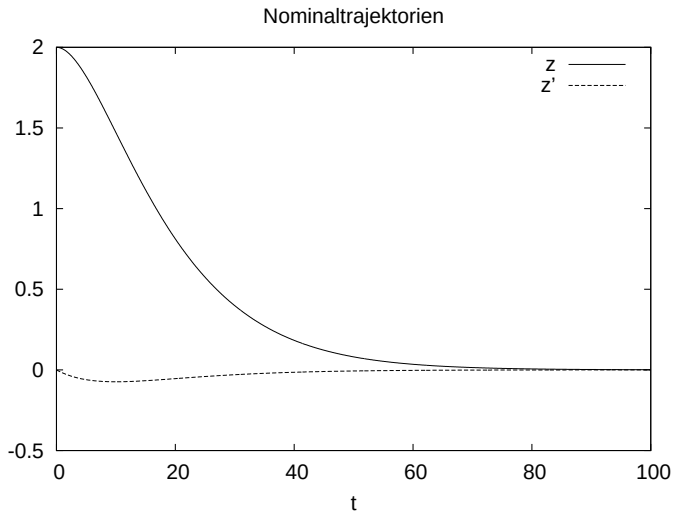


Abbildung 6.21: Numerisch mit DAESOL-II berechnete Nominaltrajektorien des harmonischen Oszillators für $\gamma = 0.2$, $\omega_0 = 0.2$, $z_0 \equiv y_{1,0} = 2$, $z_0' \equiv \dot{z}_0 \equiv y_{2,0} = 0$ und die Toleranz $TOL = 10^{-6}$. Hierbei bezeichnet in der Abbildung $z(t) \equiv y_1(t)$ den Ort des Oszillators beschreibt und $z'(t) \equiv \dot{z}(t) \equiv y_2(t)$ seine Geschwindigkeit.

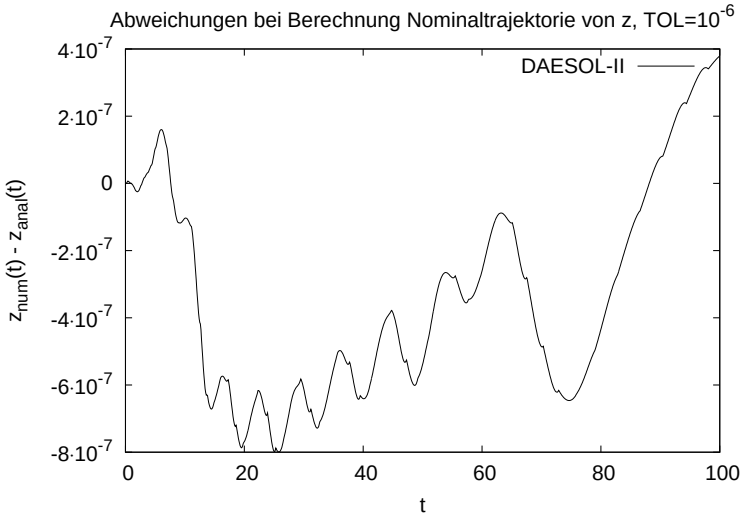


Abbildung 6.22: Abweichungen der mit DAESOL-II berechneten Nominaltrajektorie für $z(t) \equiv y_1(t)$ für den harmonischen Oszillator für die Werte $\gamma = 0.2$, $\omega_0 = 0.2$, $z_0 \equiv y_{1,0} = 2$, $\dot{z}_0 \equiv y_{2,0} = 0$ und die Toleranz $TOL = 10^{-6}$ von der analytischen Lösung über dem Integrationsintervall.

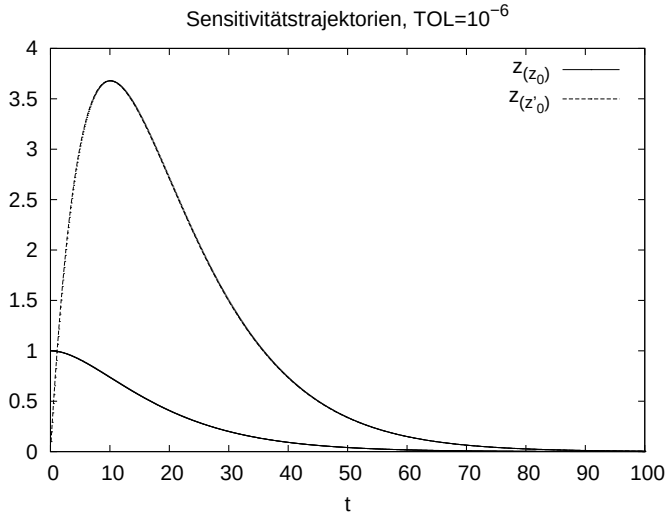


Abbildung 6.23: Numerisch von DAESOL-II mittels direkter Lösung der Var-ODE berechnete Sensitivitäten $z_{(z_0)}(t) \equiv \frac{\partial z(t)}{\partial z_0}$ und $z_{(z_0')}(t) \equiv \frac{\partial z(t)}{\partial \dot{z}_0}$ für die Lösung $z(t)$ des harmonischen Oszillators mit den Werten $\gamma = 0.2$, $\omega_0 = 0.2$, $z_0 = 2$, $\dot{z}_0 = 0$ und der Toleranz $TOL = 10^{-6}$.

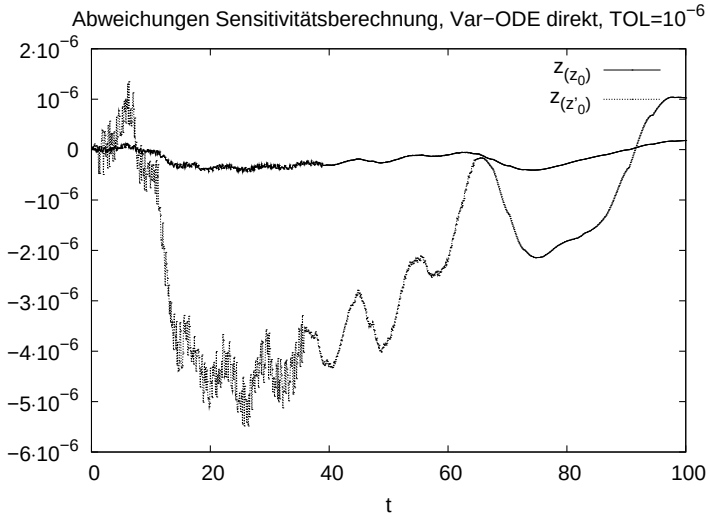


Abbildung 6.24: Abweichungen bei der Berechnung der Sensitivitäten $z_{(z_0)}(t) \equiv \frac{\partial z(t)}{\partial z_0}$ bzw. $z_{(z_0')}(t) \equiv \frac{\partial z(t)}{\partial \dot{z}_0}$ mit DAESOL-II mittels direkter Lösung der Var-ODE von der analytisch erhaltenen Sensitivität der Lösung $z(t)$ des harmonischen Oszillators nach dem Anfangswert z_0 bzw. $\dot{z}_0$ für $\gamma = 0.2$, $\omega_0 = 0.2$, $z_0 = 2$, $\dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-6}$.

6.2 Beispiele aus Problemsammlungen

In diesem Abschnitt präsentieren wir numerische Ergebnisse bei Anwendung von DAESOL-II auf Beispiele aus Testsammlungen. Wir haben dazu Beispiele aus den bekannten (STIFF) DETEST Testsets von Enright, Hull et. al [HEFS72, EHL75], sowie aus dem ODE/DAE Testset von Mazzia und Iavernaro [MI03] an der Universität Bari, früher unterhalten vom CWI in Amsterdam, ausgewählt.

6.2.1 Sparse-Testproblem

Das erste Beispiel ist eine Modifikation des Problem DETEST C4 und dient zur Demonstration der Effizienz des in DAESOL-II integrierten Sparse-Lösers UMFPACK[1] für große dünnbesetzte Probleme. Ursprung des Problems ist die Diskretisierung einer parabolischen partiellen Differentialgleichung, welche auf eine Bandstruktur in der Ableitung der Modellfunktion führt.
Das Problem ist von variabler Dimension n_y, $y = (y_1, \ldots, y_{n_y})^T$ und hat die Gestalt

$$f(t,y) = \begin{pmatrix} -2 & 1 & & & 0 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 0 & & & 1 & -2 \end{pmatrix} y.$$

Als Anfangswert verwenden wir $y_0 = (1, 0, \ldots, 0)^T$ und integrieren auf dem Intervall $I = [0, 20]$ für verschiedene Werte von n_y, jeweils unter Benutzung der ATLAS-LAPACK[1] Routinen bzw. UMFPACK zur Lösung der linearen Gleichungssysteme. Dabei werden in diesem Beispiel die dünnbesetzten Ableitungsmatrizen von Hand analytisch bereitgestellt.

[1] Für Referenzen siehe Kapitel 5

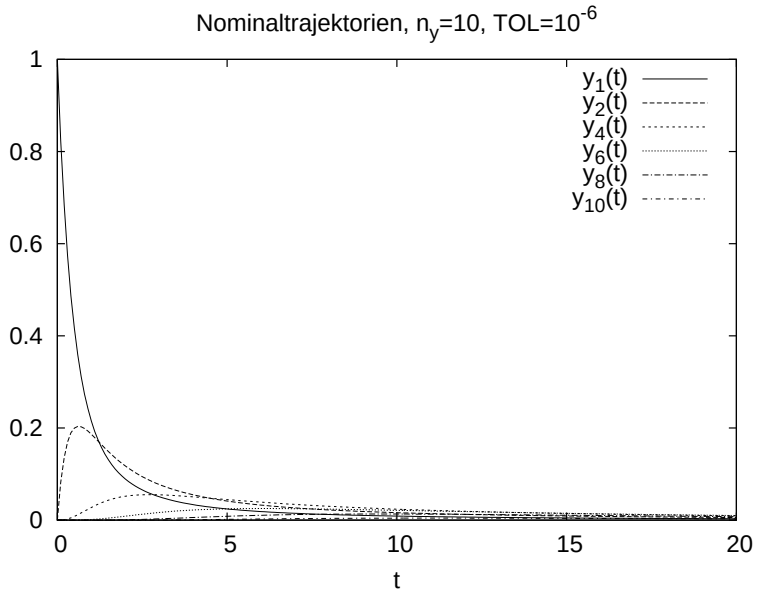


Abbildung 6.25: Einige der numerisch mittels DAESOL-II berechneten Nominaltrajektorien des modifizierten Beispiels DETEST C4 für TOL=10^{-6} und die Dimension $n_y = 10$.

Methode	n_y	CPU-Zeit in s
LAPACK	50	0.03
LAPACK	100	0.07
LAPACK	200	0.32
LAPACK	500	2.58
LAPACK	1000	8.18
UMFPACK	50	0.04
UMFPACK	100	0.08
UMFPACK	200	0.14
UMFPACK	500	0.18
UMFPACK	1000	0.40

Tabelle 6.5: Zeitaufwand zur Berechnung der Nominaltrajektorien des modifizierten Beispiels DETEST C4 mittels DAESOL-II mit UMFPACK bzw. ATLAS-LAPACK zur Lösung der linearen Gleichungssysteme für TOL=10^{-6} und verschiedene Werte für die Dimension n_y.

Wir sehen also, dass bei niedrigeren Problemdimensionen die Sparse-Behandlung aufgrund des dabei entstehenden Overheads keine Vorteile bringt. Bei höheren Dimensionen ist der Geschwindigkeitsvorteil allerdings deutlich bemerkbar. Dabei ist noch anzumerken, dass UMFPACK ursprünglich nicht für Bandmatrizen optimiert und auch nicht auf diese beschränkt ist, sondern allgemeiner unstrukturierte dünnbesetzte Matrizen behandeln kann.

6.2.2 Van-der-Pol Oszillator

Das van-der-Pol Beispiel hat seinen Ursprung in der Elektronik und beschreibt das Verhalten von bestimmten nichtlinearen Schaltkreisen. Die van-der-Pol Gleichung ist eine nichtlineare gewöhnliche Differentialgleichung zweiter Ordnung und hat die Gestalt

$$\ddot{z} + \mu(z^2 - 1)\dot{z} + z = 0$$

mit $z(t) \in \mathbb{R}$ und $\mu > 0, \mu \in \mathbb{R}$. Dabei gewichtet der Parameter μ den nichtlinearen Teil der Gleichung.
Sie hat zwei periodische Lösungen, nämlich die triviale instabile Lösung $z(t) \equiv 0$ und den nichttrivialen sogenannten Grenzzyklus, der in etwa den Anfangswerten $z(0) = 2$, $\dot{z}(0) = 0$ entspricht. Der Name Grenzzyklus rührt daher, dass alle nichttrivialen Lösungen der Gleichung für $t \to \infty$ dagegen konvergieren.
Wenn man das Problem umformuliert in ein System von zwei Differentialgleichungen erster Ordnung, so erhält man mit $y = (y_1, y_2)^T \in \mathbb{R}^2$ die rechte Seite

$$f(t, y) = \begin{pmatrix} y_2 \\ \mu(1 - y_1^2)y_2 - y_1 \end{pmatrix}.$$

Wir lösen das Anfangswertproblem für die Anfangswerte

$$y_0 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

und den Wert $\mu = 1000$ auf dem Intervall $I = [0, 2\mu]$.

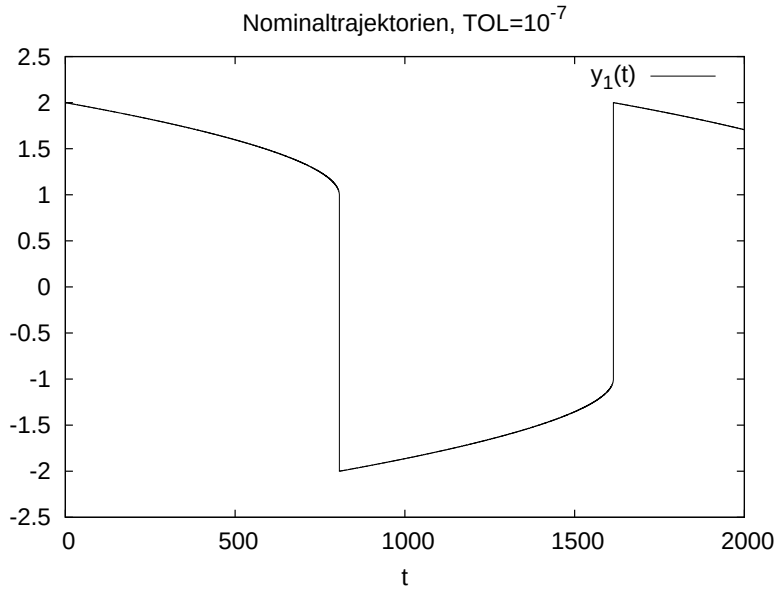


Abbildung 6.26: Numerisch mittels DAESOL-II berechnete Nominaltrajektorie $y_1(t) \equiv z(t)$ des van-der-Pol Beispiels für $\mu = 1000$, die Anfangswerte $y_{1,0} \equiv z_0 = 2$, $y_{2,0} \equiv \dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-7}$.

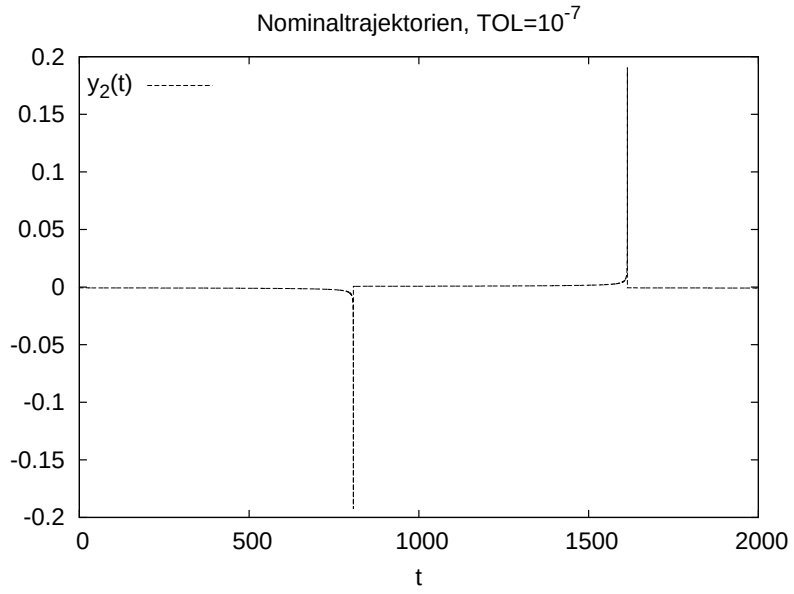


Abbildung 6.27: Numerisch mittels DAESOL-II berechnete Nominaltrajektorie $y_2(t) \equiv \dot{z}(t)$ des van-der-Pol Beispiels für $\mu = 1000$, die Anfangswerte $y_{1,0} \equiv z_0 = 2$, $y_{2,0} \equiv \dot{z}_0 = 0$ und die Toleranz $TOL = 10^{-7}$.

	TOL	#Steps	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL	10^{-4}	405	1644	226	56
DAESOL	10^{-7}	941	2997	257	57
DAESOL-II	10^{-4}	340	1105	315	52
DAESOL-II	10^{-7}	1009	3035	447	72

Tabelle 6.6: Numerischer Aufwand zur Berechnung der Nominaltrajektorien des van-der-Pol Beispiels mittels DAESOL bzw. DAESOL-II für $\mu = 1000$, die Anfangswerte $y_{1,0} \equiv z_0 = 2$, $y_{2,0} \equiv \dot{z}_0 = 0$ und verschiedene Toleranzen.

6.2.3 Chemisches Akzo Nobel Beispiel

Das chemische Akzo Nobel Beispiel ist ein Testproblem aus der Akzo Nobel Forschungszentrale. Es beschreibt eine chemische Reaktion in dem zwei chemische Spezies, FLB[1] und ZHU[1], unter kontinuierlicher Zugabe von Kohlendioxid gemischt werden. Dabei ist die Produktspezies ZLA[1] von Interesse. Das Reaktionsschema ist gegeben durch

$$\begin{aligned}
2\,FLB + 0.5\,CO_2 &\xrightarrow{k_1} FLBT + H_2O \\
ZLA + FLB &\underset{k_2}{\overset{k_2/K}{\rightleftharpoons}} FLBT + ZHU \\
FLB + 2ZHU + CO_2 &\xrightarrow{k_3} LB + Nitrat \\
FLB.ZHU + 0.5CO_2 &\xrightarrow{k_4} ZLA + H_2O \\
FLB + ZHU &\rightleftharpoons FLB.ZHU\ .
\end{aligned}$$

Wir haben $y = (y_1, \ldots, y_5)^T \in \mathbb{R}^5$, $z = z_1 \in \mathbb{R}$ und formulieren die Konzentration der einzelnen Spezies als

$$\begin{array}{lcl|lcl}
y_1 &=& [FLB] & y_4 &=& [ZHU] \\
y_2 &=& [CO_2] & y_5 &=& [ZLA] \\
y_3 &=& [FLBT] & z_1 &=& [FLB.ZHU]
\end{array}$$

[1] von Akzo Nobel anonymisiert

sowie die Reaktionsgeschwindigkeiten $r_1, \ldots, r_5 \in \mathbb{R}$ und den Zufluß von Kohlendioxid pro Volumeneinheit F_{in}

$$\begin{array}{lcl|lcl} r_1 & = & k_1 y_1^4 y_2^{\frac{1}{2}} & r_4 & = & k_3 y_1 y_4^2 \\ r_2 & = & k_2 y_3 y_4 & r_5 & = & k_4 z_1^2 y_2^{\frac{1}{2}} \\ r_3 & = & \frac{k_2}{K} y_1 y_5 & F_{in} & = & klA \cdot \left(\frac{p(CO_2)}{H} - y_2\right), \end{array}$$

wobei klA der Massentransferkoeffizient, H die Henrykonstante und $p(CO_2)$ der Partialdruck von Kohlendioxid ist, der unabhängig von $[CO_2]$ angenommen wird.
Wir erhalten dann folgendes DAE-System von Index 1 mit 5 differentiellen und einer algebraischen Gleichung:

$$f(t,y,z) = \begin{pmatrix} -2r_1 & +r_2 & -r_3 & -r_4 & & \\ -0.5\, r_1 & & & -r_4 & -0.5\, r_5 & +F_{in} \\ r_1 & -r_2 & +r_3 & & & \\ & -r_2 & +r_3 & -2r_4 & & \\ & r_2 & -r_3 & & +r_5 & \end{pmatrix}$$

$$g(t,y,z) = \left(K_s y_1 y_4 - z_1 \right).$$

Die Werte der Reaktionsraten und der anderen Parameter und Konstanten sind gegeben durch

$$\begin{array}{lcl|lcl|lcl} k_1 & = & 18.7 & k_4 & = & 0.42 & K_s & = & \frac{[FLB.ZHU]}{[FLB]\cdot[ZHU]} = 115.83 \\ k_2 & = & 0.58 & K & = & 34.4 & p(CO_2) & = & 0.9 \\ k_3 & = & 0.09 & klA & = & 3.3 & H & = & 737. \end{array}$$

Wir integrieren das mit den folgenden konsistenten Anfangskonzentrationen

$$\begin{array}{lcl|lcl} y_1(0) & = & 0.444 & y_4(0) & = & 0.007 \\ y_2(0) & = & 0.00123 & y_5(0) & = & 0 \\ y_3(0) & = & 0 & z_1(0) & = & K_s y_1(0) y_4(0) \end{array}$$

steife Anfangswertproblem auf dem Intervall $I = [0, 180]$.

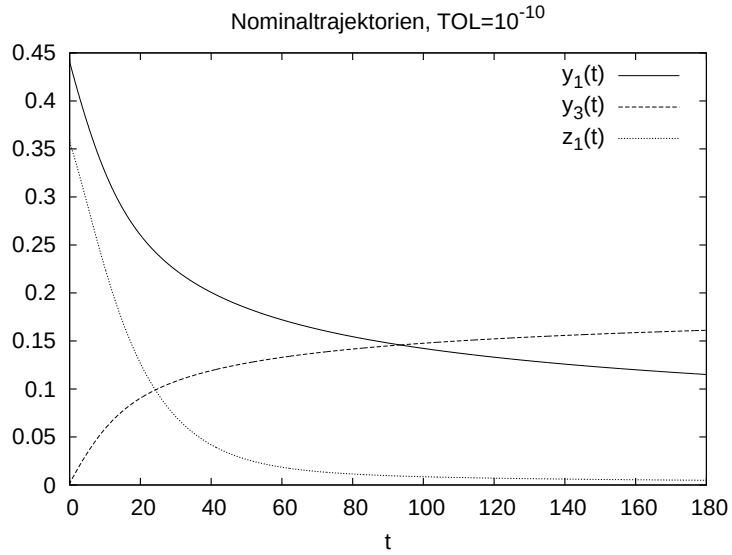


Abbildung 6.28: Numerisch mittels DAESOL-II berechnete Nominaltrajektorien der Konzentrationen $y_1(t) \equiv [FLB](t)$, $y_3(t) \equiv [FLBT](t)$, $z_1(t) \equiv [FLB.ZHU](t)$ für das chemische Akzo-Nobel Beispiel mit der Toleranz $TOL = 10^{-10}$ und konsistenten Anfangswerten.

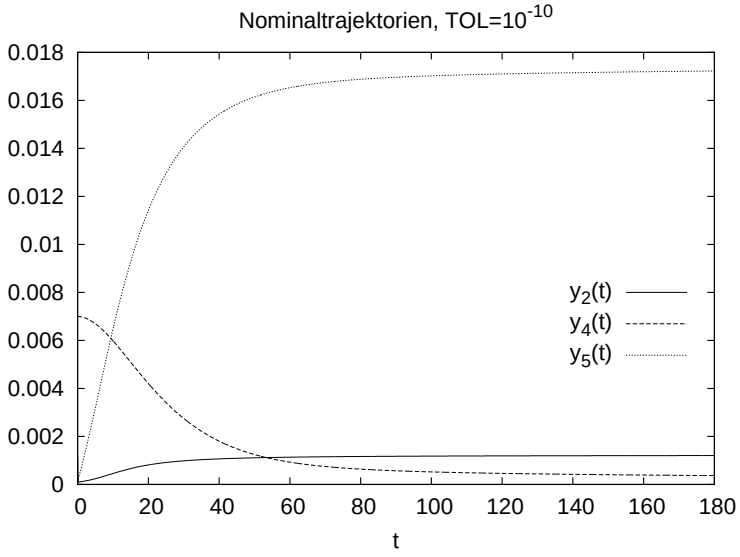


Abbildung 6.29: Numerisch mittels DAESOL-II berechnete Nominaltrajektorien der Konzentrationen $y_2(t) \equiv [CO_2](t)$, $y_4(t) \equiv [ZHU](t)$, $y_5(t) \equiv [ZLA](t)$ für das chemische Akzo-Nobel Beispiel mit der Toleranz $TOL = 10^{-10}$ und konsistenten Anfangswerten.

	TOL	#Steps	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL-II	10^{-6}	147	347	28	6
DAESOL-II	10^{-8}	244	570	43	6
DAESOL-II	10^{-10}	420	1013	31	6

Tabelle 6.7: Numerischer Aufwand zur Berechnung der Nominaltrajektorien des chemischen Akzo Nobel Beispiels mittels DAESOL-II für verschiedene Toleranzen TOL.

6.2.4 Oregonator

Der Oregonator ist Modell für die bekannte Belousov-Zhabotinskii Reaktion (BZ-Reaktion). Wenn bestimmte Spezies, unter anderem Bromige Säure, Bromid-Ionen und Cerium-Ionen kombiniert werden, so läuft eine chemische Reaktion ab, die nach einer inaktiven Anfangsphase unter Änderungen in Struktur und der Farbe von rot nach blau und umgekehrt oszilliert. Diese Farbänderungen werden durch Änderungen der Oxidationszahl des Ceriums durch abwechselnde Oxidation und Reduktion verursacht.
Field, Körös und Noyes [FKN72] formulierten das Oregonator-Modell, das die wichtigsten Teile des Prozesses beschreibt, der für die Oszillationen verantwortlich ist.
Der Mechanismus kann zusammengefaßt werden zu den folgenden drei konkurrierenden Abläufen:

- Die Reduktion von Bromat zu Brom.
- Den Anstieg der Konzentration von Hypobromiger Säure mit sich beschleunigender Rate und der Produktion von Ce(IV). Hierbei tritt der plötzliche Farbwechsel von rot nach blau auf.
- Die Reduktion von Ce(IV) zu Ce(III). Hierbei haben wir einen Farbwechsel von blau nach rot.

Von diesem Mechanismus kann man folgendes Reaktionsschema für den Oregonator ableiten

$$\begin{aligned} BrO_3^- + Br^- &\xrightarrow{k_3} HBrO_2 + HOBr \\ HBrO_2 + Br^- &\xrightarrow{k_2} 2HOBr \\ BrO_3^- + HBrO_2 &\xrightarrow{k_5} 2HBrO_2 + 2CE(IV) \\ 2HBrO_2 &\xrightarrow{k_4} BrO_3^- + HOBr \\ Org + CE(IV) &\xrightarrow{k_c} 0.5Br^- \end{aligned}$$

wobei Org alle oxidierbaren Spezies beinhaltet. Dies führt auf das folgendende Differentialgleichungssystem mit 3 Gleichungen.
Mit $y = (y_1, y_2, y_3)^T \in \mathbb{R}^3$ und $y_1 = [HBrO_2]$, $y_2 = [Br^-]$ sowie
$y_3 = [CE(IV)]$ erhalten wir

$$f(t,y) = \begin{pmatrix} s(y_2 - y_1 y_2 + y_1 - q y_1^2) \\ \frac{1}{s}(-y_2 - y_1 y_2 + y_3) \\ w(y_1 - y_3) \end{pmatrix}$$

mit den Parametern

$$s = 77.27, \quad w = 0.161, \quad q = 8.375 \cdot 10^{-6},$$

und den Anfangswerten $y_0 = (1, 2, 3)^T$. Wir integrieren auf dem Intervall $I = [0, 400]$.

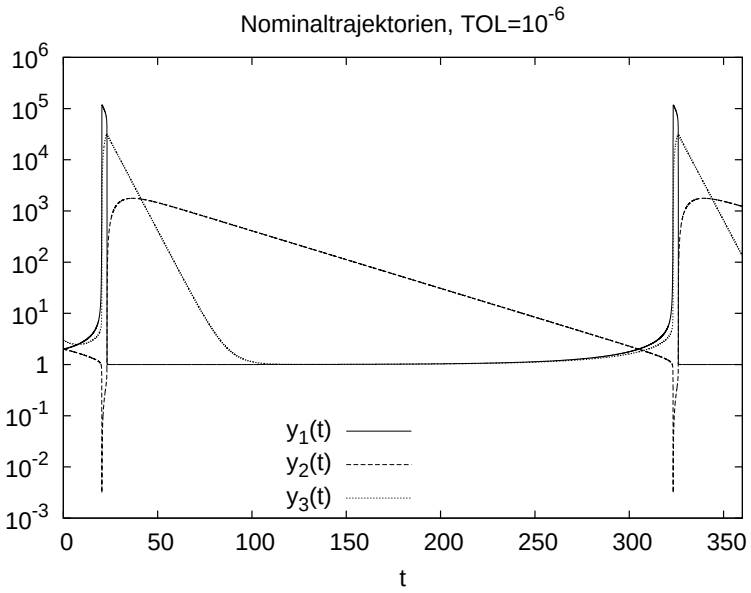


Abbildung 6.30: Numerisch mittels DAESOL-II berechnete Nominaltrajektorien der Konzentrationen $y_1(t) \equiv [HBrO_2](t)$, $y_2(t) \equiv [BR^-](t)$, $y_3(t) \equiv [CE(IV)](t)$ für das Oregonator Beispiel mit der Toleranz $TOL = 10^{-6}$ und den Anfangswerten $y_{1,0} = 1$, $y_{2,0} = 2$ und $y_{3,0} = 3$.

	TOL	#Steps	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL-II	10^{-6}	813	2754	715	182
DAESOL-II	10^{-8}	1425	4463	709	133
DAESOL-II	10^{-10}	2600	7494	708	111

Tabelle 6.8: Numerischer Aufwand zur Berechnung der Nominaltrajektorien des Oregonator Beispiels mittels DAESOL-II für verschiedene Toleranzen TOL und die Anfangswerte $y_{1,0} = 1$, $y_{2,0} = 2$ und $y_{3,0} = 3$.

6.3 Destillationskolonne

Das folgende Beispiel betrifft die Simulation eines Destillationsprozesses, der dazu dient, ein Gemisch aus Methanol und n-Propanol mit sehr hoher Reinheit zu trennen. Ursprünglich ist es als ein Echtzeit-Optimalsteurungsproblem formuliert, bei dem das Ziel der Steuerung des Destillationsprozeses eine möglichst hohe Reinheit des Destillats und des Bodenproduktes ist. Für mehr

Details hierzu verweisen wir auf die Arbeit von Diehl [Die01]. Wir betrachten im Folgenden nur die Simulation des Prozesses für festgelegte konstante Kontrollen.
Die betrachtete Destillationskolonne besteht aus einem Erhitzer, einem wassergekühlten Kondensierer und 40 Böden.
Die Wärmezufuhr zum Erhitzer bezeichnen wir mit Q und die Rückflussrate aus dem Kondensierer in die Kolonne mit L_{vol}.
Wir numerieren die 40 Böden von unten nach oben durch und beziehen uns mit dem Index $l = 1, \ldots, 40$ auf den entsprechenden Boden, wobei der Boden $l = 20 = N_F$ derjenige ist, bei dem das zu trennende Gemisch zugeführt wird. Mit Index $l = 0$ bezeichnen wir den Erhitzer, mit $l = 41$ den Kondensierer.
Die Temperaturen an den entsprechenden Stellen bezeichnen wir mit $T_l, l = 0, \ldots, 41$, die Konzentrationen von flüssigem Methanol mit $X_l, l = 0, \ldots, 41$. Da wir nur ein Gemisch aus 2 Substanzen haben, sind die Konzentrationen von n-Propanol dadurch auch eindeutig bestimmt.
Den molaren Dampf- bzw. Flüssigkeitsfluss aus jedem Boden bezeichnen wir mit $V_l, l = 0, \ldots, 40$ bzw. $L_l, l = 1, \ldots, 41$, und ferner mit B den Abfluss des flüssigen Bodenprodukts aus dem Erhitzer, mit D den des Destillats aus dem Kondensierer und mit $n_l, l = 1 \ldots, 40$ schließlich die molaren Holdups der Böden.
Wir erhalten dann ein DAE-Modell von Index 1 für die Destillationskolonne mit 82 differentiellen Variablen und 122 algebraischen Variablen, 2 (konstanten) Kontrollen und 17 Parametern der Form

$$\begin{aligned} \dot{x}(t) &= f(x(t), z(t), p, q) \\ 0 &= g(x(t), z(t), p, q). \end{aligned}$$

Dabei bestehen $x \in \mathbb{R}^{82}$, $z \in \mathbb{R}^{122}$ bzw. $u \in \mathbb{R}^2$ aus den Komponenten

$$\begin{aligned} x &= (X_0, \ldots, X_{41}, n_1, \ldots, n_{40})^T \\ z &= (L_1, \ldots, L_{40}, V_1, \ldots, V_{40}, T_0, \ldots, T_{41})^T \\ u &= (L_{vol}, Q)^T. \end{aligned}$$

Wir integrieren auf dem Intervall $I = [0, 900]$ mit auf dem Intervall als konstant angenommenen Kontrollen $Q \approx 4.078$ und $L_{vol} \approx 2.681$ und Anfangswerten aus dem Betrieb der Destillationskolonne nach einer kurzen Störung

des normalen Ablaufs. Im Folgenden stellen wir einige Komponenten der Lösung sowie die dazugehörigen Sensitivitäten nach den Kontrollen dar.

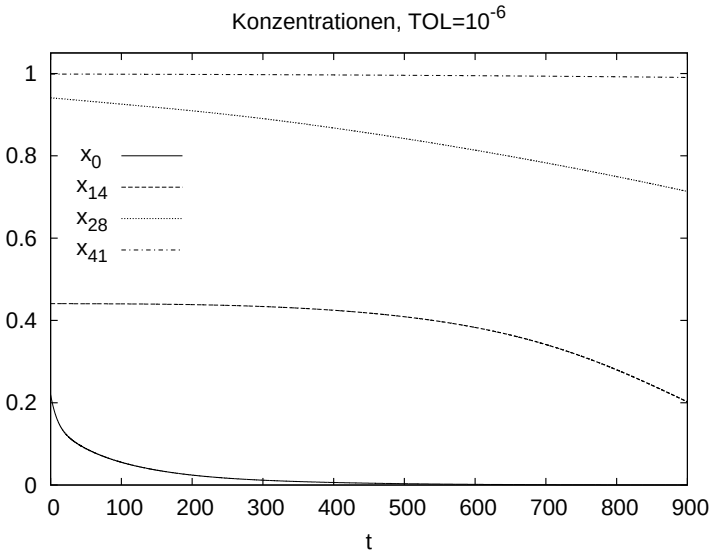


Abbildung 6.31: Die mittels DAESOL-II berechneten Konzentrationen von Methanol im Erhitzer (x_0), im Kondensierer (x_{41}), sowie auf den Böden 14 und 28 (x_{14}, x_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

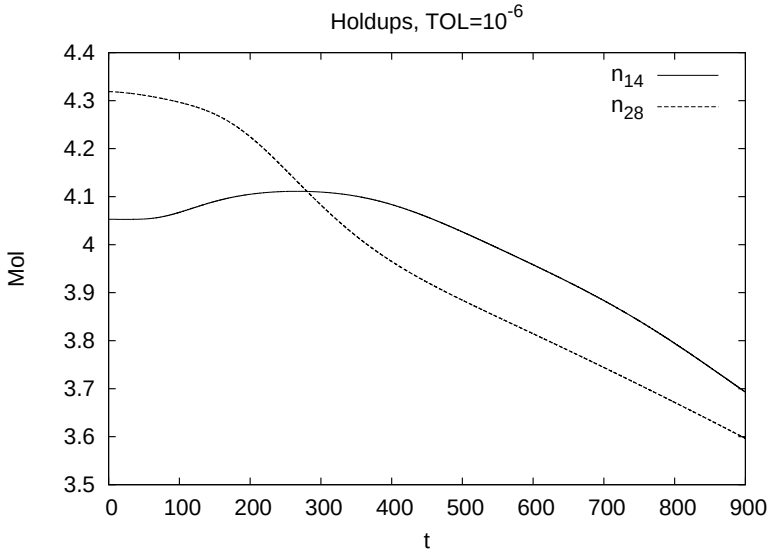


Abbildung 6.32: Die mittels DAESOL-II berechneten molekularen Holdups von den Böden 14 und 28 (n_{14}, n_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

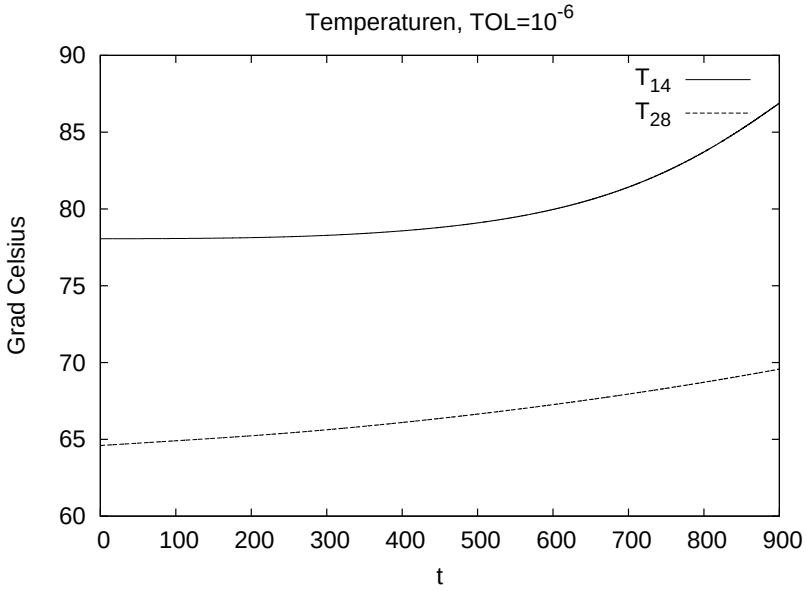


Abbildung 6.33: Die mittels DAESOL-II berechneten Temperaturen auf den Böden 14 und 28 (T_{14}, T_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

	TOL	#Steps	#Eval. f	#Decomp. Jac	#Deriv. f
DAESOL-II	10^{-6}	128	328	16	5
DAESOL-II	10^{-8}	181	445	16	5
DAESOL-II	10^{-10}	305	751	10	3

Tabelle 6.9: Numerischer Aufwand zur Berechnung der Nominaltrajektorien bei der Simulation des Destillationsprozesses mittels DAESOL-II für verschiedene Toleranzen TOL.

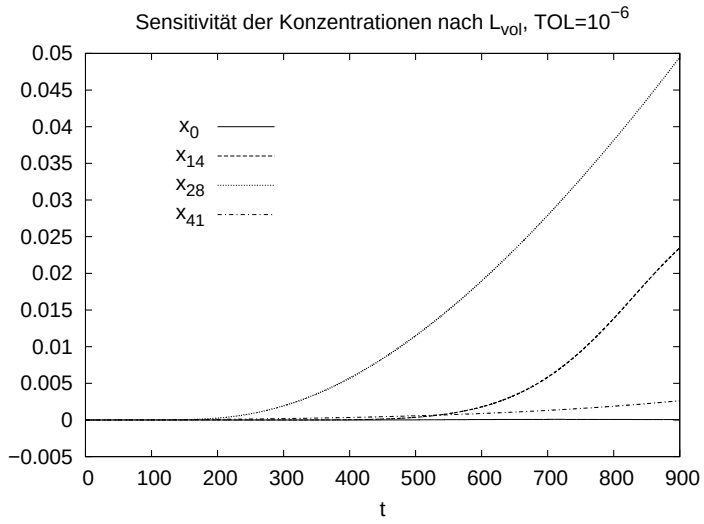


Abbildung 6.34: Die mittels DAESOL-II berechneten Sensitivitäten nach der Rückflussrate L_{vol} der Konzentrationen von Methanol im Erhitzer (x_0), im Kondensierer (x_{41}), sowie auf den Böden 14 und 28 (x_{14}, x_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

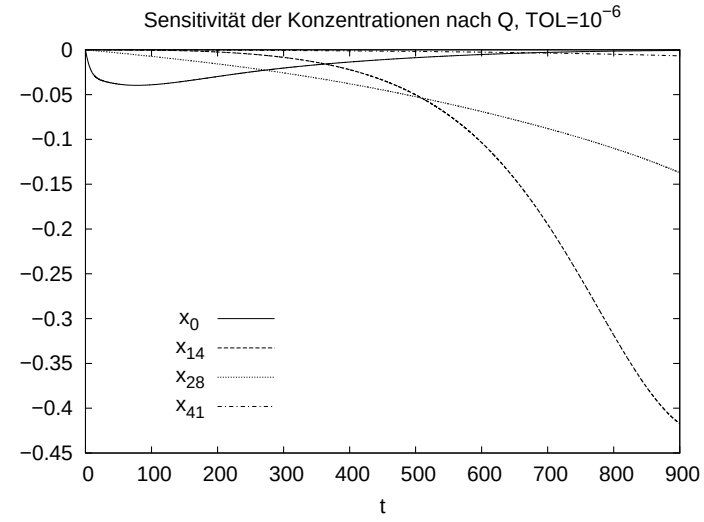


Abbildung 6.35: Die mittels DAESOL-II berechneten Sensitivitäten nach der Wärmezufuhr Q der Konzentrationen von Methanol im Erhitzer (x_0), im Kondensierer (x_{41}), sowie auf den Böden 14 und 28 (x_{14}, x_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

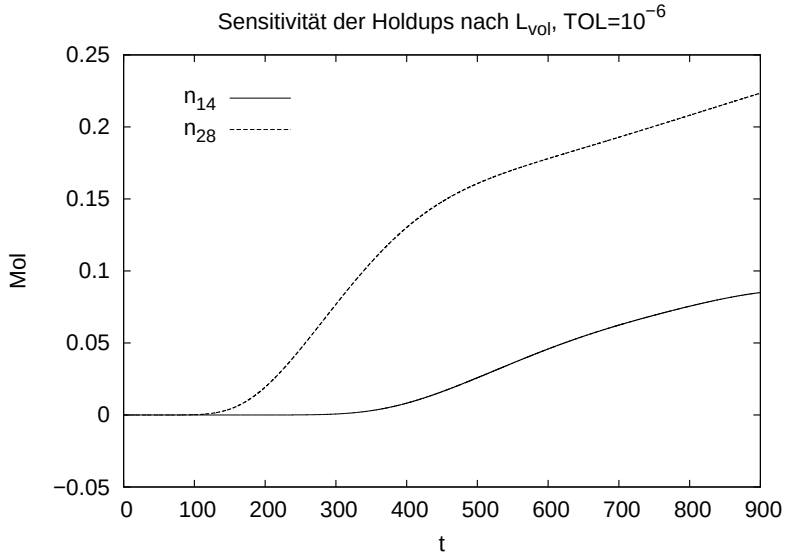


Abbildung 6.36: Die mittels DAESOL-II berechneten Sensitivitäten nach der Rückflussrate L_{vol} der molekularen Holdups von den Böden 14 und 28 (n_{14}, n_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

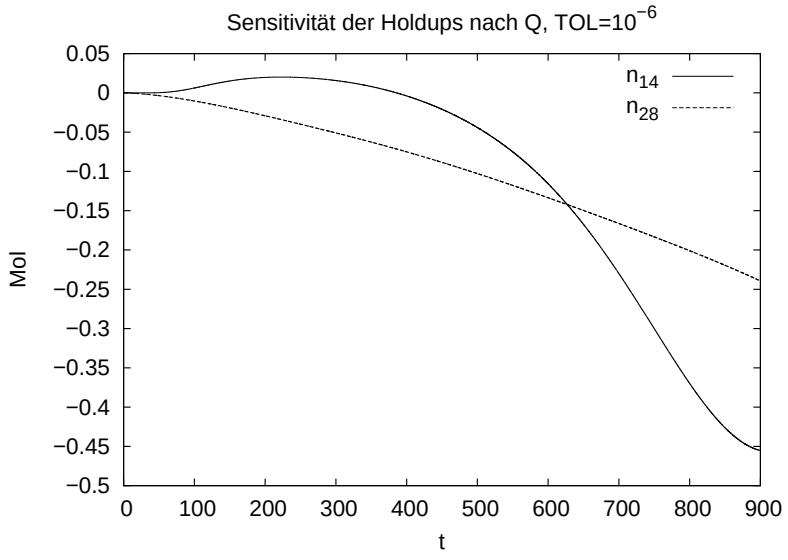


Abbildung 6.37: Die mittels DAESOL-II berechneten Sensitivitäten nach der Wärmezufuhr Q der molekularen Holdups von den Böden 14 und 28 (n_{14}, n_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

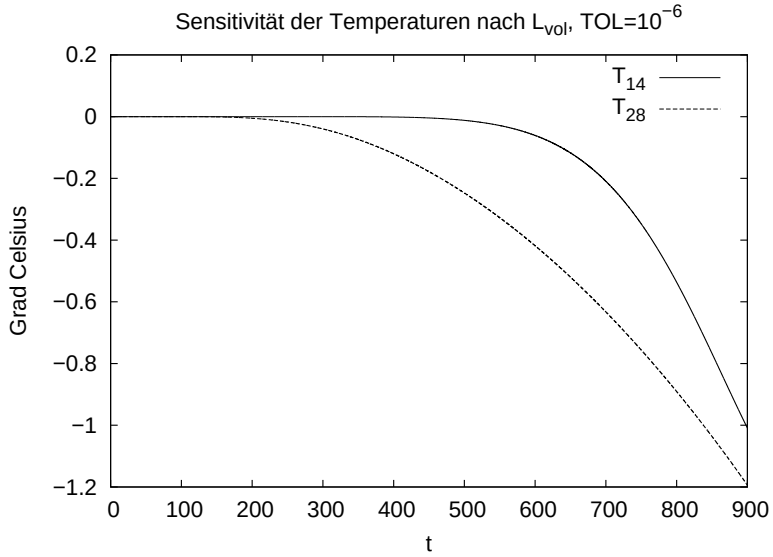


Abbildung 6.38: Die mittels DAESOL-II berechneten Sensitivitäten nach der Rückflussrate L_{vol} der Temperaturen auf den Böden 14 und 28 (T_{14}, T_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

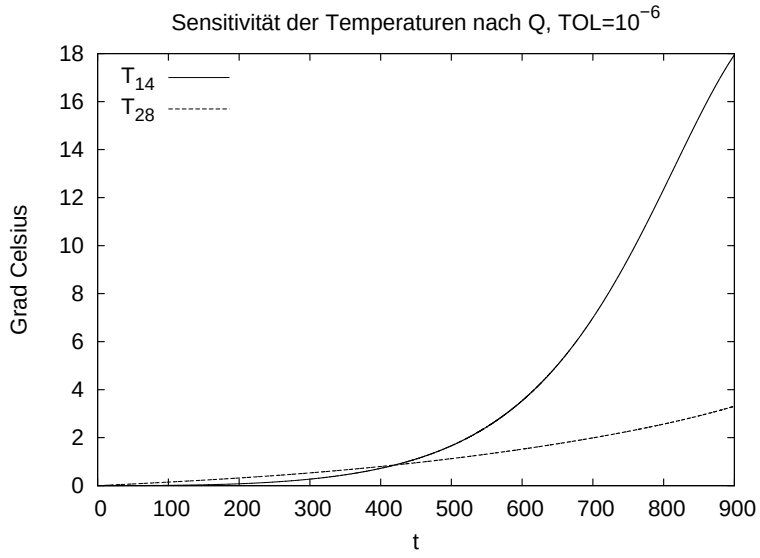


Abbildung 6.39: Die mittels DAESOL-II berechneten Sensitivitäten nach der Wärmezufuhr Q der Temperaturen auf den Böden 14 und 28 (T_{14}, T_{28}) der Destillationskolonne für eine Toleranz von $TOL = 10^{-6}$.

Methode	#Eval. f	#Decomp. Jac	#Deriv. f
Var. Traj.	958	16	5
V-DAE Newton	328	16	320
V-DAE Direkt	328	142	131

Tabelle 6.10: Numerischer Aufwand bei der Sensitivitätsberechnung der Lösungen des Destillationsprozesses nach der Rückflussrate L_{vol} und der Wärmezufuhr Q mittels DAESOL-II mit den verschiedenen implementierten Varianten für ein Toleranz von $TOL = 10^{-6}$.

Bei der Betrachtung der Sensitivitäten kann man erkennen, dass die durch die Änderungen an den Steuerungen verursachten Lösungsänderungen bei den Holdups und Temperaturen sich nach einer zeitlichen Verzögerung zuerst bzw. am intensivsten an den der Störung nahegelegensten Böden auswirken. Bei den Konzentrationen ist das Verhalten komplexer und kann nicht so einfach erklärt werden.

6.4 Peroxidase-Oxidase-Reaktion

Die Peroxidase-Oxidase-Reaktion (PO-Reaktion) beschreibt die Oxidation von NADH mittels molekularem Sauerstoff mit dem Enzym Peroxidase als Katalysator. Sie ist Gegenstand vieler aktueller Untersuchungen, weil sie als Beispiel nichtlinearer Dynamik ein sehr komplexes dynamisches Verhalten zeigt, dass je nach Reaktionsbedingungen aus periodischen oder chaotischen Oszillationen oder auch dem Übergang in einen stationären Zustand bestehen kann.
Die Gesamtreaktion besteht aus einer Anzahl von elementare Einzelreaktionen, die gut untersucht und deren kinetisches Verhalten hinreichend gut bekannt ist. Es läßt sich folgender detaillierter Reaktionsmechanismus angeben. Für eine detaillierte Beschreibung hierzu, insbesondere die kinetischen Parameter und die Beschreibung der einzelnen Spezies, verweisen wir auf [HKLO01] und Standardlehrbücher der Biochemie.

$$
\begin{array}{rcl|rcl}
NADH + O_2 + H^+ & \longrightarrow & NAD^+ + H_2O_2 & 2NAD & \longrightarrow & NAD_2 \\
H_2O_2 + Per^{3+} & \longrightarrow & coI & Per^{3+} + NAD & \longrightarrow & Per^{2+}NAD^+ \\
coI + NADH & \longrightarrow & coII + NAD & Per^{2+} + O_2 & \longrightarrow & coIII \\
coII + NADH & \longrightarrow & Per^{3+} + NAD & & \longrightarrow & NADH \\
NAD + O_2 & \longrightarrow & NAD^+ + O_2^- & O_{2_{gas}} & \longrightarrow & O_{2_{fl}} \\
O_2^- + Per^{3+} & \longrightarrow & coIII & O_{2_{fl}} & \longrightarrow & O_{2_{gas}} \\
2O_2^- + 2H^+ & \longrightarrow & H_2O_2 + O_2 & Enz_{inakt} + O_2^- & \longrightarrow & Enz_{akt} \\
coIII + NAD & \longrightarrow & coI + NAD^+ & Enz_{act} & \longrightarrow & Enz_{inakt}
\end{array}
$$

Durch stöchiometrische Analyse und die Vermeidung der expliziten Modellierung der reinen Produkte des Systems, kann man das System auf 10 Zustandsvariablen $y_1, \ldots, y_{10}$ reduzieren. Dies führt dann schließlich auf ein ODE-System mit der folgenden rechten Seite [ZLK+05]:

$$f(t,y) = \begin{pmatrix} -k_1y_1y_2 - k_3y_7y_1 - k_4Dy_1 + k_{12} \\ -k_1y_1y_2 - k_5y_5y_2 + k_7y_6^2 - k_{11}y_4y_2 + k_{13} - k_{16}y_2 \\ -k_2y_3y_9 + k_4Dy_1 - k_6y_6y_3 - k_{10}y_3y_5 \\ k_{10}y_3y_5 - k_{11}y_4y_2 \\ (k_3y_7 + k_4D)y_1 - y_5(k_5y_2 - k_8y_8 - 2k_9y_5 - k_{10}y_3) \\ k_5y_5y_2 - k_6y_6y_3 - 2k_7y_6^2 \\ k_2y_3y_9 - k_3y_7y_1 + k_8y_8y_5 \\ k_6y_6y_3 - k_8y_8y_5 + k_{11}y_4y_2 \\ k_1y_1y_2 - k_2y_3y_9 + k_7y_6^2 \\ k_14y_6^5k_{17}^5 + y_6^5 - k_{15}y_{10} \end{pmatrix}$$

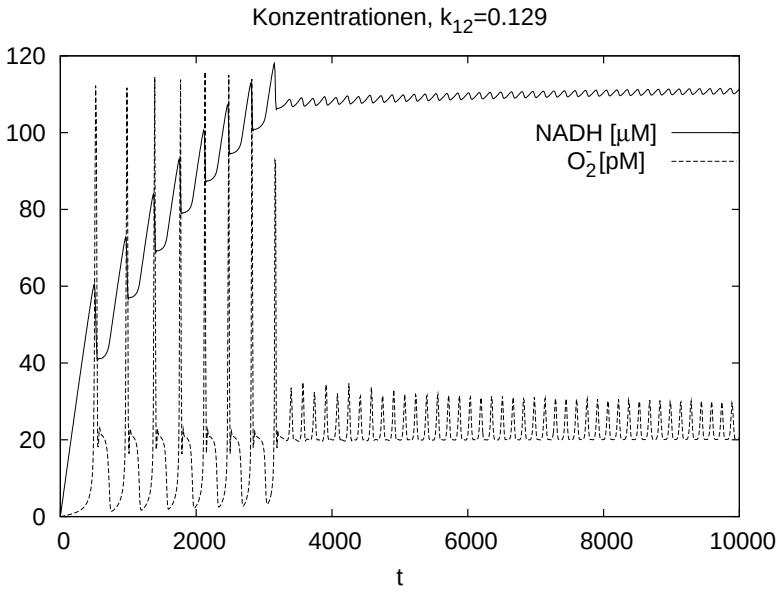
mit $D := POg - (y_3 + y_4 + y_7 + y_8)$ und den Konzentrationen der Spezies

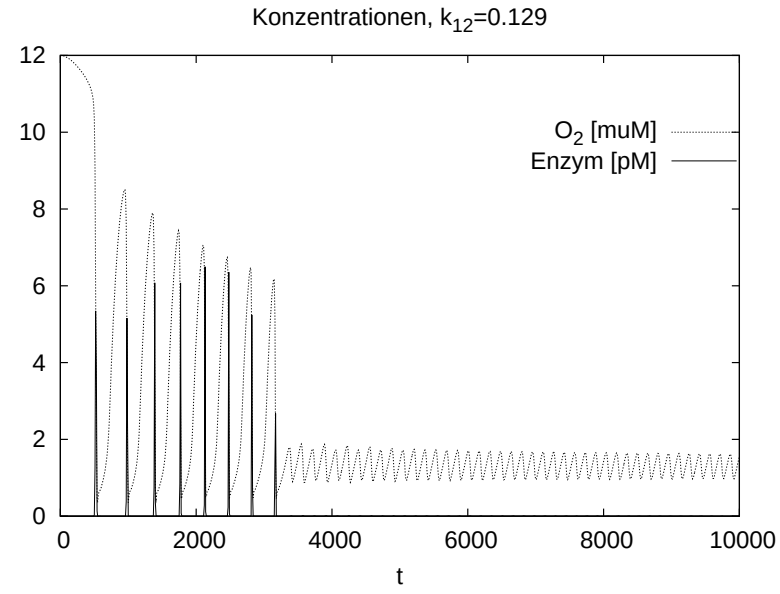
y_1	=	$[NADH]$	y_6	=	$[O_2^-]$
y_2	=	$[O_2]$	y_7	=	$[coI]$
y_3	=	$[Per^{3+}]$	y_8	=	$[coIII]$
y_4	=	$[Per^{2+}]$	y_9	=	$[H_2O_2]$
y_5	=	$[NAD]$	y_{10}	=	$[Enz]$

und den Parametern und Konstanten

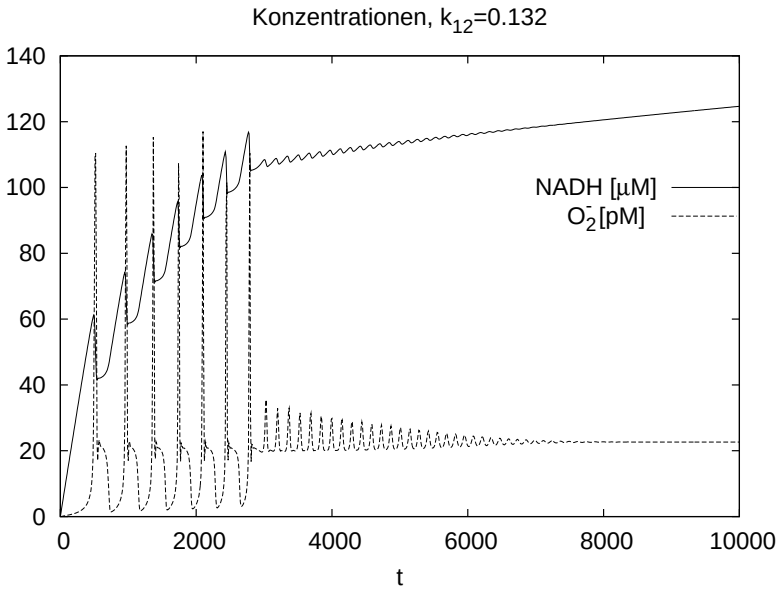
k_1	=	$3 \cdot 10^{-6}$,	k_7	=	20.0,	k_{13}	=	0.072,
k_2	=	18.0,	k_8	=	110.0,	k_{14}	=	0.005,
k_3	=	0.04,	k_9	=	56.0,	k_{15}	=	1.6,
k_4	=	0.026,	k_{10}	=	1.8,	k_{16}	=	0.006,
k_5	=	20.0,	k_{11}	=	0.1,	k_{17}	=	0.4,
k_6	=	1.7,	k_{12}	=	variabel,	POg	=	1.5.

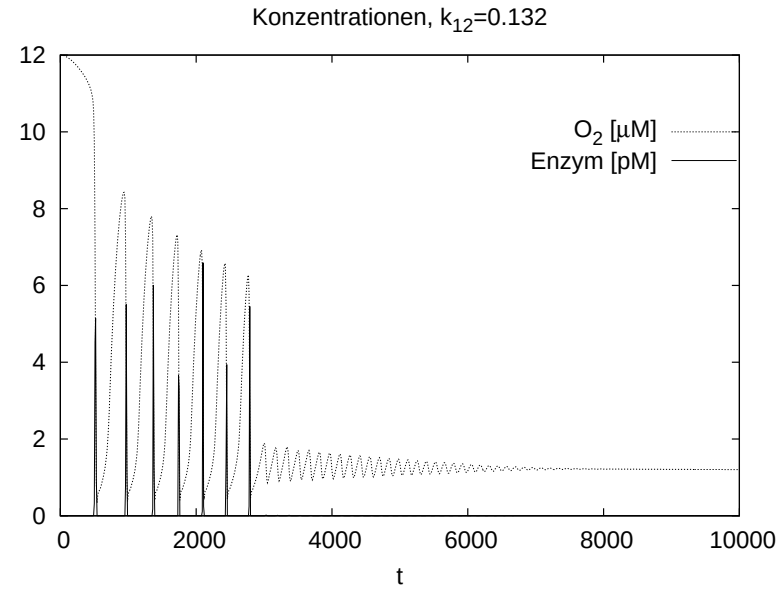
Als Anfangswerte verwenden wir $y_0 = (0, 12.0, 1.5, 0, 0, 0, 0, 0, 0, 0)^T$ und integrieren auf dem Intervall $I = [0, 10^4]$.
Dann lassen sich in Abhängigkeit von k_{12}, der Zuflussrate von $NADH$, sehr anschaulich verschiedenen dynamischen Verhaltensweisen des Systems und seine Empfindlichkeit gegenüber der Zuflussrate beobachten.



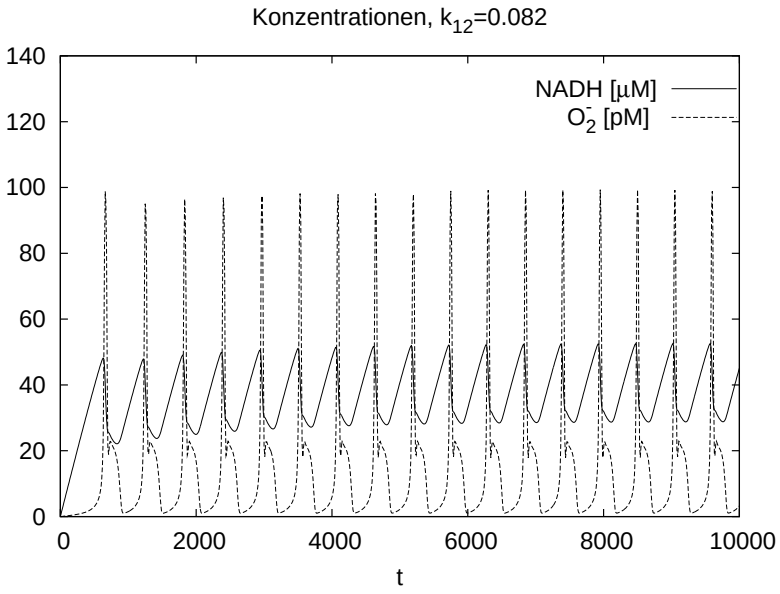


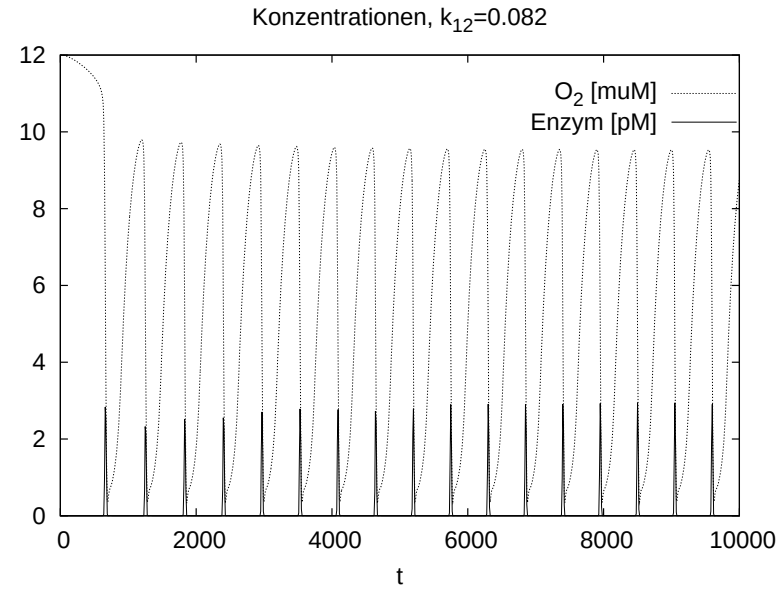
Für eine NADH-Zuflußrate von $k_{12} = 0.129\mu M/s$ durchläuft das System also zunächst eine transiente Phase, die durch Relaxations-Oszillationen mit großen Amplituden charakterisiert ist. Danach geht es über zu kleinen, regulären Oszillationen, wobei der Übergang bei ca. $3200s$ erfolgt. Dabei wird die Aktivierung des Enzyms, die vorher durch das Superoxid O_2^- verursacht wurde, dynamisch ausgeschaltet, wie auf den Abbildungen gut zu erkennen ist.





Für eine NADH-Zuflußrate von $k_{12} = 0.132\mu M/s$ durchläuft das System auch zunächst eine transiente Phase, die durch Relaxations-Oszillationen mit großen Amplituden charakterisiert ist. Danach geht es über zu einer transienten Phase mit kleinen, regulären, aber gedämpften Oszillationen, und schließlich näher es sich einem stationären Zustand an. Dabei kann wieder das Abschalten des Enzyms beobachtet werden.





Für eine NADH-Zuflußrate von $k_{12} = 0.089 \mu M/s$ erhält man schließlich ein System, das durch Relaxations-Oszillationen mit gleichbleibend großen Amplituden charakterisiert ist. Hierbei findet keine Abschaltung des Enzyms statt.

Je nach Zuflussrate und damit verbunden dem dynamischen Verhalten des Systems können wir dann aufgrund der adaptiven Kontrollen des Integrators auch einen unterschiedlichen Aufwand bei der Berechnung des zeitlichen Verlaufs des Systems beobachten.

k_{12}	TOL	#Steps	#Eval. f	#Decomp. Jac	#Deriv. f
0.0129	10^{-6}	4841	14864	2640	474
0.0129	10^{-8}	8680	24960	2640	400
0.0129	10^{-10}	16699	46566	2399	316
0.0132	10^{-6}	3510	10697	1657	278
0.0132	10^{-8}	6387	18267	1715	262
0.0132	10^{-10}	12265	33580	1616	200
0.082	10^{-6}	4208	13743	3202	690
0.082	10^{-8}	7606	23244	3372	596
0.082	10^{-10}	14542	41767	3454	531

Tabelle 6.11: Numerischer Aufwand zur Berechnung der Nominaltrajektorien des Peroxidase-Oxidase Reaktion mittels DAESOL-II für verschiedene Zuflußraten k_{12} von $NADH$ und für verschiedene Toleranzen TOL.

Kapitel 7

Zusammenfassung und Ausblick

Zum Abschluß werden in diesem letzten Kapitel noch einmal die wichtigsten Punkte der Diplomarbeit kurz zusammengefaßt. Außerdem wollen wir mögliche zukünftige Erweiterungen des Integrators DAESOL-II und deren Motivation diskutieren.

7.1 Zusammenfassung

In dieser Diplomarbeit haben wir nach der Vorstellung der benötigten Theorie für DAEs (siehe Kapitel 2) und den allgemeinen linearen Mehrschrittverfahren die BDF-Verfahren eingeführt (siehe Kapitel 3), sowie die notwendigen theoretischen Grundlagen und die wichtigsten Ideen zur effizienten Sensitivitätsgenerierung erläutert (siehe Kapitel 4).

Dann haben wir Ideen und Strategien für ein effizientes BDF-Verfahren präsentiert (siehe Kapitel 5), das Anfangswertprobleme für linear implizite DAEs vom Index 1 löst, wie sie in der Praxis oft bei der Modellierung von chemischen, verfahrenstechnischen und biologischen Prozessen entstehen. Weiterhin erlaubt es die Berechnung von Sensitivitäten der Lösung des Anfangswertproblems in Bezug auf Anfangswerte, Parameter und Kontrollen.

Diese Strategien wurden im Integrator DAESOL-II von Grund auf neu in ANSI-C implementiert. Damit wurde ein Programm geschaffen, welches sehr

effizient gleichermaßen zur Lösung von DAE-IVPs, wie auch zur Sensitivitätsgenerierung eingesetzt werden kann und sich somit hervorragend zur Integration in Optimerungs- bzw. Optimalsteuerung-Codes anbietet. Dabei wurden in DAESOL-II zur effizienten Berechnung von Lösungen und Sensitivitäten die folgenden Punkte realisiert:

- Der Kern des Integrators besteht aus einem BDF-Verfahren variabler Schrittweite und Ordnung. Dabei beruhen die Strategien zur Fehlerschätzung und Schrittweitensteuerung auf dem tatsächlichen variablen Gitter.

- Der modulare Aufbau des Programms gestattet eine einfache Erweiterung, sowie den Austausch bzw. die Anpassung der verwendeten Strategien u.a. zum Start des Verfahrens und der Lösung der nichtlinearen Gleichungssysteme.

- Die Monitor-Strategie für das implementierte Newton-ähnliche Verfahren, welche den Aufwand für die Lösung der nichtlinearen Gleichungssysteme durch eine adaptive Kontrolle der Iterationsmatrizen aufbauend auf dem lokalen Kontraktionssatz reduziert.

- Die Verwendung von sich selbst an die Hardwaregegebenheiten anpassenden Softwarebibliotheken ermöglicht eine hohe Effizienz auf einer Vielzahl verschiedener Systeme ohne großen Anpassungsaufwand.

- Die im Optimierungskontext wichtige Relaxierungsmöglichkeit der algebraischen Gleichungen, um eine Integration von DAEs auch mit inkonsistenten Anfangswerten starten zu können.

- Die effiziente Sensitivitätsgenerierung aufbauend auf dem Prinzip der Internen Numerischen Differentiation. Dabei existiert die Möglichkeit, direkt Richtungsableitungen zu berechnen und so Speicherplatz und Aufwand zu sparen.

Abschließend haben wir die Effizienz des von uns implementierten Integrators DAESOL-II bei der Lösung von Anfangswertproblemen und der Generierung von Sensitivitäten anhand mehrerer komplexer und anspruchsvoller Anwendungsbeispiele demonstriert (siehe Kapitel 6).

7.2 Ausblick

DAESOL-II in seiner in dieser Arbeit vorgestellten ersten Version bildet ein effizientes und starkes Fundament für zukünftige Erweiterungen und Verbesserungen. Einige Möglichkeiten dazu wollen wir im Folgenden diskutieren, wobei wir diese zum Teil schon in der Arbeit skizziert haben.

- Zur Gewinnung von konsistenten Anfangswerten für die Integration bei stark nichtlinearen DAEs, und/oder solchen für die keine guten Schätzungen der Anfangswerte vorliegen, ist die Implementation eines gedämpften Newton-Verfahrens oder eines Homotopieverfahrens sinnvoll.

- Im Hinblick auf einen effizienteren Start bzw. Neustart des BDF-Verfahrens nach Unstetigkeitsstellen oder Änderungen der Modellfunktionen bietet sich die Implementierung eines Einschrittverfahrens höherer Ordnung als Starter des BDF-Verfahrens an, wie es zum Beispiel von Bauer [Bau99] beschrieben wird.

- Eine Implementierung einer Strategie zur Schaltpunktsuche und Updates der Sensitivitätsmatrizen zur Behandlung von DAE-Problemen mit mehreren Modellen, bei denen der Wechsel zwischen den Modellen implizit durch Schaltfunktionen gegeben ist, wie sie Brandt-Pollmann [BP04] beschrieben hat. Die Implementierung dieser Strategie für den Fall von ODE-Problemen hat bereits im Rahmen einer anderen Diplomarbeit auf der Basis von DAESOL-II begonnen.

- Im Optimierungs- bzw. Optimalsteuerungskontext, wo DAEs als Nebenbedingungen auftreten, werden Sensitivitäten nach Anfangswerten, Parametern und Kontrollen benötigt. Hierbei ist auch eine Anwendung des Prinzips der Internen Numerischen Differentiation auf die adjungierte Variationsdifferentialgleichung sehr interessant. Dadurch erhält man einen Rückwärtsmodus der IND, der eine sehr effiziente Möglichkeit der Gewinnung von Sensitivitätsinformationen erlaubt, wenn die Ableitungen nur weniger Komponenten der Lösung nach vielen Anfangswerten, Parametern und Kontrollen benötigt werden.
  Da hierbei, analog zum Rückwärtsmodus der Automatischen Differentiation, die Zwischenwerte der Nominaltrajektorie zur Berechnung der

Ableitungen benötigt werden, ist dies u.U. mit einem sehr hohen Speicherbedarf verbunden. Hier bietet eine sogenannte Checkpointing-Strategie Abhilfe, wie sie in der Arbeit von Walther [Wal00] vorgestellt wird. Hierbei wird an definierten Stellen, z.B. bei Neuzerlegung der Iterationsmatrizen, der Status des Integrators und der Wert der Nominaltrajektorie gespeichert, und bei Bedarf werden dann die benötigten Zwischenwerte erneut berechnet.

- Ebenfalls im Zusammenhang mit der Optimierung und besonders der optimalen Versuchsplanung ist eine Möglichkeit der Generierung von zweiten und höheren Ableitungen der Lösungen von Interesse. Dies könnte mit Hilfe der Automatischen Differentiation der Modellfunktionen, insbesondere der Kombination von Vorwärts- und Rückwärtsmodus der AD, sehr effizient implementiert werden.

- Im Zusammenhang mit der Behandlung von Problemen aus der Biochemie und Biotechnologie, die in der letzten Zeit von immer größer werdendem wissenschaftlichen und wirtschaftlichem Interesse ist, bietet sich die Integration von speziellen Methoden für die Lösung der nichtlinearen Gleichungssysteme an. Denn bei diesen Anwendungen treten oft Modelle auf, die aus der Diskretisierung von partiellen Differentialgleichungen herrühren und die somit eine bestimmte Struktur haben, die es auszunutzen gilt.

- In Modellen aus der Wirtschaftstheorie, aber z.B. auch der Biologie, treten bisweilen sogenannte retardierte Differentialgleichungen auf, bei denen die Modellfunktionen nicht nur von den Zustandsvariablen im aktuellen Zeitpunkt, sondern auch zu - möglicherweise implizit definierten - früheren Zeitpunkten abhängen.
  Die Möglichkeit zur kontinuierlichen Darstellung vergangener Trajektorienwerte durch die Interpolationspolynome des BDF-Verfahrens bietet hier einen Ansatz zur Behandlung dieser Art von Problemen, wie es von Bock und Schlöder in [BS81] für Adamsverfahren dargestellt wurde.

- Zur begrenzten Behandlung von DAE-Problemen mit höherem Index ist die Implementation eines Projektionsverfahrens denkbar, wie es bei-

spielsweise bei Eich [Eic91] verwendet wird.

- Zu guter Letzt bietet sich im Hinblick auf die Lösung von sehr großen Problemen sowie zur Verbesserung der Laufzeiten eine Parallelisierung des Verfahren an. Diese könnte zum einem natürlich in Hinblick auf die Lösung der auftretenden Gleichungssysteme erfolgen, wobei auch über eine Version mit Datenparallelität nachgedacht werden könnte, womit dann extrem große Probleme, die ansonsten speicherplatztechnisch nicht behandelbar wären, gelöst werden könnten.
  Aber auch das Integrationsverfahren an sich läßt sich in nicht direkt voneinander abhängende Segmente aufteilen. So hängt beispielsweise die Berechnung des nächsten Wertes der Nominaltrajektorie weder von der Sensitivitätsgenerierung noch der Generierung der kontinuierlichen Lösungsdarstellung in vorherigen Schritten ab, so dass diese Berechnungen leicht voneinander entkoppelt und parallel betrieben werden können.

Literaturverzeichnis

[ABB+99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[ATL] ATLAS Homepage. http://math-atlas.sourceforge.net/.

[Bau99] I. Bauer. *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik*. PhD thesis, University of Heidelberg, 1999. Download at: http://www.ub.uni-heidelberg.de/archiv/1513.

[BBLS99] H.G. Bock, I. Bauer, D.B. Leineweber, and J.P. Schlöder. Direct multiple shooting methods for control and optimization of DAE in chemical engineering. In F. Keil, W. Mackens, H. Voß, and J. Werther, editors, *Scientific Computing in Chemical Engineering II*, volume 2, pages 2–18, Springer, Berlin, 1999.

[BCC+92] C.H. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR generating derivative codes from fortran programs. *Scientific Programming*, 1:11–29, 1992.

[BCP96] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*. Classics in Applied Mathematics. 14. Philadelphia, PA: SIAM, Society for Industrial and Applied Mathematics. x, 1996.

[BDD+02] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, and et. al. An updated set of basic linear algebra subprograms (BLAS). *ACM Trans. Math. Soft.*, 28:135–151, 2002.

[BES88] H.G. Bock, E. Eich, and J.P. Schlöder. Numerical solution of constrained least squares boundary value problems in differential-algebraic equations. In K. Strehmel, editor, *Numerical Treatment of Differential Equations*. Teubner, Leipzig, 1988.

[BLA02] Basic Linear Algebra Subprograms technical forum standard. *International Journal of High Performance Applications and Supercomputing*, 16, 2002.

[Ble86] G. Bleser. Eine effiziente Ordnungs- und Schrittweitensteuerung unter Verwendung von Fehlerformeln für variable Gitter und ihre Realisierung in Mehrschrittverfahren vom BDF-Typ. Master's thesis, University of Bonn, 1986.

[Boc81] H.G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K.H. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, volume 18 of *Springer Series in Chemical Physics*, pages 102–125. Springer, Heidelberg, 1981.

[Boc87] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. University of Bonn, Bonn, 1987.

[BP04] U. Brandt-Pollmann. *Numerical solution of optimal control problems with implicitly defined discontinuities with applications in engineering*. PhD thesis, IWR, University of Heidelberg, 2004.

[BS81] H.G. Bock and J.P. Schlöder. Numerical solution of retarded differential equations with state-dependent time lags. *ZAMM*, 61:269, 1981.

[BSS95] H.G. Bock, J.P. Schlöder, and V.H. Schulz. *Numerik grosser Differentiell-Algebraischer Gleichungen - Simulation und Opti-*

mierung, in H. Schuler (Hrsg.): Prozeßsimulation, chapter 2, pages 35–80. VCH-Verlag Weinheim, 1995.

[CH52] C.F. Curtiss and J.O. Hirschfelder. Integration of stiff equations. *Proc. Nat. Acad. Sci.*, 38:235–243, 1952.

[CL55] E.A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGrw-Hill, New York, 1955.

[CL84] M. Crouzeix and F.J. Lisbona. The convergence of variable-stepsize, variable-formula, multistep methods. *SIAM Journal on Numerical Analysis*, 21(3):512–534, 1984.

[CLM87] M. Calvo, F.J. Lisbona, and J. Montijano. On the stability of variable nordsieck BDF methods. *SIAM J. Numer. Anal*, 24::844–854, 1987.

[Cry72] C.W. Cryer. On the instability of high order backward-difference multistep methods. *BIT*, 12:17–25, 1972.

[Dah56] G. Dahlquist. Convergence and stability in numerical integration of ordinary differential equations. *Math. Scand.*, 4:33–53, 1956.

[Dah63] G. Dahlquist. A special stability problem for linear multistep methods. *BIT*, 3:27–43, 1963.

[Dav] T.A. Davis. UMFPACK version 4.4 user guide. Available from http://www.cise.ufl.edu/research/sparse/umfpack/.

[Dav04] T. A. Davis. Algorithm 832: UMFPACK - an unsymmetric-pattern multifrontal method with a column pre-ordering strategy. *ACM Trans. Math. Software*, 30:196–199, 2004.

[Die01] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, University of Heidelberg, 2001. http://www.ub.uni-heidelberg.de/archiv/1659/.

[EHL75] W. H. Enright, T. E. Hull, and B. Lindberg. Comparing numerical methods of stiff systems of ODEs. *BIT*, 15:10–48, 1975.

[Eic87] E. Eich. Numerische Behandlung semi-expliziter differentiell-algebraischer Gleichungssysteme vom Index 1 mit BDF-verfahren. Master's thesis, University of Bonn, Bonn, 1987.

[Eic91] E. Eich. *Projizierende Mehrschrittverfahren zur numerischen Lösung von Bewegungsgleichungen technischer Mehrkörpersysteme mit Zwangsbedingungen und Unstetigkeiten.* PhD thesis, University of Augsburg, 1991.

[Eic93] E. Eich. Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints. *SIAM J. Numer. Anal.*, 30:1467–1482, 1993.

[Enk84] K. Enke. Ein effizientes Rückwärtsdifferentiationsverfahren mit variabler Schrittweite und Ordnung zur Lösung steifer Anfangswertaufgaben. Master's thesis, University of Bonn, 1984.

[FKN72] R.J. Field, E. Körös, and R.M. Noyes. Oscillation in chemical systems part 2. thorough analysis of temporal oscillations in the bromate-cerium-malonic acid system. *Journal of the American Chemical Society*, 94:8649–8664, 1972.

[Gea71] C.W. Gear. *Numerical initial value problems in ordinary differential equations.* Prentice Hall, 1971.

[Gea74] C.W. Gear. Asymptotic estimation of errors and derivatives for the numerical solution of ordinary differential equations. *IFIP 74*, pages 447–451, 1974.

[Gea80] C.W. Gear. Runge-kutta starters for multistep methods. *ACM Trans. Math. Softw.*, 6:263–279, 1980.

[Gea88] C.W. Gear. Differential-algebraic equation index transformations. *SIAM J. Sci. Stat. Comp.*, 9:39–47, 1988.

[GJM+99] A. Griewank, D. Juedes, H. Mitev, J. Utke, O. Vogel, and A. Walther. ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. Technical report, Technical University of Dresden, Institute of Scientific Computing and Institute of Geometry, 1999.

[GNU] The GNU General Public License (GPL). Available from http://www.gnu.org/licenses/gpl.html.

[Gri83] R.D. Grigorieff. Stability of multistep-methods on variable grids. *Numerische Mathematik*, 42:359–377, 1983.

[Gri00] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation.* Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000.

[GT74] C.W. Gear and K.W. Tu. The effect of variable mesh size on the stability of multistep methods. *SIAM J. Numer. Anal.*, 11:1025–1043, 1974.

[GV83] C.W. Gear and T. Vu. Smooth numerical solutions of ordinary differential equations. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations.* Birkhäuser, Boston, 1983.

[GW74] C.W. Gear and D.S. Watanabe. Stability and convergence of variable order multistep methods. *SIAM J. Numer. Anal.*, 11:1044–1058, 1974.

[HEFS72] T.E. Hull, W.H. Enright, B.M. Fellen, and A.E. Sedgwick. Comparing numerical methods for ordinary differential equations. *SIAM J. Numer. Anal.*, 9:603–637, 1972.

[HKLO01] M.J.B. Hauser, U. Kummer, A.Z. Larsen, and L.F. Olsen. Oscillatory dynamics protect enzymes and possibly cells against toxic intermediates. *Faraday Discuss*, 120:215–227, 2001.

[HLR89] E. Hairer, C. Lubich, and M. Roche. *The numerical solution of differential-algebraic systems by Runge-Kutta methods.* Lecture Notes in Mathematics 1409, Springer-Verlag., 1989.

[HNW93] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*, volume 8 of *Springer Series in Computational Mathematics.* Springer, Berlin, 2nd edition, 1993.

[HNW96] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations II*, volume 14 of *Springer Series in Computational Mathematics*. Springer, Berlin, 2nd edition, 1996.

[KÖ2] S. Körkel. *Numerische Methoden für Optimale Versuchsplanungsprobleme bei nichtlinearen DAE-Modellen*. PhD thesis, University of Heidelberg, Heidelberg, 2002.

[Lei95] D.B. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Master's thesis, University of Heidelberg, 1995.

[Lei99] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschr.-Ber. VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[MI03] F. Mazzia and F. Iavernaro. Test set for initial value problem solver. Department of mathematics, University of Bari, August 2003. Available at http://www.dm.uniba.it/ testset.

[Nor62] A. Nordsieck. On numerical integration of ordinary differential equations. *Math. Comput.*, 16:22–49, 1962.

[OR66] J.M. Ortega and W.C. Rheinboldt. On discretization and differentiation of operators with application to newton's method. *SIAM J. Numer. Anal.*, 3:143–156, 1966.

[PRM97] L.R. Petzold, Y. Ren, and T. Maly. Regularization of higher-index differential-algebraic equations with rank-deficient constraints. *SIAM Journal on Scientific Computing*, 18:753–774, 1997.

[PSV94] C.C. Pantelides, R.W.H. Sargent, and V.S. Vassiliadis. Optimal control of multistage systems described by high-index differential-algebraic equations. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*. Birkhäuser, Basel, 1994.

[Rhe84] W.C. Rheinboldt. Differential-algebraic systems as differential equations on manifolds. *Math. of Comput.*, 43:473–482, 1984.

[SB92] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer, New York, 1992.

[SW95] K. Strehmel and R. Weiner. *Numerik gewöhnlicher Differentialgleichungen.* Teubner, 1995.

[vH] Dimitri van Heesch. Homepage of the doxygen project. http://www.doxygen.org.

[vS97] R. von Schwerin. *Numerical methods, algorithms, and software for higher index nonlinear differential-algebraic equations in multibody system simulation.* PhD thesis, University of Heidelberg, 1997.

[vSB95] R. von Schwerin and H.G. Bock. A runge-kutta-starter for a multistep-method for differential-algebraic systems with discontinuous effects. *APNUM*, 18:337–350, 1995.

[Wal93] W. Walter. *Gewöhnliche Differentialgleichungen.* Springer, 1993.

[Wal00] A. Walther. *Program Reversal Schedules for Single- and Multiprocessor Machines.* PhD thesis, Technical University of Dreasden, 2000.

[WPD01] R. C. Whaley, A. Petitet, and J.J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001.

[ZLK+05] J. Zobeley, D. Lebiedz, J. Kammerer, A. Ishmurzin, and U. Kummer. A new time-dependent complexity reduction method for biochemical systems. *Lecture notes in computer science: Transactions on Computational Systems Biology I*, 3380:90 pp, 2005.