INAUGURAL - DISSERTATION
zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht - Karls - Universität
Heidelberg

vorgelegt von
Diplom-Physiker Ulrich Brandt-Pollmann
aus Lahn-Gießen

Tag der mündlichen Prüfung: 21.09.2004

Numerical solution of optimal control problems with implicitly defined discontinuities with applications in engineering

Gutachter: Prof. Dr. Dr. h.c. Hans Georg Bock

Gutachter: Prof. Dr. Dr. h.c. Jürgen Warnatz

Abstract

In the thesis on hand we treat optimal control problems for implicitly discontinuous dynamical processes.

We give a general model formulation which includes implicitly given state dependent discontinuities in the right hand sides of the DAE system. The formulation is adapted to real-world applications from chemical and biotechnological engineering. The resulting problems are large scale constrained problems of optimal control with implicitly given discontinuities of a priori unknown chronology and number.

Our solution approach builds on the direct multiple shooting approach which allows the combination of appropriate DAE solvers with modern simultaneous optimization strategies. To solve the underlying optimization problem we apply SQP methods.

We explain our strategy to provide sensitivity information at the presence of implicitly given discontinuities for large scale models. Efficient techniques for the derivative generation of the right hand sides particularly adapted to structural sparsity pattern changes of the adjacent Jacobians are presented.

We formulate an algorithm to treat the optimization problem which depends on the chronology and number of discontinuities occuring in a digraph given by the successive trajectories of the SQP steps.

We explain our modeling of a complex rack-in process of a distillation column and present the models of two biotechnological processes. Each of the models is equipped with characteristical implicit state dependent discontinuities of a priori unknown chronology.

In numerical experiments we show the efficient applicability of our algorithms to the presented chemical process and to the two biotechnological applications. We apply our approach to optimal feedback control of a biotechnological application with implicit discontinuities.

Zusammenfassung

In dieser Arbeit werden implizit zustandsabhängig unstetige dynamische Prozesse behandelt.

Es wird eine allgemeine mathematische Modellformulierung angegeben, die implizit gegebene zustandsabhängige Unstetigkeiten in den rechten Seiten des DAE-Systems einschließt. Anschließend wird die Formulierung an konkrete chemische und biotechnologische Prozesse angepaßt, woraus große nicht-lineare optimale Kontrollprobleme mit implizit gegebenen Unstetigkeiten von a priori unbekannter Anzahl und Chronologie resultieren.

Unser Lösungsansatz basiert auf dem direkten Mehrzielverfahren, welches die Kombination eine problemangepaßten DAE Lösers mit modernen simultanen Verfahren der Optimierung ermöglicht. Zur Lösung des Optimierungsproblems verwenden wir SQP Verfahren.

Wir erklären unsere Strategie zur Bereitstellung von Sensitivitätsinformation bei implizit gegebenen Unstetigkeiten für *large scale* Modelle. Effiziente Techniken für die Ableitungsgenerierung der rechten Seiten, die speziell an Dünnbesetztheitsmusterwechsel der entsprechenden Jacobi-Matrizen angepaßt sind, werden präsentiert.

Wir formulieren einen Algorithmus zur Behandlung des Optimierungsproblems, das von der Chronologie sowie der Anzahl der Unstetigkeiten abhängt, die im durch die Trajektorien in den einzelnen SQP Schritten gegebenen gerichteten Graphen auftreten.

Unsere Modellierung eines komplexen Anfahrprozesses einer Destillationskolonne wir eingehend erläutert. Außerdem gehen wir auf die Modelle zweier biotechnologischer Prozesse detaillierter ein. Alle diese Modelle weisen charakteristische implizit zustandsabhängige Unstetigkeiten a priori unbekannter Chronologie auf.

In numerischen Experimenten zeigen wir die effiziente Anwendbarkeit unserer Algorithmen auf die präsentierten chemischen und biotechnologischen Anwendungen.

Wir wenden unseren Ansatz zur Feedback Steuerung biotechnologischen Anwendung mit impliziten Unstetigkeiten an.

Acknowledgements

Above all, I owe thanks to my advisors Prof. Dr. Dr. h.c. Hans Georg Bock and Dr. Johannes P. Schlöder for their excellent support during the last years. It is a pleasure to acknowledge the immense benefit of their deep knowledge and their impressive overview for my work. I thank them just as much for the cordial and productive atmosphere they create in our research group.
Further, I heartily thank my office mate Dr. Katja Mombaur for discussions, numerous joint scientific activities and her positive attitude toward whatever comes.
Let me extend my thanks to my colleague Andreas Alexis Sigurt Schäfer for his patience concerning uncounted MUSCOD-II questions and for his highly structured style of implementation. I am obliged to Dr. Moritz Diehl, especially for helping me to get into contact with NMPC, and to Christian Kraus for valuable mathematical and nonmathematical conversations.
Thomas Klöpfer always had an open ear for all kinds of questions and trouble concerning our computers, even on weekends and holidays.
Throughout the time of my thesis research, I have been accompanied and supported in many ways by members of our research group and by other people at the IWR. I would like to address thanks to all of them, especially to Dr. Stefan Körkel, Dr. Ekaterina Kostina, and Sebastian Sager.
I am indebted to Dr. Dirk Lebiedz for several interesting joint projects and publications, and for private volleyball lessons.
Let me also thankfully mention my industrial project partner Dr. Jens Köhler (BASF) who contributed to this work by posing the right questions, and who was always available for a scientific discussion.
I am deeply grateful to Margret Rothfuss for her patience, her clarity, and her ability to get things off the ground.
I leave a special note to Ted for offering encouragement and consolation whenever needed.
Finally, I want to thank my fiancée Dr. Julia Hartmann for her love and for her unconfined support.

The financial support by the *Bundesministerium für Bildung und Forschung* within the context of the project *Nicht-Standard Probleme der Optimalen Steuerung* is gratefully acknowledged.

Contents

Introduction

In the last years optimal control of complex chemical and biotechnological processes has attracted increasing interest. Due to the competition on globalized markets becoming more and more severe, the availability of highly efficient methods for process control turns out to be crucial. The need for model-based optimization to reduce energy and resource expenses and for ameliorated product quality increases.

Distillation processes play an eminent role in chemical engineering. To accurately describe complex distillation processes rigorous highly nonlinear models are often needed. In order to precisely represent the complex instationary states of distillation processes e.g. phase transitions and the dynamics of the vapor-gas mixture have to be taken into account explicitely. This precise modeling especially becomes crucial for the representation of start-up processes. In contrast to the optimization of processes which operate not too far from steady-state, qualitative changes in the system dynamics, i.e., model changes, which are implicitly given, due to e.g. phase transitions may arise in an a priori unknown chronology and number.

Some years ago the modeling of biotechnological processes, e.g., the modeling of the generation of antibiotics was brought into focus. Those models (e.g. [Küh02], [Kin94]) show a characteristical behavior in the sense that due to locally vanishing substrate concentrations implicitly given qualitative model changes arise. Numerically these discontinuities show a different behavior than the discontinuities normally arising in rack-in processes of distillation columns.

The focus of this thesis is to adequately model the complex system dynamics and develop algorithms which can cope with the nonlinear implicit state dependent discontinuous dynamical models where neither the number nor the chronology of arising switching points can be a priori determined. Until now it was possible for comparatatively small numbers of switching point to formulate transition phases and thus avoiding the explicit treatment of discontinuities (see e.g. [Mom02]). This strategy requires the a priori knowledge of number and chronology of the switching points. Furthermore, since the additional phases significantly increase the size of the quadratic program (QP) this technique can only be applied for comparatetively small numbers of switching points.

It is a highly challenging task to provide optimal solutions for the above mentioned problems.

For the development of adequate algorithms and their implementation in efficient tools we were able to build on the deep knowledge on optimal control available in the research group of Bock and Schlöder at the Interdisciplinary Center for Scientific Computing, University of Heidelberg.

The development of an integrator which can treat highly stiff implicitly discontinuous dynamical processes was performed on the basis of the existing BDF code DAESOL. In combination with our approach to provide sensitivity information which is particularly adapted to large scale rigorous models of distillation columns with implicitly given discontinuities, the integrator represents the state-of-the-art for this class of tasks.

Recently eminent progress has been achieved in the context of optimal control of complex processes. Based on ideas of Bock and Plitt ([Pli81], [BP84], [Boc87]) who invented the powerful direct multiple shooting approach for optimal control, Leineweber and coworkers ([Lei99]) developed the modern optimal control package MUSCOD-II forming the basis for the optimization part of the work on hand.
Our strategy represents the first optimal control strategy which can deal with large scale models of differential algebraic equations with implicitly defined state dependent discontinuities which arise in a priori unknown chronology and number.
The goal of this interdisciplinary thesis is to develop and implement

- accurate models for the complex rack-in distillation process,
- a highly efficient DAE integrator for severely stiff implicitly discontinuous processes,
- an optimization algorithm which can cope with qualitative changes in the differential algebraic equations.

We thus provide an algorithm which can treat implicitly discontinuous optimal control problems where neither number nor chronology of the switching events are a priori known.

This for the first time allows in the context of chemical industry the optimization of general rack-in and rack-out processes where e.g. phase transitions may appear in varying chronology. For biotechnological applications we pave the way for the optimization of processes where substances may temporarily vanish in arbitrary chronology.

Outline of the thesis

Since we don not expect every reader to be interested in every single chapter of the thesis we tried to keep the chapters as independent as possible. Apart from Chapter 5 which requires the previous chapters the chapters should be independently readable.

In Chapter 1 we formulate the class of optimal control problems we treat with our algorithms. Afterwards, we review in short a broader class of algorithms which have been proposed as a solution ansatz and briefly discuss their advantages and disadvantages.
We subsequently present the discretization which transforms the originally infinite dimensional optimal control problem to a finite dimensional one. The underlying differential algebraic equation (DAE) is given in a relaxed formulation, which allows an inconsistent solving of the associated initial value problem (IVP).
The parameterized form of the optimal control problem gives rise to a particularly structured nonlinear program (NLP). The chapter ends with an illustrative example.

Chapter 2 is divided into two sections.
In the first of these sections we focus on the local optimality conditions for the NLP problem arising from the parameterization given in the first chapter. In the second section the SQP method is introduced. Since for sequential solving of a quadratic program (QP) second derivative information is needed, we discuss the generation of Hessian approximation based on curvature information.

The third chapter deals with the solution algorithm for the underlying differential algebraic equation (DAE). In the first section, we give a brief survey of existing initial value problem (IVP) solvers. The succeeding section is devoted to the BDF (backward differentiation formula) method which is appropriate for our requirements. We briefly explain our trajectory memory technique for storing trajectory information, paving the way for an efficient treatment of implicitly defined discontinuities both within the integrator and in the optimization techniques, presented in the later chapters. In the third section of Chapter 3, we explain the efficient treatment of implicitly defined discontinuities - either discontinuities in the states themselves or discontinuities in the state-derivatives - of the DAE. Moreover, the solution strategy for the variational DAE in order to provide derivative information of the solution with respect to initial values and controls at the presence of discontinuities is shown. Thereafter, we present two applications which show the characteristic behavior of instationary implicit discontinuous chemical processes and biotechnological batch processes, respectively.

The topic of Chapter 4 is the derivative generation. In the first section we list the derivatives needed in the SQP algorithm, the generation of sensitivity information is dealt with in the second. The third section briefly reviews the techniques of automatic differentiation which are necessary for the solution of our problems. Finally we show how the sparsity structures in derivative matrices can be exploited efficiently. We also analyze the special properties of the sparsity patterns of the Jacobians of the right hand sides of the DAE at the presence of implicit discontinuities in the state variables.

In Chapter 5 we give a solution strategy for optimal control problems with implicit discontinuities. We first formulate the modified optimal control problem, explicitely allowing implicitly given discontinuities in the state vectors which may lead to qualitative changes of the cost function. The second section treats the monitoring of the areas accessed by the possibly infeasible trajectory. In the subsequent section the updating of the consistency conditions of the multiple shooting discretization is presented. At last, the monitor for the - due to differing area digraphs - qualitatively different trajectories is explained.

The final chapter concentrates on applications. We first discuss our modeling of the distillation process. The model precisely describes the dynamics of the liquid-vapor mixture and explicitely allows vanishing liquid holdup. Subsequently, we present optimization results of the rack-in process of the distillation column. Afterwards the optimization of two biotechnological processes is presented. Finally we explain the feedback optimization of a batch fermentation process.

In the first Appendix (Appendix A) we explain the modeling of the distillation process.

In Appendix B we demonstrate the usage of algorithmic differentiation for differential algebraic equations with implicitly defined discontinuities in our optimal control package.

Chapter 1

The optimal control problem - Multiple shooting discretization

In this chapter we give a general formulation of the class of optimal control problems we treat with our algorithms with focus on the applications to be presented later. We first state the problem and give a brief overview of existing approaches to motivate our choice of the direct multiple shooting method. Afterwards we present the arising nonlinear program (NLP). In this chapter we do not focus on implicit discontinuities in the states or their derivatives of the states when paving the way for the solution strategy but defer this to later chapters.

The following optimal control problem formulation covers a wide range of practical problems, e.g. optimization of dynamical processes in chemical [LBS97], [BP02] or biotechnological engineering [KWB$^+$95], [BP03] with implicitly given discontinuities. Beyond these examples optimal spatiotemporal control problems in bio-mathematics [LBP03], [LBP04b], chemistry [LBP04c], biochemistry [LBP04a] or financial sciences [WBPMS03] can be treated in the context of this formulation.

1.1 The optimal control problem - continuous form

The systems treated in the following can be described by a differential algebraic equation with implicit discontinuities in the state variables

$$\left.\begin{array}{rcl} A(t,z(t),u(t),p)\cdot\dot{x}(t) &=& f(t,z(t),u(t),p,\mathrm{sgn}(\sigma(t))) \\ 0 &=& g(t,z(t),u(t),p,\mathrm{sgn}(\sigma(t))) \\ z(t_s^+) &=& z(t_s^-)+\hat{\Delta}^z(t_s,z(t_s),p) \end{array}\right\} t,t_s\in[t_i,t_f]. \qquad (1.1)$$

The time interval $[t_{initial}, t_{final}]$ (short $[t_i, t_f]$) is of length T. We require the matrix A to be of full rank. The vector $x(t) \in \mathbb{R}^{n_x}$ denotes the *differential*, $y(t) \in \mathbb{R}^{n_y}$ the *algebraic* variables. They are combined to the state variable vector $z(t) \in \mathbb{R}^{n_z}$, $n_z = n_x + n_y$. $u(t) \in \mathbb{R}^{n_u}$ describes the vector of *control* functions, $p \in \mathbb{R}^{n_p}$ the constant *parameters.* $\sigma(t) \in \mathbb{R}^{n_\sigma}$ is the vector of switching functions. The right hand sides f and g do not directly depend on the values of the switching functions $\sigma(t)$ but just on the individual signs of the switching vector. We define

$$\operatorname{sgn}(v) : \mathbb{R}^{n_v} \mapsto \{-1, 1\}^{n_v}$$

which maps a vector $v \in \mathbb{R}^{n_v}$ to its sign vector of dimension n_v. For the sign function we define

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{for} \quad x < 0 \\ +1 & \text{for} \quad x \geq 0 \end{cases}$$

for a scalar value. The function $\Delta^z(t_s, z(t_s), p)$ gives the jump height of the state vector z at the switching point t_s. Multiple switching points t_s are allowed in the interval $[t_i, t_f]$. The function $\hat{\Delta}^z(t, z(t), p)$ describes the sufficiently smooth jump function which describes the jump in the state variables at the switching point t_s, if existing.

$$z(t_s^-) = \lim_{\epsilon \to 0} z(t_s - \epsilon)$$

is the left hand limit of the state vector $z(t)$. For a more detailed discussion of implicitly given discontinuities refer to Chapter 3.

Since A is invertible, the total time-differentiation of the *algebraic equation* of (1.1) leads to

$$\frac{\partial g}{\partial y}\dot{y} = -\frac{\partial g}{\partial t} - \frac{\partial g}{\partial x}A^{-1}f.$$

The Jacobian $\frac{\partial g}{\partial y}$ is required to be of full rank. We then speak of a DAE system of differential index I (see e.g. [Gea88]) of semi-explicit type. The name *index I* comes from the fact that exactly one differentiation is necessary to transfer the DAE into an ordinary differential equation. DAEs of higher index are treated e.g. in [Sch99b]. The *quasi-linear semi-explicit* formulation (1.1) of the DAE is not the most general form. Certain applications require a fully implicit formulation (see e.g. [Han94] (linear case) or [Min04])

$$0 = F(z(t), \dot{z}(t), u(t), p) \tag{1.2}$$

which we will not discuss in this thesis.

The objective function is in general of Bolza type and consists of the Mayer and the Lagrange term

$$\underbrace{\Theta(t_f, z(t_f), p)}_{\text{Mayer term}} + \underbrace{\int_{t_i}^{t_f} \Phi(t, z(t), u(t), p)\, dt}_{\text{Lagrange term}}\,. \tag{1.3}$$

A different class of objective functions is given by Least Square terms motivating other algorithms in the context of minimization of the objective function, see e.g. Körkel [Kör02].

We claim certain equality constraints

$$r(z(t_f), p) = 0$$

at the end point t_f to be fulfilled. They may arise from specific configurations at initial values or may fix final states the process is driven into, e.g. a certain composition of substances in the outflow stream.

The terminal inequality constraints are of the form

$$r(z(t_f), p) \geq 0.$$

The vector $r \in \mathbb{R}^{n_{r_e}+n_{r_i}}$ combines the terminal equality and inequality conditions. The first n_{r_e} components are treated as equality the last n_{r_i} as inequality conditions. Additionally, path constraints

$$h(t, z(t), u(t), p) \geq 0, \quad t \in [t_i, t_f].$$

can arise, e.g. due to safety restrictions.
Since Mayer and Lagrange terms can easily be transformed into one another we restrict ourselves to the treatment of Mayer terms without loss of generality.

We can formulate the optimal control problem in the following form

$$\min_{t_f, z, u, p} \Xi(t_f, z(t_f), p) \tag{1.4a}$$

subject to

$$\begin{aligned}
A(t, z(t), u(t), p) \cdot \dot{x}(t) &= f(t, z(t), u(t), p, \operatorname{sgn}(\sigma(t))), & t \in [t_i, t_f] && (1.4b)\\
0 &= g(t, z(t), u(t), p, \operatorname{sgn}(\sigma(t))), & t \in [t_i, t_f] && (1.4c)\\
z(t_s^+) &= z(t_s^-) + \hat{\Delta}^z(t_s, z(t_s), p) & && (1.4d)\\
0 &\leq h(t, z(t), u(t), p), & t \in [t_i, t_f] && (1.4e)\\
0 &= r(z(t_f), p) & && (1.4f)\\
0 &\leq r(z(t_f), p) & . && (1.4g)
\end{aligned}$$

1.2 The optimal control problem - general treatment

In this section we will briefly introduce indirect methods for optimal conrol problems. These methods originate in the calculus of variations. For a detailed descussion refer e.g. to [Boc78].

Indirect methods prove to be appropriate for some applications where high accuracy is needed.
Due to the often high sensitivity of the single shooting method to variations in the initial values of the *Lagrange*-multipliers, it is in many cases necessary to apply multiple shooting methods ([Osb69], [Bul71]). Application of indirect methods to optimal control problems of larger dimension is hardly possible (see e.g. Feeherey [Fee99]).

In contrast to indirect ones, direct optimal control methods do not explicitely operate on the optimality conditions.

Direct methods evaluate the objective function directly on the basis of some initial values for the discretized controls which are subsequently improved in an iterative manner to fulfill equality and inequality constraints and minimize the objective function. The original continuous optimization problem is transformed to a parameter optimization problem.
Differerent approaches to treat the subjacent system of differential equations have been proposed. One is the complete discretization where the complete time horizon

is transfered into a system of algebraic equations e.g. via collocation. Optimization parameters are the independent control variables and parameters and the dependent variables, say the state variables at the discrete time points (see e.g. Schulz [SBS98], Steinbach [Ste95]). The possible drawback of extraordinarily large parameter optimization problems becoming to large for contemporary optimization methods can be circumvented by keeping the dependent state variables out of the parameter optimization process. This can be done by solving the time discretized system of differential equations using a nonlinear equation solver. This method was first proposed by Nilchan [Nil97] and Li et al. [LW97].

Instead of explicitely parameterizing the time domain our approach relies on initial value problem solvers. As a consequence, the underlying NLP problem only contains the independent optimization variables and the boundary conditions. The required sensitivities can be calculated when integrating the model equations by solving the variational differential equation (4.8).

1.3 The optimal control problem - parameterized form

All direct optimization approaches transform the infinite dimensional optimization problem to a finite dimensional one. The direct multiple shooting method (see Plitt [Pli81], Bock and Plitt [BP84], [BES88], [LBBS03]) achieves this by a state and control discretization. The for large scale models expensive integration and sensitivity generation processes on different multiple shooting intervals are totally decoupled and can be performed in parallel ([LSBS03], [GB94], [BPDLP04]).

Despite the fact that we can in principle treat multi stage problems (e.g. [Lei96]), we restrict our attention to single stage formulations in the following. Nevertheless the application of the complex distillation process is composed of several model stages (see chapter 6).

1.3.1 Discretization of controls

We transform arbitrary time lengths T to the unit-interval $\tau \in [0,1]$ by use of the linear time transformation

$$\tau : [t_i, t_f] \mapsto [0,1]$$

with the process time $t = \tau T + t_i$. With this transformation it suffices to consider optimization problems on the fixed time interval $[0, 1]$. Free end times are handled by treating the end time as a free parameter.

A priori we discretize the time domain by introducing a multiple shooting grid

$$0 = \tau_0 < \tau_1 < \ldots < \tau_{n_{\mathrm{ms}}} = 1. \tag{1.5}$$

On every multiple shooting interval we define a piecewise approximation to the controls $u(\tau)$ by

$$u(\tau) = \phi_j(\tau, q_j), \quad \tau \in [\tau_j, \tau_{j+1}]$$

with locally elementary functions $\phi_j(\tau, q_j)$ with local control parameters q_j. In our case they are chosen to be piecewise constant leading to

$$\phi_j(\tau, q_j) = q_j.$$

In principle other possibly higher order representations can be used (see [Pli81]). Continuity conditions at the multiple shooting points can be realized via equality conditions.

1.3.2 Multiple shooting discretization of the state variables

We introduce $2(n_{\mathrm{ms}} + 1)$ parameters $s_j^x \in \mathbb{R}^{n_x}$ and $s_j^y \in \mathbb{R}^{n_y}$ as differential and algebraic states at the multiple shooting nodes with

$$x_j(\tau_j) = s_j^x, \quad y_j(\tau_j) = s_j^y. \tag{1.6}$$

We solve the initial value problems on the multiple shooting intervals

$$\begin{aligned} A(\cdot)\dot{x}_j(\tau) &= f(z_j(\tau), \phi_j(\tau, q_j), p, \mathrm{sgn}(\sigma)_j)T \\ 0 &= g(z_j(\tau), \phi_j(\tau, q_j), p, \mathrm{sgn}(\sigma)_j) \\ &\quad -\alpha_j(\tau) g_j(s_j^z, \phi_j(\tau_j, q_j), p, \mathrm{sgn}(\sigma)_j), \\ z_j(\tau_s^+) &= z_j(\tau_s^-) + \hat{\Delta}_j^z(\tau_s, z_j(\tau_s), p) \end{aligned} \tag{1.7}$$

on the multiple shooting intervals.
Again, we combine the differential and algebraic variables $x \in \mathbb{R}^{n_x}$ and $y \in \mathbb{R}^{n_y}$ to the state variable vector $z \in \mathbb{R}^{n_z}$ with $n_x + n_y = n_z$. Similarly, the optimization parameters $s_j^x \in \mathbb{R}^{n_x}$ and $s_j^y \in \mathbb{R}^{n_y}$ are combined to $s_j^z \in \mathbb{R}^{n_z}$. The scalar damping

factor $\alpha_j(\tau)$ has to be chosen to satisfy $\alpha_j(\tau_j) = 1$ and to be non-increasing and non-negative on the corresponding interval. We apply

$$\alpha_j(\tau) = \exp\left(-\beta \frac{\tau - \tau_j}{\tau_{j+1} - \tau_j}\right) \tag{1.8}$$

with $\beta \geq 0$*. The damping factor $\alpha_j(\tau)$ causes the solution to successively approach the nominal solution manifold. The relaxed formulation transforms the possibly inconsistent initial value DAE problem to a consistent one, the initial values $(s_j^z \in \mathbb{R}^{n_z}, \phi_j(\tau), q_j)$ by definition make the *relaxed DAE* consistent. This multiple shooting variant was originally proposed by Bock et al. [BES88]. A generalized variant goes back to Schulz et al. [SBS98] and Leineweber [LBBS03].

The supplementary matching conditions

$$x_j(\tau_{j+1}; s_j^z, q_j) = s_{j+1}^x, \quad i = 0, 1, \ldots, n_{\text{ms}} - 1 \tag{1.9}$$

ensure that the state vector at the end of the interval j coincides with the initial state vector of the following multiple shooting interval $j+1$. Additionally we impose the consistency conditions

$$g(s_j^z, \phi_j(\tau), q_j, p, \text{sgn}(\sigma)_j) = 0, \quad i = 0, 1, \ldots, n_{\text{ms}} \tag{1.10}$$

to demand for the vanishing of the inconsistencies in the solution. The conditions (1.10) and (1.9) eliminate the additional degrees of freedom we introduced by the supplementary optimization parameters $s_j^z \in \mathbb{R}^{n_z}$. For fixed initial values we obtain the additional constraint $s_0 = x_0$. At intermediate iterates, the constraints are generally violated, which makes the path to the solution infeasible. Simulation and optimization proceed *simultaneously*.

The feature of the direct multiple shooting method to allow inconsistent start values at the multiple shooting points circumvents the potentially very expensive consistent initialization at intermediate points. For stability and efficiency reasons multiple shooting for the solution of optimization boundary value problems has proved to be superior to single shooting methods (see [Boc83], [Boc87]).

*In the thesis on hand we apply $\beta \in [2, 10]$ which has proven to be appropriate.

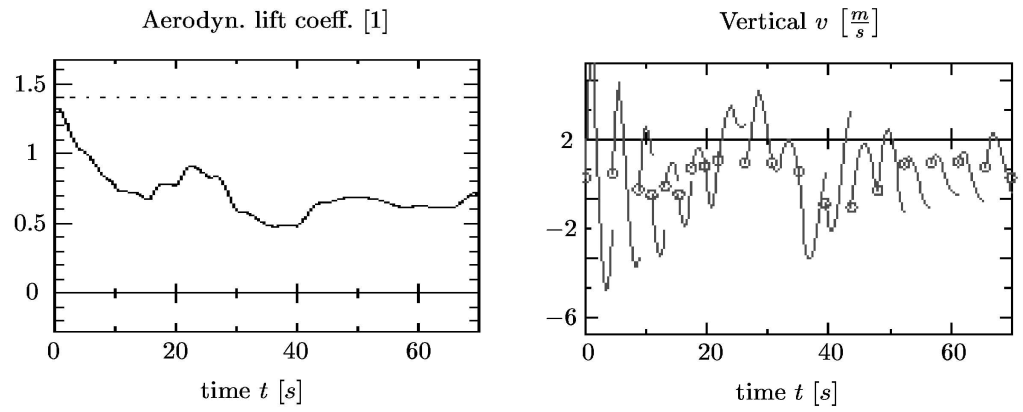

Figure 1.1: First SQP step in solving the classical hang glider range maximization problem ([BNPS91]). In the left picture the piecewise cubic control and in the right picture the still inconsistent trajectory for the vertical velocity is shown.

1.3.3 Path and control constraint discretization

To transform the infinite dimensional path constraints we discretize them on the a priori chosen multiple shooting grid (1.5), which leads to

$$h(s_j^z, \phi_j(\tau), q_j, p) \geq 0, \quad i = 0, \ldots, n_{\mathrm{ms}}. \tag{1.11}$$

1.3.4 Parallelization

The multiple shooting time discretization allows for parallelization. The integration processes of state and sensitivity generation are decoupled and can be performed simultaneously on a parallel computer ([LSBS03], [BPDLP04]) which significantly reduces the running time. To equally charge the processors we, approximately charge every processor with the same number of multiple shooting intervals if no initial guess of the computational costs for the sensitivity generation on the multiple shooting intervals is available, . In the second SQP step we redistribute exploiting the individual run-times of each processor.

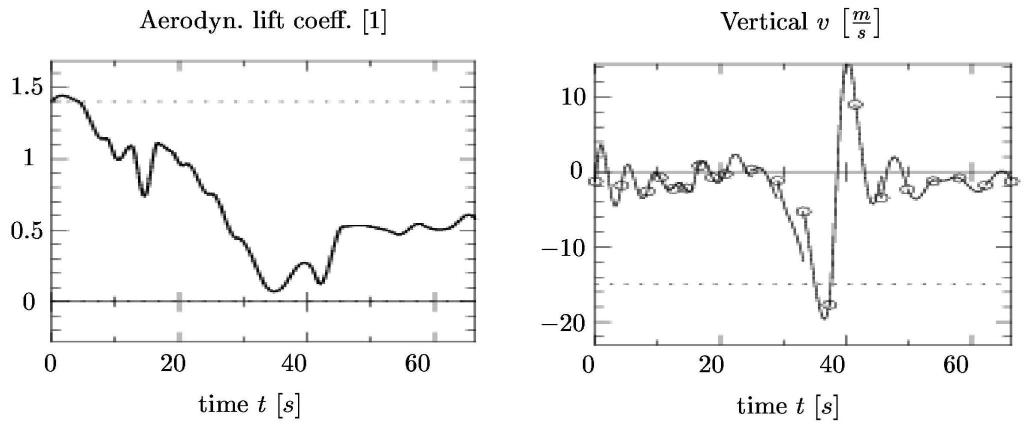

Figure 1.2: Third SQP step: With still violated consistency conditions at the multiple shooting nodes.

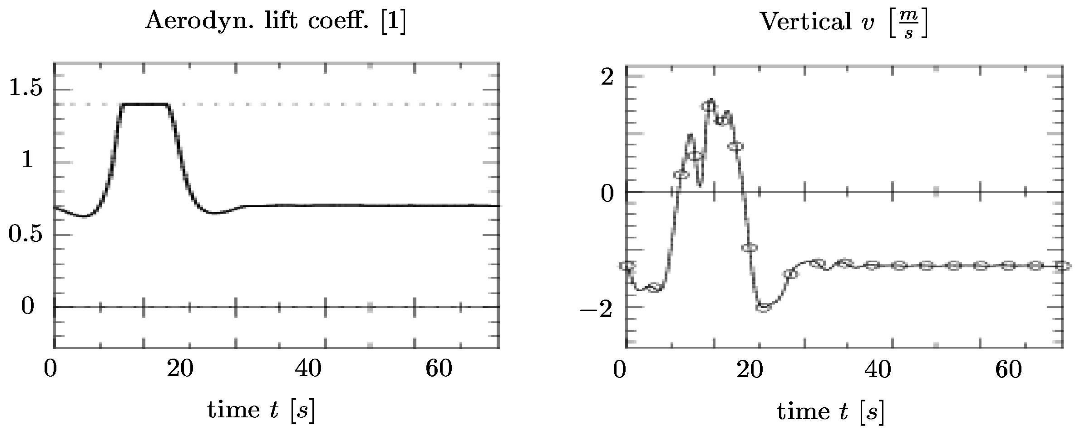

Figure 1.3: Last SQP step, consistency is obtained.

1.4 The NLP problem

The direct multiple shooting discretization of the optimal control problem presented in the last section results in the finite dimensional nonlinear programming problem (NLP)

$$\min_{s_j^z, q_j, p} \sum_{j=0}^{n_{\mathrm{ms}}-1} \Xi_j(s_j^z, \phi_j(\tau, q_j), p) \tag{1.12a}$$

subject to

$$x_j(\tau_{j+1}; s_j^z, q_j, p, \mathrm{sgn}(\sigma)_j) = s_{j+1}^x, \quad i = 0, \ldots, n_{\mathrm{ms}} - 1 \tag{1.12b}$$

$$g(s_j^x, \phi(\tau_j), q_j, p, \mathrm{sgn}(\sigma)_j) = 0, \quad i = 0, \ldots, n_{\mathrm{ms}} \tag{1.12c}$$

$$h(s_j^z, \phi_j(\tau), q_j, p) \geq 0, \quad i = 0, \ldots, n_{\mathrm{ms}} \tag{1.12d}$$

$$r(s_{n_{\mathrm{ms}}}^z, p) = 0, \tag{1.12e}$$

$$r(s_{n_{\mathrm{ms}}}^z, p) \geq 0, \tag{1.12f}$$

where (1.12b) and (1.12c) describe the continuity and consistency conditions, (1.12d) gives the path constraints and the vector $r(s_{n_{\mathrm{ms}}}^z, p)$ ((1.12e) and (1.12f)) combines the equality and inequality terminal boundary conditions. For various reasons coupled multi-point constraints may be necessary. For performance reasons MUSCOD-II only allows of linearly coupled constraints. The formulation (see section 1.3.2) can also be used to treat time delays in the controls by reformulating the problem adequately ([WBPMS03]). In our implementation we explicitely distinguish between coupled and uncoupled path constraints.
For all NLP variables constraints may be specified individually. This turns out to be crucial because it prevents the model from being evaluated in areas where it lacks validity.

Chapter 2

Characterization of optimality and SQP method

In the following chapter we review some characterization theorems, which allow to rate points with respect to optimality and feasibility. In the second section we explain the SQP (sequential quadratic programming) method. We also expound the usage of Quasi-Newton methods as we apply them in our applications.

2.1 Local optimality conditions

This section gives optimality conditions for locally smooth NLP problems. In the first part we review those conditions and discuss their limits. The second part explicitly discusses structural changes in the hypersurface given by the objective due to implicitly defined discontinuities in the state variables of the dynamic process.

2.1.1 Optimality conditions for the classical NLP problem

We set

$$w = (q_0, \dots, q_{n_q-1}, s_0, \dots, s_{n_s-1}, p_0, \dots, p_{n_p-1}) \in \mathbb{R}^{n_w} \tag{2.1}$$

with n_q to be the number of the control parameters, n_p the number of the parameters and $n_s = n_z(n_{\text{ms}}+1)$ the number of variables representing initial values, differential and algebraic variables and the multiple shooting points and

$$F(w) := \min_{s_i^z, q_i, p} \sum_{i=0}^{n_{\text{ms}}-1} \Xi_i(s_i^z, \phi_i(\tau, q_i), p) \tag{2.2}$$

for the objective, and thereby convert the NLP problem (1.12) to the classical NLP problem to be discussed in the following.

Let $F : \mathbb{R}^{n_w} \mapsto \mathbb{R}$ be a $\mathcal{C}^2$ function. The classical NLP is given by

$$\min_{w \in \mathbb{R}^{n_w}} F(w) \quad \text{s.t.} \quad \begin{cases} G(w) &= 0 \\ H(w) &\geq 0, \end{cases} \tag{2.3}$$

where the equality constraint functions $G : \mathbb{R}^{n_w} \mapsto \mathbb{R}^{n_G}$ and the inequality constraint functions $H : \mathbb{R}^{n_w} \mapsto \mathbb{R}^{n_H}$ are also assumed twice continuously differentiable. We subsume the gradients of the components of the vector valued functions G and H in the generalized gradients

$$\nabla_w G := \left(\frac{\partial G}{\partial w}\right)^T, \qquad \nabla_w H := \left(\frac{\partial H}{\partial w}\right)^T.$$

The transposes of these generalized gradients are obviously the Jacobians of G and H.

Definition 2.4 (Feasibility)
The **feasible area** *of the NLP (2.3) is the set of points w*

$$\Omega = \{ w \in \mathbb{R}^{n_w} | \, G(w) = 0 \quad \wedge \quad H(w) \geq 0 \} \,.$$

Points $w \in \mathbb{R}^{n_w} \setminus \Omega$ are called **infeasible***.*

The constraint $H_i(w) \geq 0$ is said to be *active*, if $H_i(w) = 0$ holds. Let $\mathcal{G}$ be the index set of the equality constraints and $\mathcal{H}$ the index set of the inequality constraints.

Definition 2.5 (Active set)
The index set

$$\mathcal{A}(w) = \mathcal{G} \cup \{ i \in \mathcal{H} \mid H_i(w) = 0 \}$$

of active inequalities for a point $w \in \Omega$ is called the active set.

Definition 2.6 (Local minimizer)
A point $w^* \in \mathbb{R}^{n_w}$ is called a local minimizer of the NLP problem if w^* is feasible w.r.t. all constraints and there exists a neighborhood $\mathcal{U}_\epsilon(w^*)$ with

$$F(w^*) \leq F(w) \ \text{ for all } \ w \in \mathcal{U}_\epsilon(w^*).$$

Introducing the *Lagrange function*

$$\mathcal{L}(w, \lambda, \mu) = F(w) - \lambda^T G(w) - \mu^T H(w) \tag{2.7}$$

with the Lagrange multipliers $\lambda \in \mathbb{R}^{n_G}$ and $\mu \in \mathbb{R}^{n_H}$ allows for investigations concerning local optimality (2.6) in the presence of equality and inequality constraints.

Definition 2.8 (Linear independence constraint qualification (LICQ))
If the set of active constraint gradients

$$\{\nabla_w G, \nabla_w H_i, i \in \mathcal{A}(w^*)\}$$

is linearly independent at a point w^* and a given active set $\mathcal{A}(w)$ we say that the linear independence constraint qualification (LICQ) is fulfilled at w^*.

Based on the definition (2.8) we can formulate the Karush-Kuhn-Tucker conditions

Theorem 1 (Karush-Kuhn-Tucker conditions)
Let w^* be a local minimizer (2.6) of (2.3) for which LICQ (2.8) holds. Then there exist unique Lagrange multipliers $\lambda^* \in \mathbb{R}^{n_G}$ and $\mu^* \in \mathbb{R}^{n_H}$ such that the following conditions are satisfied at (w^*, λ^*, μ^*):

$$\nabla_w \mathcal{L}(w^*, \lambda^*, \mu^*) = 0, \tag{2.9a}$$
$$G(w^*) = 0, \tag{2.9b}$$
$$H(w^*) \geq 0, \tag{2.9c}$$
$$\mu^* \geq 0, \tag{2.9d}$$
$$\mu_i^* H_i(w^*) = 0, \; i \in \mathcal{H}. \tag{2.9e}$$

The point (w^*, λ^*, μ^*) is called a Karush-Kuhn-Tucker (KKT) point. The KKT conditions were originally derived by Karush [Kar39] in 1939. In 1951 Kuhn and Tucker [KT51] rediscovered the conditions. At active constraints the corresponding Lagrange multipliers may become zero. The active constraints with vanishing multipliers are often called weakly active whereas those active constraints with strictly positive multipliers are said to be strongly active. The complementary condition (2.9e) requires multipliers belonging to inactive constraints to be zero. A proof of the above theorem can e.g. be found in the book of Fletcher ([Fle87]). For a given NLP and a solution point w^* the uniqueness of the triple (w^*, λ^*, μ^*) is guaranteed by the LICQ (2.8).

Theorem 2 (Second order necessary conditions)
Let w^* be a local solution of (2.3) and the LICQ holds. Furthermore let the Lagrange multipliers satisfy the KKT conditions (Theorem 1). For every vector $\delta \in \mathbb{R}^{n_w}$ which satisfies

$$\begin{aligned} \nabla_w G_i(w^*)^T \delta &= 0, \; i \in \mathcal{G} \\ \nabla_w H_i(w^*)^T \delta &= 0, \; i \in \mathcal{H} \end{aligned}$$

it follows that

$$\delta^T \Delta_w \mathcal{L}(w^*, \lambda^*, \mu^*)\delta \geq 0, \tag{2.10}$$

where

$$\Delta_w \mathcal{L}(w^*, \lambda^*, \mu^*) = \frac{\partial^2 \mathcal{L}}{\partial w^2}(w^*, \lambda^*, \mu^*)$$

denotes the Hessian matrix of the Lagrangian function $\mathcal{L}$.

A proof of the above theorem can be found e.g. in the book of Nocedal and Wright [NW99].

Theorem 3 (Second order sufficient conditions)
Let w^* be a KKT point with the multipliers λ^* and μ^*. For every non-zero vector $\delta \in \mathbb{R}^{n_w}$ which satisfies the condition (2.10) and

$$\nabla_w H_i(w^*)^T\delta \;=\; 0, \quad \text{for all } \; i \in \mathcal{A}(w^*) \cap \mathcal{H} \; \text{ with } \; \mu_i^* > 0, \tag{2.11}$$

$$\nabla_w H_i(w^*)^T\delta \;\geq\; 0, \quad \text{for all } \; i \in \mathcal{A}(w^*) \cap \mathcal{H} \; \text{ with } \; \mu_i^* = 0, \tag{2.12}$$

w^* is a strict local minimizer if

$$\delta^T \Delta_w \mathcal{L}(w^*, \lambda^*, \mu^*)\delta > 0. \tag{2.13}$$

To pass from the second order necessary conditions (Theorem 2) to the second order sufficient conditions just the condition (2.10) has to be strictly fulfilled, in addition, the active inequality constraints with vanishing Lagrange multipliers have to be treated separately.

2.2 The SQP method

The sequential quadratic programming approach (SQP) is well adapted to problems with non-linearities. In order to find a KKT point $y^* = (w^*, \lambda^*, \mu^*)$ of the general NLP (2.3)

$$\min_{w \in \mathbb{R}^{n_w}} F(w) \quad \text{s.t.} \quad \begin{cases} G(w) &= & 0 \\ H(w) &\geq & 0, \end{cases}$$

we replace the original problem by a sequence of easier subproblems. To this end we locally replace the *Lagrangian* by a quadratic approximation.
To find the local minimizer w^* with the Lagrange multipliers λ^* and μ^* we introduce the sequence

$$y_{k+1} = y_k + \alpha_k \delta_k \tag{2.14}$$

again using the abbreviation $y_k = (w_k, \lambda_k, \mu_k)$. δ_k describes the step direction, $\alpha_k \in]0, 1]$ the strictly positive relaxation factor. We intend to generate a sequence of approximations $y_k = (w_k, \lambda_k, \mu_k)$ to both the local solution vector w^* itself and to the Lagrange multipliers λ^* and μ^*.
The solution of the sequentially solved problem is the local minimizer of the quadratic function

$$\frac{1}{2}\delta^T \nabla_w^2 \mathcal{L}(w_k, \lambda_k, \mu_k)\delta + \nabla_w F(w_k)\delta.$$

Referring to the Taylor series expansion of the Lagrangian around w_k, we skipped the constant term $\mathcal{L}_k$ and used the equivalence

$$\nabla_w \mathcal{L}(w^*, \lambda, \mu)^T \delta \equiv \nabla F(w^*)^T \delta.$$

This holds if δ is orthogonal to the gradients of both the equality constraints and the active inequality constraints, and if the Lagrange multipliers of the inactive inequality constraints vanish.
We of course require the solution w^* to be feasible whereas intermediate points need not be. Hence we can linearly approximate the equality and inequality conditions

$$\begin{aligned} G(w_k) + \nabla_w G(w_k)^T \delta &= 0 \\ H(w_k) + \nabla_w H(w_k)^T \delta &\geq 0 \end{aligned}$$

forcing the violation of the constraint to vanish in the linear approximation in every step.
Combining the above equations, we can formulate the resulting quadratic program (QP)

$$\min_{\delta \in \mathbb{R}^{n_w}} \frac{1}{2}\delta^T \nabla_w^2 \mathcal{L}(w, \lambda, \mu)\delta + \nabla_w F(w)\delta$$

$$\text{subject to } \begin{cases} G(w_k) + \nabla_w G(w_k)^T \delta &= 0 \\ H(w_k) + \nabla_w H(w_k)^T \delta &\geq 0. \end{cases}$$

Replacing the second derivative of the Lagrangian by a more general matrix $\mathcal{W}$ and allowing $\delta \in \amalg$ we obtain the subproblem

$$\min_{\delta \in \amalg} Q_k(\delta) \;\text{ subject to } \begin{cases} G(w_k) + \nabla_w G(w_k)^T \delta = 0 \\ H(w_k) + \nabla_w H(w_k)^T \delta \geq 0 \end{cases} \tag{2.15}$$

with

$$\mathcal{Q}_k(\delta) := \frac{1}{2}\delta^T \nabla_w^2 \mathcal{W}_k \delta + \nabla_w F(w)^T \delta.$$

SQP methods mainly differ in the choice of the step length α_k used for the globalization, the choice of the matrix $\mathcal{W}$ and in the choice of the domain $\amalg$. The first sequential quadratic programming algorithm was proposed by Wilson in 1963 [Wil63].

Focusing on the quadratic subproblem (2.15) we directly see that, due to the linear constraints the constraint qualification conditions, Theorem 1 forces the existence of a KKT point for the subproblem.

In the subsequent part we focus on a special SQP method leading us to some theoretical insight.
For the choice $\alpha_k \equiv 1$, $\mathcal{W}$ an exact Hessian of the Lagrangian, and $\Pi = \mathbb{R}^{n_w}$ we obtain the Newton iteration scheme

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}(y_k) & -\nabla_w G(w_k) \\ -\nabla_w G(w_k) & 0 \end{pmatrix} \cdot \delta_k + \begin{pmatrix} \nabla_w \mathcal{L}(y_k) \\ G(w_k) \end{pmatrix} = 0$$

with $y_k = (w_k, \lambda_k)$ and

$$\delta_k = \begin{pmatrix} w_{k+1} - w_k \\ \lambda_k \end{pmatrix}$$

excluding the inequality constraints for the moment.

Theorem 4 (Convergence of the SQP method)
If the initial guess $y_0 = (w_0, \lambda_0)$ is sufficiently close to $y^* = (w^*, \lambda^*)$ satisfying the second order sufficient conditions (Theorem 3) and if the matrix

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}(y_0) & -\nabla_w G(w_0) \\ -\nabla_w G(w_0) & 0 \end{pmatrix}$$

has full rank, then the sequence of iterates y_k converges of second order to y^*, i.e., there exists a constant $c > 0$ with

$$\|y_{k+1} - y^*\| \le c\|y_k - y^*\|^2.$$

A proof of Theorem 4 can be found in Fletcher [Fle87].

The initial choice of y_0 is crucial for the convergence of the SQP algorithm. It is more important to have an accurate guess for w_0 than for the Lagrange multipliers.

The classical SQP algorithm has excellent local convergence properties. and clear Nevertheless it suffers from several severe drawbacks.

At points far from the solution, the QP (2.15) may been unbounded or may not even exist. In order to cope with this drawback, a trust-region approach has been proposed. Introducing an upper bound $0 < \mathbf{u}_k$ for the step length $\|\delta_k\| \leq \mathbf{u}_k$ in (2.15) removes the possibility of unbounded correction steps but the restriction might cause the problem to become infeasible. Another idea is to add a Levenberg-Marquardt term to the Hessian $\nabla^2_w \mathcal{L} + \alpha \mathbb{1}$. This may help to preserve the problem from becoming unbounded but leads to the classical Levenberg-Marquardt algorithm drawbacks like degenerate solutions etc. (see e.g. [Cha79]). Other ways to circumvent the mentioned problems are proposed in Fletcher [Fle71].

In addition it turns out that the second derivatives $\nabla^2_w \mathcal{L}(y_k)$ of the Lagrangian required for the classical SQP approach might be difficult to obtain. Especially before the development of automatic differentiation it was hard if not sometimes even impossible to reliably obtain second derivative information. The calculation of the exact Hessian remains expensive (e.g. [Sch99a]). In the following we give a brief motivation for classical Quasi-Newton methods and then turn to the constrained optimization case. For a more detailed discussion refer e.g. to Davidon ([Dav91]) or to Fletcher and Powell ([Fle95], [Pow78b], [Pow85]).

2.2.1 Quasi-Newton methods

Changes in the gradients of the Lagrangian can be measured and used to find an approximate to the Hessian. Exploiting those information leads to significant improvement over steepest descent, under some little restrictive assumptions two-step superlinear convergence can be proved.

Since we do not provide any second derivative information by the integrator (see Chapter 3) but curvature information of the sensitivities is required for the SQP algorithm we provide this approximation to the Lagrangian Hessian $\nabla^2 \mathcal{L}$ by usage of Quasi-Newton methods.

In the following section we will first give an introduction to the unconstrained case of quasi-Newton methods and then concentrate on the constrained case.

2.2.1.1 Unconstrained case

Quasi-Newton methods do not directly use any second derivative information. On each iterate they only require knowledge of the gradient of the objective. Let us introduce the quadratic model

$$\mathcal{M}_k(\delta) := \frac{1}{2}\delta^T \mathcal{B}_k \delta + \nabla_w F(w_k)^T \delta + F(w_k) \tag{2.16}$$

at the current iterate w_k, with $\mathcal{B}_k : \mathbb{R}^{n_w} \mapsto \mathbb{R}^{n_w}$ symmetric and positive definite. The matrix $\mathcal{B}_k$ intended to be an approximation to the exact Hessian $\nabla_w^2 \mathcal{L}$ will be updated or revised in every iteration.

The step $\hat{\delta}_k$ towards the minimizer of (2.16) can be calculated by solving the linear system

$$\mathcal{B}_k \hat{\delta}_k = \nabla_w F(w_k).$$

Davidon's idea was to avoid the explicit calculation of the Hessian and instead simply update it and account for the curvature measured in the last step. With the step sequence

$$w_{k+1} = w_k + \alpha_k \delta_k$$

in complete analogy to the sequence (2.14) we obtain the *secant equation*

$$\mathcal{B}_{k+1} \alpha_k \delta_k = \nabla_w F(w_{k+1}) - \nabla_w F(w_k) \tag{2.17}$$

demanding the *curvature condition*

$$\alpha_k \delta_k \cdot (\nabla_w F(w_{k+1}) - \nabla_w F(w_k)) > 0 \tag{2.18}$$

to be accomplished. For non-convex functions this condition will not automatically hold. The line-search parameter α_k has to be chosen to satisfy the Wolfe conditions. Since the *secant condition* (2.17) imposes only n_w conditions, the symmetric matrix $\mathcal{B}_k$ remains under-determined for $n_w > 1$. One way is to allow only a rank one update

$$\mathcal{B}_{k+1} = \mathcal{B}_k + \frac{((\nabla_w F(w_{k+1}) - \nabla_w F(w_k)) - \mathcal{B}_k \delta_k)\, \delta_k^T}{\delta_k^T \delta_k}. \tag{2.19}$$

This update formula is often called *least change secant update* since the Hessian approximation $\mathcal{B}_k$ changes minimally in every iteration satisfying the *secant equation* (2.17) (e.g. [Bro67],[Dav68],[FM90]).

Another way to uniquely determine $\mathcal{B}_{k+1}$ is to ask for minimal changes from $\mathcal{B}_k$ to $\mathcal{B}_{k+1}$ with respect to a certain norm

$$\min_{\mathcal{B}} \|\mathcal{B} - \mathcal{B}_k\| \ \text{ s.t. } \left\{ \begin{array}{rcl} \mathcal{B} & = & \mathcal{B}^T \\ \mathcal{B}\alpha_k\delta_k & = & \nabla_w F(w_{k+1}) - \nabla_w F(w_k). \end{array} \right. \tag{2.20}$$

A common choice for the norm is the weighted Frobenius norm leading to the Davidon-Fletcher-Powell (DFP) update formula for the Hessian approximation. When substituting

$$\mathcal{D}_k = \mathcal{B}_k^{-1}$$

with the abbreviation

$$\Upsilon_k = \nabla_w F(w_{k+1}) - \nabla_w F(w_k)$$

we obtain the DFP-formula

$$\mathcal{D}_{k+1} = \mathcal{D}_k - \frac{\mathcal{D}_k \Upsilon_k \Upsilon_k^T \mathcal{D}_k}{\Upsilon_k^T \mathcal{D}_k \Upsilon_k^T} + \frac{\alpha_k \delta_k \delta_k^T}{\Upsilon_k^T \delta_k} \tag{2.21}$$

for the inverse of the approximation to the Hessian. The last two terms on the right hand side of the formula are of rank one, and thus the complete modification of $\mathcal{D}_k$ is of rank two.
An alternative to the DFP update formula is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula. It is obtained in a similar fashion like the DFP-formula with the main difference that the conditions (2.20) are forced for the inverse $\mathcal{D}_k$ of the Hessian approximation instead of forcing them for the approximation of the Hessian $\mathcal{B}_k$ itself. Under these modified conditions we uniquely obtain $\mathcal{D}_{k+1}$ via the BFGS update formula

$$\mathcal{D}_{k+1} = \left(\mathbb{1} - \frac{\alpha_k \delta_k \delta_k^T}{\Upsilon_k^T \delta_k} \right) \cdot \mathcal{D}_k \cdot \left(\mathbb{1} - \frac{\alpha_k \delta_k \delta_k^T}{\Upsilon_k^T \delta_k} \right) + \frac{\alpha_k \delta_k \delta_k^T}{\Upsilon_k^T \delta_k}. \tag{2.22}$$

Applying the Sherman-Morrison formula to (2.22), we obtain an update formula for the approximation of the Hessian itself rather than for the approximation of the inverse:

$$\mathcal{B}_{k+1} = \mathcal{B}_k - \frac{\mathcal{B}_k \delta_k \delta_k^T \mathcal{B}_k}{\delta_k^T \mathcal{B}_k \delta_k} + \frac{\Upsilon_k \Upsilon_k^T}{\alpha_k \Upsilon_k^T \delta_k}. \tag{2.23}$$

The costs per iteration are of order of magnitude $\mathcal{O}(n_w^2)$. It can also be shown that the algorithm converges q-superlinearly.

The DFP- and the BFGS-formula are duals of one another. One can be obtained from the other by interchanging $\alpha_k\delta_k = w_{k+1} - w_k$ with Υ_k and $\mathcal{B}_k$ with $\mathcal{D}_k$ in (2.21)

and (2.23) respectively. Both rank two update formulas are members of the Broyden class ([Bro67]). The self-correction ability of the update formulas concerning bad approximations of the Hessian matrices via $\mathcal{D}_k$ in the kth step is very effective for the BFGS update formula (detailed discussions can be found in [Noc92]). The self-correction performance of the DFP update formulas is in general less effective. This property is believed to be the reason for the generally poorer practical performance of the DFP update compared to the BFGS method. One has to keep in mind that the self-correction ability of BFGS only holds when an adequate line search is performed, i.e., when the Wolfe line search conditions ensure that the appropriate curvature information can be exploited.
Global convergence of the BFGS method can not be proven for general nonlinear objectives, although the method is remarkably robust in practice. A general discussion can be found e.g. in Ge and Powell ([GP83]). A discussion of the restrictions necessary to guarantee global convergence for the BFGS-method can be found in [Pow76].

2.2.1.2 Constrained case

Since quasi-Newton approximations have proved very efficient in the unconstrained case, it seems reasonable to transfer the ideas to the approximation of the Lagrangian Hessian $\nabla^2_w \mathcal{L}(y_k)$. Several SQP-algorithms have been proposed based on varying update strategies. Due to the practical performance of the BFGS update we focus on this algorithm.

We use the abbreviation

$$\Upsilon_k = \nabla_w \mathcal{L}(w_{k+1}) - \nabla_w \mathcal{L}(w_k). \tag{2.24}$$

Powell suggested the usage of a modified BFGS update ([Pow77]) keeping the approximation of the Hessian positive definite even if $\nabla^2_w \mathcal{L}(w_k)$ is not positive. For the constrained case the curvature condition from the unconstrained case (2.18) has to be modified to

$$\alpha_k \delta_k \cdot (\nabla_w \mathcal{L}(w_{k+1}) - \nabla_w \mathcal{L}(w_k)) = \alpha_k \delta_k \cdot \Upsilon_k > 0.$$

It can happen that this curvature condition can not be satisfied for any $\alpha_k > 0$. To ensure that the update is always well-defined we update the Hessian approximation via the BFGS update formula (2.23) we replace Υ_k by

$$\hat{\Upsilon}_k := \eta_k \Upsilon_k + \alpha_k (1 - \eta_k) \mathcal{D}_k \delta_k, \quad \eta_k \in]0, 1] \tag{2.25}$$

with the scalar η_k given by

$$\eta_k = \begin{cases} 1 & \text{if} \quad \delta_k^T \hat{\Upsilon}_k \geq \epsilon_\eta \alpha_k \delta_k^T \mathcal{D}_k \delta_k \\ \frac{(1-\epsilon_\eta)\alpha_k \delta_k^T \mathcal{D}_k \delta_k}{\alpha_k \delta_k^T \mathcal{D}_k \delta_k - \delta_k^T \hat{\Upsilon}_k} & \text{if} \quad \delta_k^T \hat{\Upsilon}_k < (1-\epsilon_\eta)\alpha_k \delta_k^T \mathcal{D}_k \delta_k. \end{cases} \tag{2.26}$$

The parameter $\epsilon_\eta \in [10^{-1}, 2 \cdot 10^{-1}]$ is an empirical parameter chosen so that $\hat{\Upsilon}_k$ is closest to Υ_k under the condition

$$\delta_k \hat{\Upsilon}_k \geq \epsilon_\eta \alpha_k \delta_k^T \mathcal{D}_k \delta_k > 0.$$

A detailed discussion can be found in [Pow85]. We obtain the modified update BFGS formula with the slight modification that Υ_k in equation (2.23) is replaced by $\hat{\Upsilon}_k$ from equation (2.24). The modified update BFGS formula guarantees the positive definiteness of the Hessian approximation. For vanishing η_k the approximation of the Hessian remains unchanged from step k to step $k+1$. $\eta_k \equiv 1$ leaves the original unmodified BFGS update formula unchanged (2.23). The value η_k is responsible for producing a positive definite matrix interpolation between the unmodified matrix obtained via the unchanged BFGS update formula and the current iterate $\mathcal{D}_k$.
We use the modified l_1 penalty function

$$\Psi(w, \varpi, \tau) = F(w) + \sum_{i=1}^{n_G} \varpi_i |G_i(w)| + \sum_{j=1}^{n_H} \tau_j |\min(0, H_j(w))| \tag{2.27}$$

with the weights $\varpi_{0,k} = |\lambda_{0,k}^q|$ and $\tau_{k,j} = |\mu_{k,j}^q|$ chosen for the first iterate. Later on the weights are updated by the formulas

$$\varpi_{k,i} = \max\left(|\lambda_{k,i}^q|, \frac{1}{2}\left(\varpi_{k-1,i} + |\lambda_{k,i}^q|\right) \right)$$

and

$$\tau_{k,j} = \max\left(\mu_{k,j}^q, \frac{1}{2}\left(\tau_{k-1,j} + |\mu_{k,j}^q|\right) \right)$$

where λ^q and μ^q again denote the Lagrange multipliers for the quadratic subproblem (2.15). The choice of $\varpi_{k,i}$ and $\tau_{k,j}$ allows a stepwise adaptation of the penalty term. This prevents that too much effort is invested in the reduction of constraint infeasibilities causing significantly shorter steps. This heuristic strategy was first proposed by Powell.

Diverse line-search strategies have been proposed by Powell himself [Pow78a] or inexact line-search strategies by Dennis et al. [DS89]. A comprehensive overview over line-search strategies can be found in [NW99].

As first shown by Maratos in [Mar78] the usage of *non-smooth* merit-functions like (2.27) may cause the algorithm to refuse full steps even very close to the solution (e.g. [Pow84]) reducing the local convergence rate to be q-linear. In order to avoid this potential inefficiency we employ a *non-monotone* strategy. The watchdog technique (Chamberlain et al. [CLPP82]) introduces a two level criterion for the acceptance of new steps. On one level we require the classical descent condition

$$\mathcal{M}_k(\alpha) \leq \mathcal{M}_k(0) + \alpha_k c \left(\partial_\alpha \mathcal{M}_k(\alpha)\right)_{\alpha=0} \tag{2.28}$$

with

$$\mathcal{M}(\alpha) = \Psi(w + \alpha\delta, \varpi, \tau)$$

but allow - on the second level - this condition to be violated for at most n subsequent steps. If a sufficient reduction of the merit function is not obtained after $m \leq n$ steps on the second level - so with temporary violation of the descent criterion (2.28) - we return to the last step before the violation and perform the step with possibly smaller step-length but conform to the descent condition.
The value c is in general chosen quite small: we usually apply $c = 10^{-4}$ like Nocedal [NW99] or Dennis [DS83] propose. The so-called *Armijo* condition enforces a sufficient decrease of the function $\mathcal{M}(\alpha)$.

Despite the fact that we would be interested in at least a local minimizer of the function $\mathcal{M}(\alpha)$ we restrict ourselves to an *inexact line-search*, significantly reducing the number of function evaluations in every step by only fulfilling Condition (2.28). We choose the step length parameter α_k to be as large as possible by not minimizing the quadratic approximation of the penalty function (2.27) but trying to estimate the largest possible step length allowed by the *Armijo* condition every nth step (watchdog).

Other strategies to circumvent the Maratos effect are to introduce a merit function that does not suffer from the Maratos effect or second order correction strategies (e.g. [CC82]). The latter is also available in the software package MUSCOD-II, see [Lei99].

We use the KKT-tolerance

$$\left|\nabla_w F(w_k)^T\right| + \sum \left|\lambda_{k,i} G_i(w_k)\right| + \sum \left|\mu_{k,j} H_j(w_k)\right| \leq \varsigma \tag{2.29}$$

where ς signifies the accuracy intended to obtain as a convergence criterion whether a KKT-point y^* is reached with sufficiently high accuracy.

An overview over implementatory details on modern SQP-methods can be found in Lalee et al. [LNP98].

Reduced-Hessian Approximations

In the beginning of this chapter we presented the classical SQP method. In the following we will briefly explain the special reduced SQP approach originally proposed by Schulz ([Sch96]) and Leineweber ([LBBS03]) allowing of an efficient exploitation of the highly structured optimization problem. The idea is to decompose the search space from which the correction step δ_k in (2.14) is computed into two subspaces $\Omega_k^{\mathcal{R}}$ and $\Omega_k^{\mathcal{N}}$ spanning the full space

$$\Omega_k = (\Omega_k^{\mathcal{R}}, \Omega_k^{\mathcal{N}}), \ \ \Omega_k^{\mathcal{R}} \in \mathbb{R}^{n \times n_{\mathcal{R}}}, \ \ \Omega_k^{\mathcal{N}} \in \mathbb{R}^{n \times n_{\mathcal{N}}} \tag{2.30}$$

with $n_{\mathcal{R}} + n_{\mathcal{N}} = n$. We partition the NLP variables $w = (w_1, w_2)$ in such a manner that $\nabla_{w_1} G_{1,k}^T$ is of full rank. The linearized constraints of G_1 are used to set up the respective coordinate basis.

The correction step δ_k can then be written as

$$\delta_k = \underbrace{\begin{pmatrix} \mathbb{1} \\ 0 \end{pmatrix}}_{\Omega_k^{\mathcal{R}}} z_k^{\mathcal{R}} - \underbrace{\begin{pmatrix} \nabla_{w_1} G_{1,k}^{T}{}^{-1} \nabla_{w_2} G_{1,k}^T \\ -\mathbb{1} \end{pmatrix}}_{\Omega_k^{\mathcal{N}}} z_k^{\mathcal{N}}. \tag{2.31}$$

The Null space component $z_k^{\mathcal{N}}$ is the solution of the QP

$$\min_{z^{\mathcal{N}}} \left(\nabla F_k^T \Omega_k^{\mathcal{N}} z_k^{\mathcal{N}} \right.$$
$$\left. + \frac{1}{2} z_k^{\mathcal{N}^T} \underbrace{\Omega_k^{\mathcal{N}^T} \nabla_{w_k}^2 \mathcal{L}_k \Omega_k^{\mathcal{N}}}_{\mathcal{B}_k^{\mathcal{N}}} z_k^{\mathcal{N}} + \underbrace{z^{\mathcal{R}^T} \Omega_k^{\mathcal{R}^T} \nabla_{w_k}^2 \mathcal{L}_k \Omega_k^{\mathcal{N}} z^{\mathcal{N}}}_{\text{cross term}} \right) \tag{2.32a}$$

subject to

$$\begin{cases} G_{2,k} + \nabla G_{2,k}^T \Omega_k^{\mathcal{N}} z_k^{\mathcal{N}} + \nabla G_{2,k}^T \Omega_k^{\mathcal{R}} z_k^{\mathcal{R}} = 0 \\ H_k + \nabla H_k^T \Omega_k^{\mathcal{N}} z_k^{\mathcal{N}} + \nabla H_k^T \Omega_k^{\mathcal{R}} z_k^{\mathcal{R}} \geq 0 \end{cases} \tag{2.32b}$$

with the range space component $z^{\mathcal{R}}$ given by

$$G_k + \left(\nabla G_k^T \Omega_k^{\mathcal{R}}\right) z_k^{\mathcal{R}} = 0$$

for the NLP

$$\min_{w \in \mathbb{R}^{n_w}} F(w) \quad \text{s.t.} \quad \begin{cases} G_1(w) &= 0 \\ G_2(w) &= 0 \\ H(w) &\geq 0 \end{cases} . \tag{2.33}$$

For the restriction of (2.3) to equality constraints in RSQP methods, $\Omega_k^{\mathcal{N}}$ is chosen to be a basis of the Null space of the constraint Jacobian. As the working sets changes in inequality constraint cases, the classical RSQP approach can hardly be transferred to this set of problems. A more promising approach is the partially reduced SQP method which only uses a subset of the equality constraints to reduce the problem ([Sch96]).

Exploiting the problem structure of the QP arising from the direct multiple shooting method

We define H_k to contain the path constraints (1.12d), the discretized control constraints and the multi-point inequality constraints. $G_{1,k}$ contains the consistency conditions and $G_{2,k}$ the continuity, multi-point equality conditions*. This choice causes the individual intervals to be decoupled allowing of parallel computation of the multiple shooting intervals. $\mathcal{B}_k^{\mathcal{N}}$ is an approximation to the partially reduced Hessian

$$\mathcal{B}_k^{\mathcal{N}} = \Omega_k^{\mathcal{N}T} \nabla_{w_k}^2 \mathcal{L}_k \Omega_k^{\mathcal{N}}$$

calculated in the thesis at hand via BFGS-updates (2.23). In order to obtain the curvature information the difference of the two reduced Lagrange gradients ${\Omega_{k+1}^{\mathcal{N}}}^T \nabla \mathcal{L}_{k+1}$ and $\Omega_k^{\mathcal{N}T} \nabla \mathcal{L}_k$ and the Null space step $z_k^{\mathcal{N}}$ are needed.

Alternatively the continuity and initial conditions could be shifted to $G_{1,k}$ causing the Hessian to loose its block-diagonal structure. In addition the reduced Jacobians $\nabla G_{2,k}^T \Omega_k^{\mathcal{N}}$ and $\nabla H_k^T \Omega_k^{\mathcal{N}}$ receive entries corresponding to variables on nodes before the constraints are defined. This approach originally proposed by Schlöder ([Sch88]) causes the reduced QP (2.32) for fixed initial values for the differential state variables

*For multi-stage formulations $G_{2,k}$ also contains the equality stage transition conditions.

to become independent of the dimension n_x of the differential state vector $x \in \mathbb{R}^{n_x}$. For a more detailed discussion refer to ([SBPD$^+$03] , [Sch04]).

Chapter 3

Numerical solution of Index I DAEs

Various methods for the solution of Index I DAEs are available in numerous implementations. The most common are onestep methods like Runge-Kutta methods (e.g. [Hin80]) and extrapolation methods ([DHZ87]), Rosenbrock methods ([Mic76], [Roc88]) and multistep methods (e.g. [CH52], [Gea71]).

This chapter is organized as follows. We first briefly motivate the choice of a BDF (backward differentiation formulas) method and then explain the BDF algorithm used in our approach. The stability of the method is also shown. In the first two sections we restrict ourselves to DAEs without implicitly given discontinuities.
In the third section we introduce implicitly discontinuous models and show how they can be handled using our BDF method. Finally we give some applications.

3.1 Solving the IVP

In process engineering the extrapolation code LIMEX (Deuflhard et al. [DHZ87] and Ehrig et al. [EN00]) is widely used.
For stiff systems (see 3.1.1) implicit integration algorithms are necessary due to stability demands. In case of stiff systems explicit methods would force the step size to be small for stability reasons and undermine the goal to choose step lengths only bounded by local error restrictions.

A very common solver for DAEs of the form (1.2) is the Code DASSL ([Pet82], [Pet91]). We built on the fast BDF DAE Code DAESOL ([BES88], [BBS99]) developed in the research group of Bock.

3.1.1 Stiffness

Often dynamical models describing chemical processes show a specific behavior called *stiffness*. The time scales on which the different species change vary significantly, often by many orders of magnitude. This phenomenon was first observed by Curtiss and Hirschfelder (see [CH52]):

> *stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit ones.*

In 1963 Dahlquist ([Dah63]) showed that the failure of explicit Runge-Kutta methods is caused by the lack of stability of these methods. Stiff problems can be characterized as follows:

- Slowly changing solutions of the initial value problem (IVP) exist and
- solutions in the direct neighborhood of the slowly changing solutions approach these very quickly

In the literature no standard definition of stiffness is available, quite helpful versions can be found in Shampine et al. ([SG75]) or in the book of Dekker and Vermer ([DV84]). Characteristic of stiffness are the eigenvalues of the Jacobian $\frac{\partial f}{\partial y}$ of the right hand side of a differential equation. Often problems satisfying

$$\left\| \frac{\partial f}{\partial y}(t, y) \right\| (t_f - t_i) \gg 1$$

or $L(t_f - t_i) \gg 1$ with Lipschitz constant L are called *stiff*, when eigenvalues λ of the Jacobian $\frac{\partial f}{\partial y}$ exist with

$$\Re(\lambda) \ll 0$$

(for a similar definition see [SW95]).

Beyond models describing chemical systems, stiff ordinary differential equations also arise from the semi-discretization of parabolic partial differential equations* (e.g. in Lebiedz and Brandt-Pollmann [LBP03]).

*Despite of the fact that it is common to speak about *stiff differential equations* it is more precisely an IVP for this differential equation which may be stiff in some regions.

3.2 A BDF method for the solution of large scale DAE systems

In this section we briefly mention the central aspects on multistep methods. Nevertheless, the main goal of the section is to discuss the main features of the BDF method. After a treatment of ordinary differential equations we move to DAEs in the second part of this section.

Definition 3.1 (Linear multistep method (LMM))
A linear multistep method with k steps for determining a grid function $\hat{y}_h(t)$ for the solution $y(t)$ of the initial value problem

$$\begin{aligned} \dot{y} &= f(t,y) \\ y(t_i) &= y_i \end{aligned} \tag{3.2}$$

on a grid

$$I_h = \{t \in [t_i, t_f] : t = t_n, n = 0, 1, \ldots, N, t_n = t_i + nh\}$$

with the k initial values

$$\hat{y}_h(t_n) = y(t_n), \;\; n = 0, 1, \ldots, k-1$$

is given by the difference equation

$$\sum_{l=0}^{k} \alpha_l \hat{y}_{n+l} = h \sum_{l=0}^{k} \beta_l f(t_{n+l}, \hat{y}_{n+l}) \tag{3.3}$$

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

The condition $|\alpha_0| + |\beta_0| \neq 0$ guarantees that the step number k is defined uniquely, whereas the condition $\alpha_k \neq 0$ ensures that the implicit equation (3.3) can be solved with respect to $\hat{y}_{n+k}$ at least for sufficiently small stepsizes h.

In order to qualify linear multistep methods we introduce the *linear difference operator*

$$L[y(t), h] = \sum_{l=0}^{k} (\alpha_l y(t + lh) - h\beta_l \dot{y}(t + lh)) \tag{3.4}$$

with $y(t) \in \mathcal{C}^{p+1}$ and $t \in [t_i, t_f]$.

Definition 3.5 (Local error of a LMM)
The local error of a LMM of the from (3.1) is defined by the difference

$$y(t_k) - \hat{y}_k$$

where $y(t_k)$ is the exact solution of the IVP $\dot{y} = f(t,y),\ y(t_i) = y_i$ and $\hat{y}_k$ the numerical solution obtained by the LMM (3.1) with the k starting values to be exact.

Theorem 5 (Local error)
The local error le_{n+k} is in linear approximation given by

$$le_{n+k} \doteq \alpha_k^{-1} L[y(t_n), h]. \tag{3.6}$$

A proof can be found in Hairer et al. [HNW93].

Definition 3.7 (Order of consistency)
A linear multistep method of the form (3.3) is consistent of order p if

$$L[y(t), h] = \mathcal{O}(h^{p+1}) \tag{3.8}$$

for $y(t) \in \mathcal{C}^{p+1}[t_i, t_f]$ and $h \to 0$.

3.2.1 Stability

Dahlquist in 1956 [Dah56] introduced the generating polynomials

$$\rho(\xi) \equiv \alpha_k \xi^k + \alpha_{k-1}\xi^{k-1} + \cdots + \alpha_0 \tag{3.9}$$

and

$$\sigma(\xi) \equiv \beta_k \xi^k + \beta_{k-1}\xi^{k-1} + \cdots + \beta_0. \tag{3.10}$$

The linear difference equation (3.3) in its homogeneous form ($h \to 0$ with $n \cdot h = const$) is solved by the trivial solution $\hat{y}_n \equiv 0$ and by $\hat{y}_n = C \cdot \xi^n,\ C \neq 0, \xi \neq 0$ if

$$\psi(\xi) = 0. \tag{3.11}$$

To assess stability we introduce Dahlquists stability test equation

$$\dot{y} = \lambda y.$$

Using the generating equations (3.9) and (3.10), we obtain the characteristic equation

$$\rho(\xi) = z\sigma(\xi) \tag{3.12}$$

where $z = h\lambda$ and $\hat{y}_n = \xi^m$.

Definition 3.13 (Root condition)
The multistep method 3.3 is called stable, if the generating polynomial $\psi(\xi)$ (3.9) satisfies the root condition, i.e.

1. *the roots of the generating polynomial $\psi(\xi)$ lie within or on the unit circle,*
2. *the roots on the unit circle are simple.*

We define the stability domain $\mathcal{S}$.

Definition 3.14 (Stability domain of the linear multistep method)
The area

$$\mathcal{S} = \left\{ z \in \mathbb{C} : \begin{array}{ll} |\xi_l| \leq 1, & \text{if } \xi_l \text{ simple root in 3.12,} \\ |\xi_l| < 1, & \text{if } \xi_l \text{ multiple root in 3.12.} \end{array} \right\} \tag{3.15}$$

is called the stability domain of the linear multistep method (3.3).

A linear multistep method of the form (3.3) is convergent of order p if and only if it is consistent of order p and satisfies the root condition (Dahlquist [Dah56]).

Let $z \in \mathbb{C}$. Then follows $z \in \mathbb{C}^-$ iff $\Re(z) \leq 0$.

Definition 3.16 (A- and $A(\alpha)$-stability)
A linear multistep method is called A-stable, if

$$\mathbb{C}^- \subset \mathcal{S},$$

and it is called $A(\alpha)$-stable, if

$$\mathcal{S} \supset \{z \in \mathbb{C}^- : |\arg(z) - \pi| \leq \alpha\}.$$

Dahlquist's second barrier limits the maximal order of an A-stable linear multistep method to 2. For very stiff systems L-stability is crucial (e.g. [HNW96]). The L-stability is equivalent to the requirement that all roots ξ_l of the characteristical equation (3.12) for all k tend to zero for $|z| \to \infty$, leading to the claim

$$\beta_i = 0, \quad 0 \leq i \leq k-1.$$

With the choice $\beta_k = 1$ this leads to a multistep method of the form

$$\sum_{l=0}^{k} \alpha_l \hat{y}_{n+l} = hf(t_{n+k}, \hat{y}_{n+k}). \tag{3.17}$$

With the requirement

$$\dot{\mathcal{P}}(t_{n+k}) - f(t_{n+k}, \hat{y}_{n+k}) = 0$$

for the interpolation polynomial $\mathcal{P}(t)$ which interpolates the support values $\hat{y}_n$, $\hat{y}_{n+1}$, $\ldots$, $\hat{y}_{n+k-1}$ at $t_n, t_{n+1}, \ldots, t_{n+k-1}$ and the a priori unknown point $(t_{n+k}, \hat{y}_{n+k})$ we obtain the following BDF method.

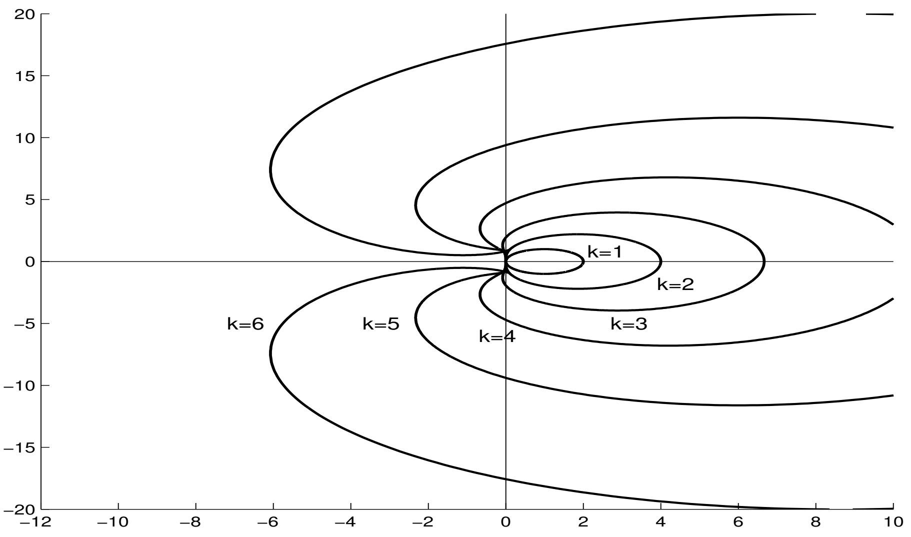

Figure 3.1: The graphics shows the stability domains for BDF methods of orders $k = 1, 2, \ldots, 6$. Obviously, the method is only A-stable for $k = 1$ and $k = 2$. $A(\alpha)$-stability arises for values $k = 1, 2, \ldots, 6$.

Definition 3.18 (BDF method)
Multistep formulas of the form

$$\sum_{l=1}^{k} \frac{1}{l} \nabla^l \hat{y}_{n+k} = h f_{n+k} \tag{3.19}$$

are called backward differentiation formulas (BDF-methods) with the divided differences

$$\begin{aligned}\nabla^0 \hat{y}_{n+k} &= \hat{y}_{n+k} \\ \nabla^l \hat{y}_{n+k} &= \frac{\nabla^{l-1}\hat{y}_{n+k} - \nabla^{l-1}\hat{y}_{n+k-1}}{t_{n+k} - t_{n+k-1}}.\end{aligned}$$

We replace the divided differences $\nabla^i \hat{y}_{n+k}$ in (3.19) by modified divided differences ([Kro79]) reducing the computational costs for updating from one integrator step to the next.

Theorem 6 (Stability of BDF)
The BDF method is $A(\alpha)$-stable for orders $k \leq 6$ and unstable for $k > 6$.

The proof was originally given by Cryer [Cry72].

This guarantees the convergence of BDF methods up to order $k = 6$ if the k initial values are accurate of order $\mathcal{O}(h^k)$.

Due to the particular shape of the stability domains (fig.3.1), classical BDF methods are adequate for problems showing large negative real parts of the eigenvalues but not too large imaginary parts. For problems with highly oscillatory modes (e.g. problem B5 of STIFF DETEST [EHL75]), A-BDF methods (e.g. Fredebeul [Fre98]) proved to be efficient.

Until now our consistency and convergence propositions are restricted to equidistant grids. For practical applications an adaptive choice of the step lengths is crucial. We briefly extend our statements to this scenario. For a more detailed description refer to Bauer ([Bau99]).
In the definition (3.1) we allow of a variable grid

$$I_h = \{t \in [t_i, t_f] : t = t_n,\ n = 0, 1, \ldots, N,\ t_n = t_i + nh_i,\ h_i = t_i - t_{i-1},\ t_i > t_{i-1}\} \tag{3.20}$$

and replace $\alpha_l, \beta_l \in \mathbb{R}$ by $\alpha_{l,n+1}, \beta_{l,n+1} \in \mathbb{R}$ under the same conditions as in (3.1) depending on the quotients $\frac{h_i}{h_{i-1}}$.

Definition 3.21 (Consistency order)
The k step BDF method on a variable grid has the consistency order p if

$$\sum_{l=0}^{k} \alpha_{l,n+1} \mathcal{P}(t_{n+l}) = h_{n+1} \dot{\mathcal{P}}(t_{n+k}) \tag{3.22}$$

for all polynomials $\mathcal{P}(t)$ of order at most p and all grids I_h (3.20).

For variable but finite step sizes the local discretization error of the BDF method with finite coefficients $\alpha_{l,n+1}$ is of order $\mathcal{O}(h_{\max}^{p+1})$ where $h_{\max}$ denotes the maximal step size h_i for functions $f(t, y) \in \mathcal{C}^p$.
Crouzeix et al. ([CL84]) regarded varying step lengths as perturbations of equidistant grids built by time steps of constant length.

Theorem 7 (Stability of the BDF method on variable grids)
The BDF method of the consistency order p satisfies the following conditions:

- *The coefficients $\alpha_{l,n+1}\left(\frac{h_{n+1}}{h_n}, \frac{h_n}{h_{n-1}}, \ldots, \frac{h_{n+1-k}}{h_{n-k}}\right)$ are continuous in a neighborhood of $(1, 1, \ldots, 1)$.*
- *The roots of the generating polynomial (3.9) lie strictly within the unit circle with the exception of one root $\xi = 1$.*

There exist bounds $q_l, q_u \in \mathbb{R}$ with

$$q_l \leq \frac{h_{n+1}}{h_n} \leq q_u, \quad \text{for all} \;\; n.$$

A proof can be found in Crouzeix et al. ([CL84]). Grigorieff ([Gri83]) gave some quite restrictive values for the lower and upper bounds q_l and q_u. In general one can say that *uniform* changes in the step sizes allow more drastic changes in the relation h_n/h_{n-1}. We do not stick to Grigorieffs restrictive boundaries.

Theorem 8
Consider a stable BDF method on a variable grid with bounded coefficients $\alpha_{l,n+1}$ with

- $\|y(t_l) - y_l\| = \mathcal{O}(h^p), \;\; l = 0, 1, \ldots, p-1$ *for initial values,*
- $\frac{h_n}{h_{n-1}} \leq q_u$ *for the step sizes.*

Then for all differential equations

$$\dot{y}(t) = f(t, y), \;\; y(t_i) = y_i$$

and f sufficiently smooth the global discretization error is given by

$$\|y(t_{n+1}) - \hat{y}_{n+1}\| \leq \alpha \cdot h^p_{\max}, \;\; t_{n+1} \in [t_i, t_f].$$

For efficiency reasons we apply a Newton representation of the BDF formulas. The polynomial $\mathcal{P}_{n+1}$ interpolates the past k known values. This demand fixes k of the $k+1$ degrees of freedom of the polynomial of order k. Additionally we require the time derivative of the polynomial at t_{n+1} to coincide with the right hand side of the differential equation

$$\dot{\mathcal{P}}_{n+1}(t_{n+1}) = f(t_{n+1}, \mathcal{P}_{n+1}(t_{n+1})). \tag{3.23}$$

This uniquely determines the polynomial $\mathcal{P}_{n+1}(t)$. The implicit equation (3.23) is solved by usage of a modified Newton method. For the start value we use the value we obtain when evaluating the interpolation polynomial (predictor polynomial) at the point t_{n+1}. The Newton matrices are kept constant (*frozen*) for several steps until certain convergence criteria for the generalized Newton method given in [Bau99] are violated. The criteria build on Bock's *local contraction theorem* [Boc87] measuring the nonlinearity of the Jacobian $\mathcal{J}$ and the quality of the approximation to the inverse of the Jacobian matrix $\hat{\mathcal{J}}^{-1}$.

The *Monitor-Strategy* we apply only reevaluates the expensive derivatives of the right hand side of the differential equation w.r.t. to state variables if the reevaluation of the approximative Jacobian $\hat{\mathcal{J}}$ does not show satisfying convergence behavior [Eic87].

Our main focus - due to the specific structure of the presented applications in the next sections - is on explicit differential algebraic equations of the form

$$\dot{x} = f(t,z) \tag{3.24a}$$
$$0 = g(t,z) \tag{3.24b}$$

$$x(t_i) = x_i \tag{3.24c}$$

where $x \in \mathbb{R}^{n_x}$ denotes the differential, $y \in \mathbb{R}^{n_y}$ the algebraic variables. We subsume differential and algebraic variables in the state vector $z = (x, y) \in \mathbb{R}^{n_x + n_y}$.
DAESOL can treat linear implicit DAEs of the form

$$A(t,z) \cdot \dot{x} = f(t,z) \tag{3.25a}$$
$$0 = g(t,z) \tag{3.25b}$$

$$x(t_i) = x_i. \tag{3.25c}$$

with $A(t,z)$ of full rank. The DAE has to be of index one. Dynamical systems originating in chemical and biotechnological process engineering, respectively, are usually of index I. In two applications shown later in the thesis on hand, we present biotechnological systems which, due to vanishing substances locally become systems of index II. Nevertheless, as a result of nonclassical continuation ([Fil64]), the system can be transformed to a system of index I.
Mechanical systems (e.g. [Mom02], [Sch99b], [Kra04]) are usually of differential index III.

In general it is a non-trivial task to find an algebraic state vector satisfying the algebraic equation in the DAE system (1.1). We apply a relaxed formulation of the DAE system (1.7) allowing of an infeasible start. Consistent initial values can in many cases be obtained using homotopy methods [Bau99].

Error estimation and step size and order strategies

The global error at $T = t_n$ is given by the difference

$$\mathrm{e}_n = \hat{z}_n - z(t_n), \tag{3.26}$$

where $\hat{z}_n$ denotes the numerically obtained approximative solution and $z(t_n)$ the exact solution of the initial value problem (3.24). With $A = \mathbb{1}$ in (3.25a) and the approximation for $h_{n+1}\dot{y}(t_{n+1})$ obtained by the solution of the implicit corrector equation (3.23), we obtain an approximation to the global error composed of the discretization error le_{n+1} and the error $\mathfrak{n}$ arising from interpolating an approximative solution instead of the correct one

$$\begin{pmatrix} \alpha_{l,n+1}\frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} \cdot \mathrm{e}_n \approx \mathfrak{n} - \begin{pmatrix} le_{n+1} \\ 0 \end{pmatrix}.$$

The classical interpolation error arising from interpolating an arbitrary function by polynomials can be shown to be smaller than the discretization error in the limit case of equidistant time grids. $\mathfrak{n}$ is supposed to play a minor role in the global error. If we assume that past states are calculated sufficiently accurate, we can restrict the error estimation as basis for step size and order control to an estimation on the local error and discard $\mathfrak{n}$ in further considerations. The error estimation strategy is similar to the one proposed by Lötstedt's [LP86]. We accept new step sizes iff

$$\begin{aligned} \tilde{le}^{\,k}_{n+1} &:= h_{n+1}\prod_{i=1}^{k}(t_{n+1} - t_{n+1-i}) \cdot \|\nabla^{k+1}\hat{y}_{n+1}\| \\ &\leq \mathrm{TOL} \end{aligned} \tag{3.27}$$

with a predefined error tolerance TOL.
To determine the order and step size for step $n + 2$, we predict the error which will be made in the next step based on the information on hand in step $n + 1$ and we assume an equidistant grid (maximal uniform step size see e.g. [Ble86]). If the predicted step size causes the predicted error (analogous to (3.27)) to satisfy the requirement

$$\tilde{le}^{\,k}_{n+2} \leq \mathrm{TOL}, \tag{3.28}$$

the step size is accepted. If the requirement (3.28) is violated, the step size is reduced using the strategy described in [BBS99].
The order control of our algorithm determines step sizes for the lasts steps order and by one augmented and diminished order with respect to the lasts steps order. The order with the largest step size is chosen ([Eic92]).

Scaling

Since the estimation of the local error (3.27) strongly depends on the chosen norm $\|\cdot\|$, an appropriate scaling is of utmost importance. In the distillation column application (6.1) concentrations of components vary by orders of magnitude. We apply the scaling

$$\breve{z}_{n+1}(i) = |z_{n+1}(i)| \frac{\text{atol}_i}{eps} \tag{3.29}$$

proposed by Petzold ([Pet91]) in the norm

$$\|z_{n+1}\| = \frac{\sqrt{\sum_{i=1}^{n} \left(\frac{z_{n+1}(i)}{\breve{z}_{n+1}(i)} \right)^2}}{n}. \tag{3.30}$$

The value *eps* is a user defined relative error tolerance, atol_i is a componentwise defined absolute error tolerance.

Alternatively, the scaling

$$\breve{z}_{n+1}(i) = \max(|z_{n+1}(i)|, \breve{z}_n(i), \text{atol}_i) \tag{3.31}$$

proposed by Deuflhard [Deu83] can be applied as we do in the context of our biotechnological application processes (6.2).

3.2.2 Storing trajectory information

As a preparation for the optimization of the discontinuous processes but also for performance reasons of the switching point algorithms, we apply a multilevel strategy to save trajectory information in every SQP-step (2.2).
In the context of the solution of the initial value problem *historical* trajectory information is necessary when evaluation of the switching vector and the classical integration process ought to be decoupled (see section 3.3.5.1). The BDF method (3.18) provides an error controlled continuous solution by the local interpolation polynomial (3.22). This representation is called *natural* since the error estimation asymptotically correct controls the error of all interim values ([BS81]).
In order to compare trajectories of different switching chronology a unique characterization of certain back dated trajectories is necessary. We apply a multi-level strategy storing grid and sensitivity information.

3.3 Implicitly defined discontinuities for DAEs

In many practical applications implicitely defined discontinuities occur. They may be due to tabulated material laws, e.g. temperature dependent enthalpies can cause implicit discontinuities. In the distillation column (see Chapter 6 and Appendix A) switching points arise due to weir overflows, out-streams of vapor phases at switching valves or phase transitions but also - as presented in two biotechnological application in Chapter 6 and at the end of this chapter - due to simplified modeling.

Before starting the rack-in of distillation processes the columns are usually floated with an inert gas (often nitrogen). So they are thermodynamically in mono-phase states on every tray. Feeding liquid feed a coexistence of liquid and vapor appears after a mono-bi-phase transition, implicitly given by the thermodynamical equations. This phase transition causes the first partial derivate of the chemical potential with respect to time at constant pressure to jump. This signifies a phase transition of first order based on Ehrenfest's definition †.

Modern DAE IVP solvers require the existence of high order derivatives for order-control, step-size-control and error-control. Despite of ignoring the implicit discontinuities the integrator might produce a reasonable solution but sensitivity information provided by the discretization is in general completely wrong.

Often a smoothing strategy is applied in order to avoid the discontinuities. But for an approximation using a *ramp* with is very close to the discontinuity the numerical problems remain unchanged. Stronger smoothings - so to say smoother ramps - cause a likely serious change of the original problem.

Several approaches to implicitly detect and treat implicit discontinuities have been proposed in the last decades. E.g. Gear et al. ([GO84]) exploit the fact that the error estimates in general increase significantly at switching points. A transition step length is then calculated to keep the local error bounded.

Numerically adapted is the explicit discontinuity treatment [Boc87].

In this section the term *discontinuities* includes discontinuities in the state variables themselves but also discontinuities in the derivatives if not specified in more detail.

In the following we concentrate on initial value problems of the relaxed DAE of the

†The newer definition of phase transitions via order parameters is compatible to Ehrenfest's classification in this special case.

form

$$\left.\begin{array}{rcl}\dot{x}(t) & = & f(t, z(t), p, \operatorname{sgn}(\sigma(t, z(t), p))) \\ 0 & = & g(t, z(t), p, \operatorname{sgn}(\sigma(t, z(t), p))) \\ & & -\alpha(t) g(t_i, z(t_i), p, \operatorname{sgn}(\sigma(t_i, z(t_i), p))) \\ \\ z(t_s^+) & = & z(t_s^-) + \hat{\Delta}^z(t_s, z(t_s), p)\end{array}\right\} t, t_s \in [t_i, t_f] \tag{3.32a}$$

with the initial conditions

$$x(t_i) = x_i. \tag{3.32b}$$

The sign structure (see Chapter 1) of the switching function

$$\sigma(t, z(t), p) \in \mathbb{R}^{n_\sigma} \tag{3.33}$$

describes the individual areas in which the right hand sides are sufficiently smooth. The zero-crossings of the components of the sufficiently smooth switching vector $\sigma(t, z(t), p)$ in (3.33) describe the area boundaries. The function vector $\hat{\Delta}^z(t, z(t), p)$ (3.32a) is the sufficiently smooth jump function which gives a rule to calculate $z(t_s^+)$ (see Chapter 1) based on the left hand limit $z(t_s^-)$ for those switches which do not only cause discontinuous changes in the derivatives of the states by in the states themselves.

3.3.1 Qualification of discontinuities

We assume for the moment that in a surrounding of a switching point t_s just one switching function $\sigma_j(t, z(t), p), \quad j \in \{0, 1, \dots, n_\sigma\}$ of the switching vector shows zero-crossing.

The right hands sides of the IVP in the surrounding of the switching point t_s thus can be divided into three areas

$$\Sigma_j^+ = \{(t, z, p) |\ \sigma_j(t, z(t), p) > 0\}, \tag{3.34a}$$

$$\Sigma_j = \{(t, z, p) |\ \sigma_j(t, z(t), p) = 0\}, \tag{3.34b}$$

$$\Sigma_j^- = \{(t, z, p) |\ \sigma_j(t, z(t), p) < 0\}. \tag{3.34c}$$

We introduce the functions

$$\mathcal{D}\sigma_j(t, z(t), p)^+ := (\sigma_j)_t(t, z(t), p) + (\sigma_j)_z(t, z(t), p) \cdot (f, g)(t, z(t), p)^+, \tag{3.35a}$$

$$\mathcal{D}\sigma_j(t, z(t), p)^- := (\sigma_j)_t(t, z(t), p) + (\sigma_j)_z(t, z(t), p) \cdot (f, g)(t, z(t), p)^- \tag{3.35b}$$

which help to qualify the actual discontinuity. $(f, g)(t, z(t), p)^+$ and $(f, g)(t, z(t), p)^-$ are the right hand sides of the differential-algebraic equation in right and left hand limit to the switching point t_s.
The dependencies of the derivatives of the switching functions are given in (3.35a) and (3.35b). For readibility arguments we skip them in the following.

For vanishing derivatives $\mathcal{D}\sigma_j$ additional derivatives of the switching functions are necessary. In this case we stop our algorithm with a corresponding error message.

In the case of $\mathcal{D}\sigma_j^+ \cdot \mathcal{D}\sigma_j^- > 0$, a classical continuation of the solution exists if no jump in states appears at the zero-crossing of the switching function. The trajectory passes through the discontinuity hypersurface given by (3.34b).
In case of a jump in the state vector the situation is slightly different since even for differing signs of $\mathcal{D}\sigma_j^+$ and $\mathcal{D}\sigma_j^-$ the switching surface might have been left. The regularity assumption than transforms to $\mathcal{D}\sigma_j^{\cdot}\sigma_j > 0$.

For $\mathcal{D}\sigma_j^+ \cdot \mathcal{D}\sigma_j^- < 0$ we distinguish between to fundamentally different cases:

- $(\mathcal{D}\sigma_j^+ > 0) \quad \wedge \quad (\mathcal{D}\sigma_j^- < 0)$:
  The discontinuity manifold can not be left $\rightarrow$ leads to *inconsistent switching* (refer to Section 3.3.7).

- $(\mathcal{D}\sigma_j^+ < 0) \quad \wedge \quad (\mathcal{D}\sigma_j^- > 0)$:
  The discontinuity manifold can be left in both directions $\rightarrow$ Bifurcation.

The differentiability of a DAE of the form (1.1) with respect to initial conditions and control variables at the presence of switching points is fundamental for the treatment of implicit discontinuous processes. In 1987, Bock gave the following theorem based on the embedding theorem for multipoint boundary value problems ([Boc87]).

Theorem 9
Let $I = [t_i, t_f]$ be an interval containing (without loss of generality) exactly one switching-point at $t_s \in I$, which bipartitions the interval into $I^- = [t_i, t_s]$ and $I^+ = (t_s, t_f]$. Let $z(t)$ be a solution of the initial value problem (3.32). The right hand sides $f(t, z(t), p, \operatorname{sgn}(\sigma_j(t, z(t), p)))$ and $g(t, z(t), p, \operatorname{sgn}(\sigma_j(t, z(t), p)))$, the function $\hat{\Delta}^z$ for the jump height and the switching functions are assumed to be sufficiently often continuously differentiable.
Let the regularity conditions

$$\mathcal{D}\sigma_j^+ \cdot \mathcal{D}\sigma_j^- > 0$$

if no jump is given and

$$\mathcal{D}\sigma_j^+ \cdot \sigma_j(t_s, z(t_s), u) > 0$$

for $\hat{\Delta}_j^z \neq 0$ *be satisfied.*

Then exists an area $\mathcal{A} \supset \{(t_s, z(t_s; t_i, z(t_i), p), p)\}$ *with* $z(t_s; t_i, z(t_i), p)$ *sufficiently smooth and all solutions of the IVP in the area* $\mathcal{A}$ *have exactly one switching point.*

A proof for the ODE case can be found in [Boc87].

3.3.2 Detecting switching points

To explicitely handle discontinuities the switching point $t_s \in [t_i, t_{i+1}]$ given by

$$0 = \sigma_j(t_s, z(t_s), p), \qquad j \in \{0, 1, \ldots, n_\sigma\} \tag{3.36}$$

with $\sigma \in \mathbb{R}^{n_\sigma}$ has to be detected with sufficient accuracy. Below we briefly describe the strategy implemented. Nothing but the sign of the switching functions characterizes the active area. We only require precise values of the switching functions in the neighborhood of the points t_s since merely the signs of the switching functions are crucial for the model. This makes it possible to decouple the evaluation of the switching functions from the evaluations of the right hand sides of the model (see 3.3.5.1) and replace it by a cheaper evaluation of an interpolation polynomial (3.37) approximating the actual values of the switching functions. This switching function monitor will be presented in the following.

If a switching function signalizes a transition into another area the first strategy we present requires the model with the old switching structure to be evaluable beyond the area boundary. This can be regarded as an overlap of the areas characterized by the switching vector (3.33).
For areas that can be extended beyond the boundaries we implemented a hierarchical multilevel switching point search. In the following we briefly describe the algorithm for detecting the switching points for the evaluation of the switching vector in every integrator step. In 3.3.5.1 we generalize this switching point search to the case when the integration process and the determination of the actual switching vector configuration are decoupled.

Whenever we detect a sign change of σ_j we apply an iterative strategy to precisely localize the switching point t_s:

Applying linear interpolation using the fact that the switching point is in the interval $[t_l, t_{l+1}]$ leads to an approximation of the switching point. We assume the switching function to be locally invertible.
For the switching point search we do not reevaluate the right hand sides since we can evaluate the *natural interpolation polynomial* (3.37). To solve

$$\sigma_j(t_s, \mathcal{P}(t_s), p) = 0$$

we first apply inverse quadratic interpolation (see e.g. [Boc74]).
Due to the implicit function theorem $\sigma_j(t, \mathcal{P}(t), p)$ is locally invertible. Assume that this inverse exists in the interval $[t_l, t_{l+1}]$[‡]. The idea is to approximate the local inverse by a quadratic polynomial $\sigma_j^\dagger(t)$ with $\sigma_j^\dagger(0) = t_s$. As starting values the values of the switching function at t_l and t_{l+1} are applied. Approximating linearly instead of quadratically would lead to the classical secant method of a local convergence order of at least $0.5(1+\sqrt{5})$.
If the quadratic interpolation does not shrink the search interval satisfactorily we first restrict to linear interpolation and the secant method and if this still does not lead to amelioration we apply a bi-section step of the interval which guarantees global convergence.
In the strategy described above we assume overlapping areas $\mathfrak{R}_j$. For e.g. concentrations of chemical substances this assumption does not need to hold. In these cases the right hand sides of the DAE are possibly inevaluable. Internally, we apply as polynomial representation the continuous solution representation of the last k values and use it as an extrapolation polynomial in these cases.
After switching points the integrator has to be restarted. Due to initially small and subsequently increasing step sizes and low orders, numerous re-decompositions of the Jacobians for the Newton iteration scheme are required. These starting phases therefore turn out to be time consuming.
In the starting phase we apply Runge-Kutta-Starters of higher order as proposed in [BGS88] providing back dated values for the BDF method.

3.3.3 Continuous representation of the trajectories

One major advantage of the BDF multistep methods compared to Deuflhards common one step solver LIMEX is the well accessible continuous solution trajectory

[‡]Since we do not evaluate the switching vector in every integrator step for performance reasons, the interval does not have to fall together with the interval given by the last two integrator step.

representation. This continuous representation is pivotal for the efficient localization of switching points at non-grid points. The self-evident choice is the polynomial

$$\mathcal{P}_{n+1}(t) = \sum_{l=0}^{k} \prod_{i=0}^{l-1} (t - t_{n+1-i}) \nabla^l \hat{y}_{n+1} \tag{3.37}$$

solving the implicit equation (3.23) in the respective interval $[t_{k+n+1}, t_{n+1}]$.

3.3.4 Updating sensitivity information at discontinuities

In the following we will derive the update formulas for the sensitivity matrices at a switching point t_s for equation (3.32).

Each switching point $t_s \in (t_j, t_{j+1})$ is implicitly defined by Condition (3.36).
Bock showed in [Boc87] for ODEs how sensitivity information w.r.t. initial values and parameters can be calculated in the presence of discontinuities. The extension for index I DAEs is straightforward.

In the following we will distinguish between discontinuities in the derivatives of the state variables

$$\lim_{\epsilon \to 0} \dot{z}(t_s + \epsilon) - \lim_{\epsilon \to 0} \dot{z}(t_s - \epsilon) \tag{3.38}$$

with we will in the following abbreviate with κ and discontinuities in the state variables themselves giving by the jump vector $\hat{\Delta}^z$ (3.32a).

The sensitivity matrix describing the sensitivity of the states z_{j+1} at the time t_{i+1} with respect to z_j at t_j is given by

$$\mathfrak{S}_z(t_{j+1}, t_j) = \frac{\partial z_{j+1}}{\partial z_j}. \tag{3.39}$$

The sensitivity matrix $\mathfrak{S}_z(t_{j+1}, t_j)$ is a composition of the three matrices

$$\lim_{\epsilon \to 0} \mathfrak{S}_z(t_s + \epsilon, t_j),$$

the update matrix $\mathfrak{A}_z$ (3.45) and

$$\lim_{\epsilon \to 0} \mathfrak{S}_z(t_{j+1}, t_s + \epsilon). \tag{3.40}$$

Using the implicit function theorem we obtain with the left hand side derivative of σ_j (3.36) the scalar valued function

$$\dot{\sigma}_j = \lim_{\epsilon \to 0} (\sigma_j)_t (t_s - \epsilon) + \left(\lim_{\epsilon \to 0} (\sigma_j)_t (t_s - \epsilon) \right) \cdot \left(\lim_{\epsilon \to 0} \dot{z}(t_s - \epsilon) \right).$$

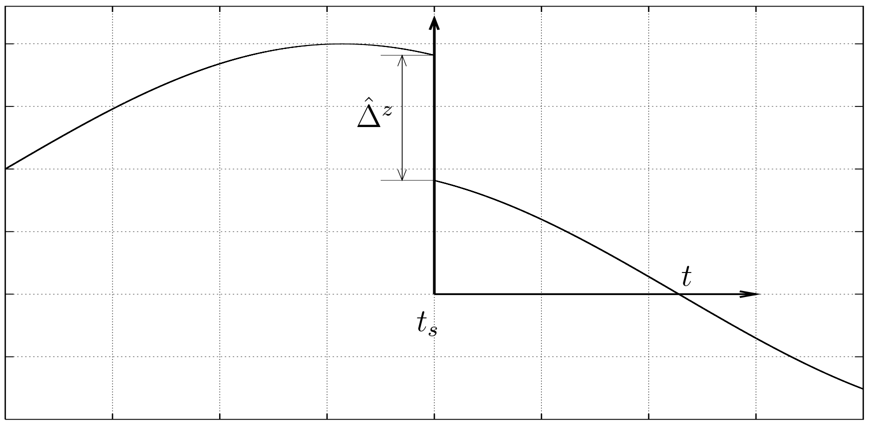

Figure 3.2: Jump height $\hat{\Delta}^z = \hat{\Delta}^z(t_s, z(t_s), p)$ at the switching point t_s

For (3.40) we find

$$\begin{aligned}\lim_{\epsilon\to 0}\mathfrak{S}_z(t_s+\epsilon,t_j) =& \lim_{\epsilon\to 0}\mathfrak{S}_z(t_s-\epsilon,t_j) + \left(\lim_{\epsilon\to 0}\frac{\partial z(t_s-\epsilon)}{\partial t_s} + \hat{\Delta}^z_{t_s}\right)\frac{\partial t_s}{\partial z_j} \\ &+ \hat{\Delta}^z_z\left(\lim_{\epsilon\to 0}\mathfrak{S}_z(t_s-\epsilon,t_j) + \lim_{\epsilon\to 0}\dot{z}(t_s-\epsilon)\frac{\partial t_s}{\partial z_j}\right).\end{aligned} \tag{3.41}$$

The sensitivity part for the right hand limit at the switching point is

$$\frac{\partial z_{j+1}}{\partial t_s} = -\lim_{\epsilon\to 0}\mathfrak{S}_z(t_{j+1}, t_s+\epsilon)\dot{z}(t_s+\epsilon) \tag{3.42}$$

and so with (3.40) we obtain

$$0 = (\sigma_j)_z\left(\frac{\partial z(t_s)}{\partial t_s}\frac{\partial t_s}{\partial z_j} + \lim_{\epsilon\to 0}\mathfrak{S}_z(t_s, t_j-\epsilon)\right) + (\sigma_j)_{t_s}\frac{\partial t_s}{\partial z_j} \tag{3.43}$$

and

$$0 = (\sigma_j)_z\left(\frac{\partial z(t_s)}{\partial t_s}\frac{\partial t_s}{\partial p} + \lim_{\epsilon\to 0}\mathfrak{S}_p(t_s, t_j-\epsilon)\right) + (\sigma_j)_{t_s}\frac{\partial t_s}{\partial p} + (\sigma_j)_p. \tag{3.44}$$

With the abbreviations

$$\mathfrak{A}_z = (\kappa-\gamma)\frac{(\sigma_j)_z}{\dot{(\sigma_j)}} + \mathbb{1} + \frac{\partial\hat{\Delta}^z}{\partial z} \tag{3.45}$$

and

$$\mathfrak{A}_p = (\kappa-\gamma)\frac{(\sigma_j)_p}{\dot{(\sigma_j)}} + \frac{\partial\hat{\Delta}^z}{\partial p} \tag{3.46}$$

we finally obtain the sensitivity matrices

$$\mathfrak{S}_z(t_{j+1}, t_j) = \lim_{\epsilon \to 0} \mathfrak{S}_z(t_{j+1}, t_s + \epsilon) \mathfrak{A}_z \mathfrak{S}_z(t_s - \epsilon, t_j) \tag{3.47}$$

and

$$\begin{aligned} \mathfrak{S}_p(t_{j+1}, t_j) = &\lim_{\epsilon \to 0} \mathfrak{S}_z(t_{j+1}, t_s + \epsilon)(\mathfrak{A}_z \mathfrak{S}_p(t_s - \epsilon, t_j) + \mathfrak{A}_p) \\ &+ \mathfrak{S}_p(t_{j+1}, t_s + \epsilon). \end{aligned} \tag{3.48}$$

For vanishing discontinuity in the derivatives $\kappa(t_s, z(t_s), p)$ (3.38) and no jumps in the state variables $\hat{\Delta}^z(t_s, z(t_s), p)$ (3.32a) themselves the matrix $\mathfrak{A}_z$ (3.45) appears as identity matrix.
If $\kappa(t_s, z(t_s), p)$ (3.38) vanishes and $\hat{\Delta}^z(t_s, z(t_s), p)$ (3.32a) no jumps occur at t_s the matrix $\mathfrak{A}_p$ (3.46) vanishes.

To sum up, the differentiability of the solution with respect to initial values and parameters can be guaranteed for a switching point $t_s \in (\tau_i, \tau_f)$ for implicit discontinuous models of the form (see Theorem 9). The formulas (3.47) and (3.48) explicitely give the updates for the sensitivity information at the switching points.

3.3.5 Algorithmic treatment of implicit discontinuities

Every particular area is characterized by the signs of the switching vector $\text{sgn}(\sigma)$. As long as the signs remain unchanged, the integrator can work in its normal manner, since the right hand sides of the DAE are sufficiently smooth. Area transitions are characterized by sign changes in this switching vector.

The algorithmic realization

1. Determine the sign structure for the switching vector.
2. Start the integration process.
3. Check the switching vector for sign changes during the integration process.
   (a) If after an integrator step a sign change took place, determine the switching point t_s.
      - If no jump took place: Go to 2. with the new start point t_s
      - If a jump function is given: Go to 1. with the new start point t_s

3.3.5.1 Decoupling of the surveillance of the switching vector from the main integration process

In practice the number of arising zero-crossings of switching functions is rare compared to the number of integrator steps. therefore it is little efficient to reevaluate the switching vector in every integrator step. Much can be gained by decoupling the evaluation of the right hand sides from the evaluation of the switching vector.

Since we store trajectory information including we are able to decouple the evaluation of the switching functions and the time discretization given by the step size control. Switchings in general occur rarely compared to the number of integrator steps. Hence we introduce a coarser equidistant time grid

$$t_i = \tau_0^\sigma < \tau_1^\sigma < \ldots < \tau_N^\sigma = t_f$$

and determine the sign structure of the switching vector by evaluating the switching function vector at the grid points τ_i^σ by usage of the *natural interpolation polynomial* (3.37)

$$\sigma(\tau_i^\sigma, \mathcal{P}(\tau_i^\sigma)).$$

Obviously, the evaluation process has to be accomplished retarded with respect to the integration process itself, since the interpolation polynomial has to be available.

In our implementation we allow of a computation of the switching vector in parallel. It is possible to explicitely use the SSE 2 vector unit of Intels Pentium IV processor [Int04]. This can be interpreted as an on board parallelity.
If a switching point t_s is detected, the integration process is stopped and - after a precise localization of the switching point (3.3.2 - 3.3.5) - restarted.

3.3.6 Discontinuity treatment - Two introductory applications

As introductory examples we present two obviously state dependent implicit discontinuous models.

In (6.1.1) a rigorous model of a multi tray distillation column is treated. In the following we virtually extract the feed tray from the column in order to point out the occurrence of implicit discontinuities and give an idea of the complexity of the thermodynamical model. Usually the standard modeling assumption that the vapor

on each tray is negligible is made. Due to the fact that we later focus on rack-in processes of distillation processes we need an approach which does not use this assumption.
We assume an *ideal tray* meaning that vapor and liquid streams leaving the tray are in equilibrium.
Picking just one tray three state dependent discontinuities occur:

- Change from one-phase state (only gas) to bi-phase state (coexistence of liquid and vapor),
- Gas outflow due to overpressure (opening of a valve at a certain barrier),
- Weir overflow.

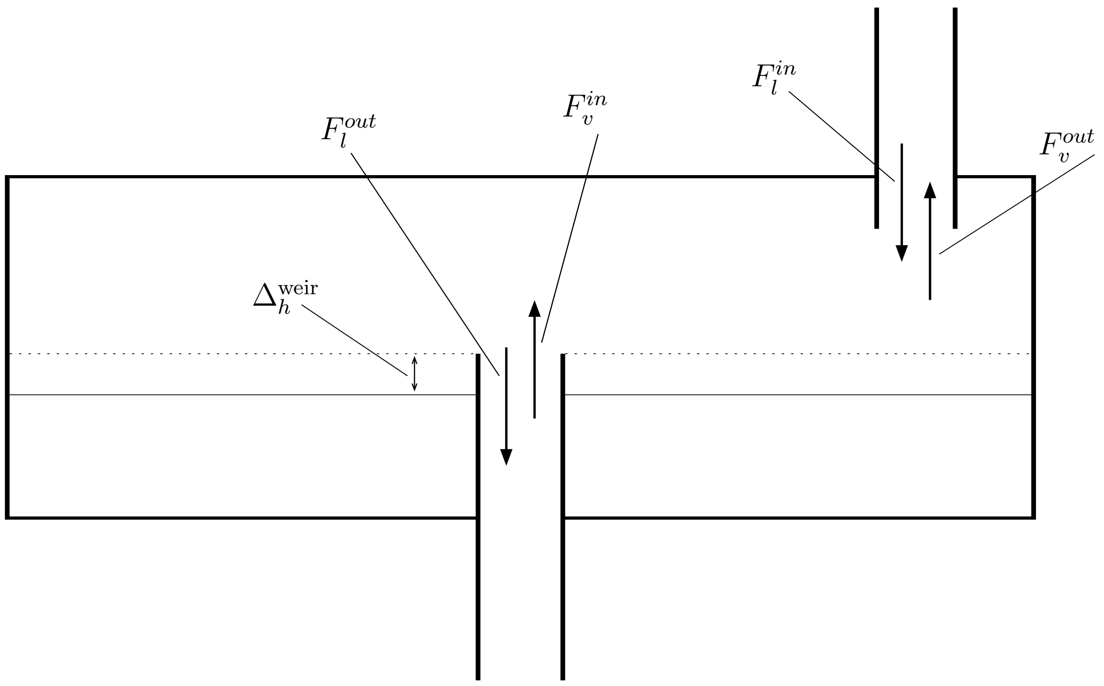

Figure 3.3: Model of a single tray

Initially the columns are usually floated with an inert gas. The model for this single tray consists of 5 differential (molar and energy balances) and 81 algebraic variables arising from the thermodynamic (or thermostatic) description of the system.

The component molar balances in the tray can be described as

$$\frac{d\left(MX_i\right)}{dt} = L_{in}X_{i,in} - LX_i - VY_i$$

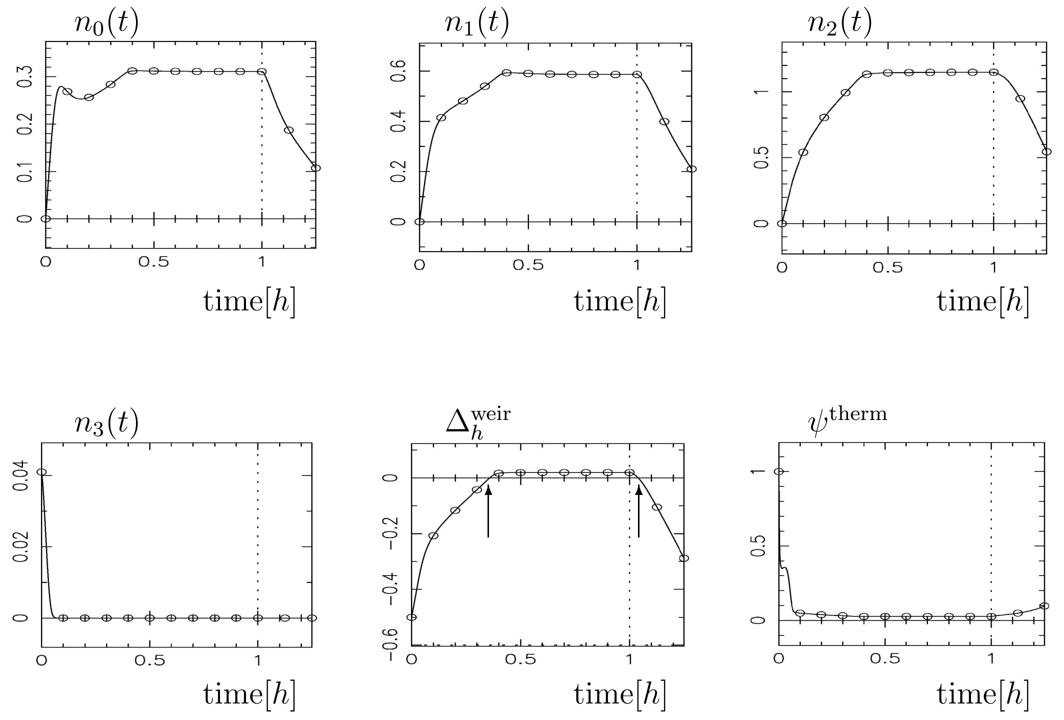

Figure 3.4: Simulation of a rack-in of a single tray: Implicit discontinuity due to weir overflow and later stop of the overflow (vertical arrows). Additionally an implicit discontinuity arises from the transition of an overheated mono-phase state ($\psi^{\text{therm}} > 1$) to the bi-phase state ($0 < \psi^{\text{therm}} < 1$) where a liquid and a vapor phase coexist.

where i represents one of the n_C components in the mixture given into the tray. M stands for the mass of the substances, H describes the enthalpies for the complete holdup, the liquid phase and the vapor phase, respectively. L and V are liquid and vapor stream, Q stands for the added energy. X and Y describe the liquid and vapor molar fractions, k gives the k-values[§]. In our case we regard four components.

$$\frac{d\left(MH^{ges}\right)}{dt} = L_{in}H_{in}^{L} - LH^{L} - VH^{V} + Q$$

describes the total enthalpy balance. The phase equilibrium conditions are given by

$$Y_i = X_i \cdot k_i\left(T, P, X_1, \ldots, Y_1, \ldots\right)$$
$$\sum_{i=1}^{n_C} Y_i = 1.$$

[§]For a detailed description refer to Appendix A.

Starting from an initial state where the tray is floated with an inert gas, the feed stream lets the liquid holdup level transcend the weir at $t \approx 0.35h$ starting a liquid outflow (first arrow in bottom middle plot in Figure 3.4). An implicit model discontinuity of the right hand side is reached.

The switching functions in this example are

- Thermodynamic vapor fraction: $\sigma_0 = \psi^{therm}$
- Overpressure: $\sigma_1 = p - 1$
- Weir overflow: $\sigma_2 = \Delta_h^{weir}$

For a constant feed-stream into the tray the heat supply is increased by 30% at $t = 1$. This causes more liquid holdup to vaporize and consequently the thermodynamic vapor fraction ψ^{therm} to increase (right plot, lower row). The level of the liquid holdup falls under the weir height, a second switching event occurs at $t \approx 1.05h$ (second arrow in bottom right plot in Figure 3.4). Already before $t \approx 0.01h$ the system state changes from a mono-phase state with a vapor phase exclusively to a bi-phase state with a coexistence of gas and vapor induced by the feed-stream into the tray. This transition can be seen in the very left of the bottom right plot in Figure 3.4. For a complete description of the model, see 6.1.1 and Appendix A.

3.3.7 Inconsistent switching

In the following subsection we focus on the case

$$\mathcal{D}\sigma_j^+ > 0 \wedge \mathcal{D}\sigma_j^- < 0. \tag{3.49}$$

At the point t_s we obtain the vanishing component of the switching vector $\sigma_j(t_s, z(t_s), p) = 0$ and

$$\mathcal{D}\sigma_j^- \cdot \mathcal{D}\sigma_j^+ < 0 \tag{3.50}$$

which would not cause the switching function to change its sign. So the initial value problem has no classical solution in this case. Filippov [Fil64] in 1964 introduced the convex hull

$$\mathcal{F} = \{f^\epsilon |\, f^\epsilon = \epsilon f^-(t, z, p) + (1 - \epsilon) f^+(t, z, p), \;\; \epsilon \in [0, 1]\} \tag{3.51}$$

of f^- and f^+ for ordinary differential equations.

For DAEs Filippov's Ansatz also holds, where only the differential equations have to be transformed. The algebraic equations remain unchanged.
Allowing all values of the convex hull (3.51) for the right hand side of the DAE, we obtain an *extended solution.* Bock [Boc87] proposes to choose

$$\epsilon = \frac{\mathcal{D}\sigma^+}{\mathcal{D}\sigma^+ - \mathcal{D}\sigma^-}$$

motivated by the fact that the solution must remain on the manifold (3.34b). Hence if the case (3.50) occurs the differential algebraic equation is replaced by

$$\mathcal{Q}\dot{x} = (f^\epsilon, g)$$

as long as (3.50) remains valid with

$$\mathcal{Q} = \begin{pmatrix} \mathbb{1}_{n_{xd}} & 0 \\ 0 & 0 \end{pmatrix}.$$

Meanwhile the switching function σ is replaced by the switches $\mathcal{D}\sigma^-$ and $\mathcal{D}\sigma^+$. Along the convex hull the DAE system can be interpreted as a system of index II (see section 1.1)

$$\begin{aligned} \dot{x} &= f(t, z(t), p, \operatorname{sgn}(\sigma(t, z(t), p))) \\ 0 &= g(t, z(t), p, \operatorname{sgn}(\sigma(t, z(t), p))) \\ 0 &= \sigma_j(t, z(t), p) \end{aligned} \tag{3.52}$$

since the derivative $(\sigma_j)_z \cdot \dot{z} + (\sigma_j)_t = 0$ also vanishes (see Chapter 1).

Whenever one of the switches $\mathcal{D}\sigma_j^-$ and $\mathcal{D}\sigma_j^+$ changes its sign, the non classical manifold (3.34b) can in general be left, transforming the right hand side back to $(f, g)^-$ or $(f, g)^+$ respectively. For the convex hull this is tantamount to setting ϵ in (3.51) to a value of zero or one.
We thus obtain a three-valued treatment of the switching functions instead of the classical two-valued.
Under certain conditions so called inconsistent switching may appear¶, e.g. in many cases for vanishing substrates in the description of bio-processes via differential algebraic equations. This phenomenon is characterized as follows. At the switching point t_s we obtain

¶Models describing e.g. static friction (coulomb friction) also show this phenomenon.

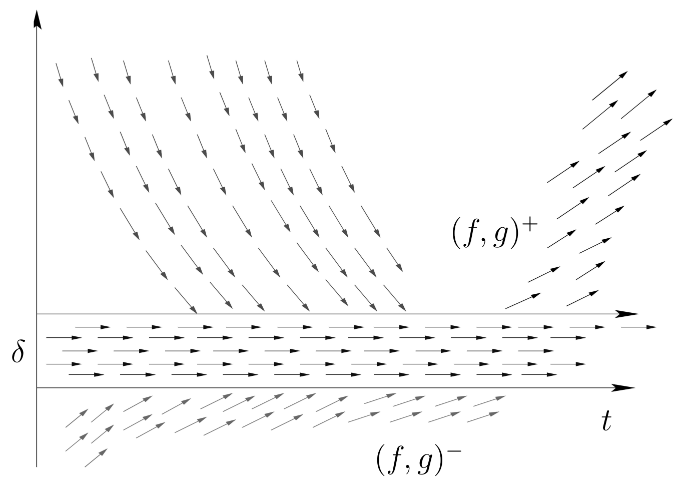

Figure 3.5: The graphics schematically shows the vector field for one concentration in the biotechnological application presented in 3.3.7.1. Numerically we treat a band of the small width δ to fulfill $\sigma_j \equiv 0$. In this band the convex combination of the right hand sides (3.51), replaces the classical right hand side of the differential algebraic equation.

$$\sigma_j(t_s, z(t_s), p) = 0 : \lim_{\varepsilon \to 0} \left(\mathcal{D}\sigma_j|_{t_s+\varepsilon, z(t_s+\varepsilon), p} \cdot \mathcal{D}\sigma_j|_{t_s-\varepsilon, z(t_s-\varepsilon), p} \right) .$$

for the switching function σ_j.

Thus the initial value problem does not have a classical solution. With the convex combination

$$\begin{aligned} \mathcal{Q}\dot{x} &= \epsilon(f,g)^+ + (1-\epsilon)(f,g)^- \\ &= \frac{1}{\mathcal{D}\sigma_j^+ - \mathcal{D}\sigma_j^-} \left(\mathcal{D}\sigma_j^+ (f,g)^- - \mathcal{D}\sigma_j^- (f,g)^+ \right) \end{aligned}$$

we obtain a DAE system of index I. For inconsistent cases the classical two-valued treatment of the switching functions has to be extended to a three-valued treatment. To sum up we obtain the classical switching point treatment for $\mathcal{D}\sigma_j^- \cdot \mathcal{D}\sigma_j^+ > 0$. In case of (3.49) the switching vector itself can be regarded to be frozen until $\mathcal{D}\sigma_j^-$ and $\mathcal{D}\sigma_j^+$ turn to have the same sign.
If both derivatives $\mathcal{D}\sigma_j^-$ and $\mathcal{D}\sigma_j^+$ change sign this leads to bifurcations $\mathcal{D}\sigma_j^- < 0 \wedge \mathcal{D}\sigma_j^+ > 0$ we (see 3.3.1). From this point it would be possible to integrate two different trajectories. We stop our algorithm and error-message this event in this case.

3.3.7.1 A biotechnological example

A quite simple but even so interesting problem showing inconsistent switching is the biotechnological process model described by Kühn et al. [Küh02]. The modeling of substrate masses with Heaviside functions in order to meet the requirement of model changes at vanishing concentrations is characteristic for the modeling of biotechnological processes.

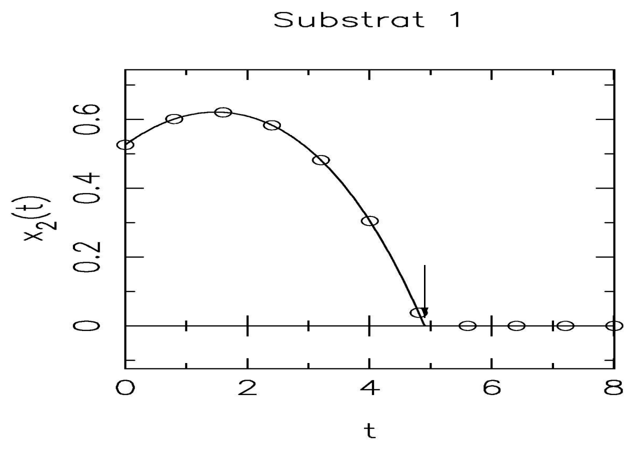

Figure 3.6: At $t \approx 4.9$ time units the substrate 1 vanishes turning the index I system locally into a system of index II. The switching function can be interpreted as algebraic conditions.

In the following we restrict to give the equations and briefly name the variables. A more detailed specification is given in subsection (6.2.1). The model equations are

$$f_0 = \dot{X} = r_X X - \frac{X}{V}\dot{V} \tag{3.53a}$$

$$f_1 = \dot{P} = r_P X - \frac{P}{V}\dot{V} \tag{3.53b}$$

$$\begin{aligned} f_2 = \dot{S}_1 &= -\left(\frac{1}{Y_{X/S_1}} r_X X + \frac{1}{Y_{P/S_1}} r_P X\right) \cdot \Theta(S_1) \\ &\quad - \frac{S_1}{V}\dot{V} + \frac{1}{V}\left(F_A S_{F_A} + F_B S_{F_B}\right) \end{aligned} \tag{3.53c}$$

$$f_3 = \dot{S}_2 = -\left(\frac{1}{Y_{X/S_1}} r_X X\right) \cdot \Theta(S_2) - \frac{S_2}{V}\dot{V} \tag{3.53d}$$

$$f_4 = \dot{V} = F_A + F_B \tag{3.53e}$$

where X denotes the biomass, P the product. The substrates are described by the variables S_1 and S_2.
The reaction rates r_X and r_P are given in (6.14) and (6.15). A brief description of the parameters can as well be found in Subsection (6.2.1), for a more detailed discussion refer e.g. to Kühn ([Küh02]).
Two implicit discontinuities occur due to the model change caused by the Heaviside functions

$$\Theta(x) = \begin{cases} 0, & \text{for} \quad x \leq 0, \\ 1, & \text{for} \quad x > 0. \end{cases} \tag{3.54}$$

In the following we first concentrate on the case of a vanishing substrate S_1 and a strictly positive concentration of the substrate S_2. For vanishing substrate S_1 we obtain a vanishing switching function $\sigma_1 = S_1$. In addition the different signs of the directional derivatives of the switching functions $\mathcal{D}\sigma_1^+$ and $\mathcal{D}\sigma_1^-$ indicate inconsistent switching according to the definition in Section 3.3.1. We obtain

$$\begin{aligned} \mathcal{D}\sigma_1^+ &= f_2^+ \\ \mathcal{D}\sigma_1^- &= f_2^- \end{aligned}$$

and hence choose

$$\begin{aligned} \epsilon =& \frac{\mathcal{D}\sigma_1^+}{\mathcal{D}\sigma_1^+ - \mathcal{D}\sigma_1^-} \\ =& \frac{-\left(\frac{1}{Y_{X/S_1}} r_X X + \frac{1}{Y_{P/S_1}} r_P X\right) - \frac{S_1}{V}\dot{V} + \frac{1}{V}\left(F_A S_{F_A} + F_B S_{F_B}\right)}{-\left(\frac{1}{Y_{X/S_1}} r_X X + \frac{1}{Y_{P/S_1}} r_P X\right)}. \end{aligned} \tag{3.55}$$

In the right hand side of (6.13) we therefore replace the equation (6.13d) for the first substrate by

$$\begin{aligned} \dot{S}_1 =& -(1-\epsilon)\left\{\left(\frac{1}{Y_{X/S_1}} r_X X + \frac{1}{Y_{P/S_1}} r_P X\right) - \frac{S_1}{V}\dot{V} + \frac{1}{V}\left(F_A S_{F_A} + F_B S_{F_B}\right)\right\} \\ &+ \epsilon\left\{-\frac{S_1}{V}\dot{V} + \frac{1}{V}\left(F_A S_{F_A} + F_B S_{F_B}\right)\right\} \end{aligned}$$

which obviously vanishes for all times with (3.55). This represents the simplest possible case where the convex combination leads to a vanishing time derivative as long as $\epsilon \notin \{0,1\}$ or equivalently as long as $\mathcal{D}\sigma_1^+ < 0 \wedge \mathcal{D}\sigma_1^- > 0$.

For a priori determined feed rates in the simulatory context we obtain a vanishing concentration of substrate 1, causing the derivative of the right hand side of the differential equation to behave in the above described manner. The simulation result for the first substrate is shown in Figure 3.6. One can interpret the combination of $\mathcal{D}\sigma_1^+ < 0$ and $\mathcal{D}\sigma_1^- > 0$ as already mentioned in equation (3.52) to cause the additional algebraic constraint $\sigma_1 = 0$ to arise locally. A more detailed description of the model is given in 6.2.1.

Chapter 4

Derivative Generation

One crucial point for the efficiency of optimal control algorithms is the efficient generation of derivatives. For the solution of optimal control problems of the form (1.4) as presented in Chapter 1, we apply the derivative based SQP algorithm as explained in Section 2.2.

This chapter is organized as follows: In the first section the derivatives needed for the SQP algorithm are briefly specified. In Section 4.2 we give a compendious introduction to the theory of automatic differentiation. We subsequently explain forward and reverse mode and how they are applied in our implementation.
The third section deals with efficient techniques for the generation of sensitivity information.
The approximation of the Hessian matrix which is also needed for the SQP algorithm has already been presented in Subsection 2.2.1.
Since the repeated generation of the Jacobian of the right hand sides is one of the most CPU time consuming parts of the optimal control algorithm, we concentrate in the fourth section on the efficient generation of these Jacobians by exploiting the respective sparsity. Some implementory information is given.

4.1 Derivatives in the SQP algorithm

The SQP algorithm for solving the general optimal control problem of the form (1.4) in every step requires the solution of the QP (2.15), which requires beyond function and constraint evaluations the gradients of the objective function $\nabla_w F(w)$ and the gradients of the equality and inequality constraints $\nabla_w G(w_k)^T$ and $\nabla_w H(w_k)^T$.

4.2 Automatic Differentiation (AD)

In the first part of this section we motivate the usage of automatic differentiation. Therefore we briefly introduce into the generation of derivatives by finite differences and via symbolic differentiation.
Subsequently we briefly explain how automatic differentiation works. The second part of this section focuses on the implementation of automatic differentiation in the context of this thesis, including the presentation of several applications.

4.2.1 Theoretical background of AD

In principle four different approaches for the generation of derivative information are available

- Differentiating by hand
- Numerical differentiation
- Symbolic differentiation
- Automatic differentiation*

The differentiation by hand can only be reliably performed for simple function evaluations. This technique is error prone and time-consuming.
The most common way to generate numerical derivative information is the classical numerical differentiation. In most methods derivatives are approximated by one sided divided differences.
Let the vector $\mathfrak{w} \in \mathbb{R}^{n_\mathfrak{w}}$ and the direction $\delta_\mathfrak{w} \in \mathbb{R}^{n_\mathfrak{w}}$ be given. The derivative of $z(t;\tau_j,\mathfrak{w})$ w.r.t. the direction $\delta_\mathfrak{w}$ can then be approximated by using finite differences of the form

$$\frac{\partial z}{\partial \mathfrak{w}}(t;\tau_j,\mathfrak{w}) = \frac{z(t;\tau_j,\mathfrak{w}+h\delta_\mathfrak{w}) - z(t;\tau_j,\mathfrak{w})}{h} + \mathcal{O}(h). \tag{4.1}$$

Here $z(t;\tau_j,\mathfrak{w})$ can e.g. represent the solution of an initial value problem of the form (1.7) and $\mathfrak{w}$ can e.g. represent the vector of initial values and control parameters (2.1).

We need to keep in mind that the accuracy of the divided differences strongly depends on the choice of h:

*Some authors speak of *algorithmic* or *computational differentiation* instead of *automatic differentiation*.

- If h is chosen small, the cancellation error reduces the number of significant digits.
- If h is chosen too large, the terms $\mathcal{O}(h^2)$ for the unsymmetric divided difference become significant.

Even if h is optimally chosen one looses almost half of the significant digits in every step for general functions.

Symbolic differentiation techniques are implemented in computer algebra packages like Mathematica [WR03], Maple [Map04], Derive [Ins04], etc.. Character strings describing the function itself are used to generate character strings describing the derivatives. This can lead to a consumption of resources making the technique hardly efficiently applicable for numerical practice, since the arising expressions may become complicated.
A quite famous example is the calculation of the gradient of Speelpenning's function [Spe80]

$$f(x) = \prod_{i=1}^{n} x_i.$$

The corresponding gradient has the symbolic form

$$\nabla f(x) = \begin{pmatrix} \prod\limits_{j=2}^{n} & x_j \\ \prod\limits_{j=1,j\neq 2}^{n} & x_j \\ \vdots & \\ \prod\limits_{j=1}^{n-1} & x_j \end{pmatrix}. \tag{4.2}$$

Symbolic differentiation does not incur truncation errors.

Taking a closer look at (4.2) one can easily see that there are a lot of common subexpressions, which can be used repeatedly for evaluating the gradient. AD exploits this directly by treating these repeatedly occurring expressions as intermediate values or functions but does not look for an explicit expression!
Like symbolic differentiation, automatic differentiation operates by systematic application of the chain rule but instead of applying it to symbolic expressions it is applied to numerical values. In the following we briefly demonstrate the basic automatic differentiation approach with focus on the requirements arising in the context of this thesis.

We want to differentiate a vector valued function

$$F : \mathfrak{R} \subset \mathbb{R}^n \mapsto \mathbb{R}^m.$$

mapping the *independent variables* x to the *dependent variables* $y = F(x)$ and require the Jacobian

$$F' : \mathfrak{R} \subset \mathbb{R}^n \mapsto \mathbb{R}^{m\times n}$$

to be a well-defined matrix valued function on $\mathfrak{R}$. Let the function F be factorizable, i.e., the algorithm to calculate F consists of a finite (possibly long) sequence elemental operations (like summation, multiplication, etc.) and elemental function calls (like sin, exp, etc.) which are sufficiently smooth.
The algorithm for calculating F is a composition of the elemental operations φ_i. Thus all quantities v_i calculated during the evaluation of a function at a particular argument can be numbered

$$[\underbrace{v_{1-n}, \ldots, v_0}_{x}, v_1, \ldots, v_{l-m}, \underbrace{v_{l-m+1}, \ldots, v_l}_{y}].$$

The values $v_{1-n}, \ldots, v_0$ are copies of the values of the independent variables, whereas the v_i, $1 \le i \le l-m$, are obtained by applying the elemental functions φ_i to some of the arguments v_j with $j < i$

$$v_i = \varphi_i(v_{j\prec i}). \tag{4.3}$$

The relation $i \prec j$ means that v_i directly depends on v_j[†]. The values $v_{l-m+1}, \ldots, v_l$ contain the values of the dependent variables.
The elemental operations can be differentiated. Repeatedly applying the chain rule directly leads to the truncation error free derivative of F. The evaluation of the derivative is closely related to the evaluation of the function itself. We distinguish between the two different strategies of *forward* and *reverse mode.*

In the **forward mode** derivatives of the intermediate results with respect to the independent variables are calculated. This mode is typically used for the generation of directional derivatives

$$\bar{y} = F'\bar{x} = \bar{F}(x, \bar{x}),$$

where $\bar{x} \in \mathbb{R}^n$ contains the following directions

$$\begin{array}{lclrcll} v_{i-n} & \equiv & x_i, & \bar{v}_{i-n} & \equiv & \bar{x}_i, & 1 \le i \le n \\ v_i & \equiv & \varphi_i(v_{j\prec i}), & \bar{v}_i & \equiv & \sum_{j\prec i} \frac{\partial \varphi_i}{\partial v_j}\bar{v}_j, & 1 \le i \le l \\ y_{m-i} & := & v_{l-i}, & \bar{y}_{m-i} & := & \bar{v}_{l-i}, & i = m-1, \ldots, 0 \end{array} \tag{4.4}$$

[†]In general $i \prec i$ can also mean iterative evaluation of elemental functions.

and $\bar{F}(x, \bar{x}) : \mathbb{R}^{2n} \mapsto \mathbb{R}^m$.
Before the elemental partial derivatives $\frac{\partial \varphi_i}{\partial v_j}$ enter into any multiplication they can be evaluated to real numbers at the current point. This is a key distinction from the general symbolic derivative treatment. With the algebraic representation of the gradient of Speelpenning's function (4.2) in mind one directly can see the profit.

Evaluating the derivatives by propagating them using forward mode through the evaluation procedure will produce costs growing linearly with the number of directions $\bar{x}$.

Crucial for a priori run-time estimations for the derivative generation is a complexity estimation of the floating point operations required for the evaluation of directional derivatives using automatic differentiation.
One can show that for (4.4) we obtain

$$OPS(F'(x)\bar{x}) \leq const \cdot OPS(F(x)) \tag{4.5}$$

with a constant $const \in [3, 5]$ and a complexity measure OPS[Gri00]. The lower bound for the worst case estimate holds for the classical measure ignoring memory access costs and costs for additions, and focusing on multiplication costs instead. The upper bound explicitely takes the former costs into account. For practical applications automatic differentiation often outperforms the usage of divided differences (see e.g. [BCKM96], [HBSC97]).

The **reverse mode** propagates derivatives of the evaluation result y with respect to intermediate quantities reverse to the original evaluation procedure. It can be regarded as a reverse application of the chain rule.

$$\bar{x}^* \equiv \bar{y}^* F' = \bar{F}^*(x, \bar{y}^*)$$

is evaluated as a function of x and $\bar{y}^*$ ($\bar{F}^* : \mathbb{R}^{n+m} \mapsto \mathbb{R}^n$).
In order to be able to propagate values reversely through the evaluation process we first require a forward evaluation run

$$\begin{array}{lll} v_{i-n} & \equiv & x_i & \quad 1 \leq i \leq n \\ v_i & \equiv & \varphi_i(v_{j \prec i}) & \quad 1 \leq i \leq l \\ y_{m-i} & := & v_{l-1}, & \quad i = m-1, \ldots, 0 \end{array}$$

in which we save the intermediate results. In the reverse run, the derivatives are calculated

$$\begin{array}{lcll} \bar{v}^*_{l-i} & \equiv & \bar{y}^*_{m-i} & 0 \leq i \leq m-1 \\ \bar{v}^*_i & \equiv & 0 & 1-n \leq i \leq l-m \\ \bar{v}^*_j & \equiv & \bar{v}^*_j + \bar{v}^*_i \frac{\partial \varphi_i}{\partial v_j} & \forall j \prec i, \ , l \geq i \geq 1 \\ \bar{x}^*_i & \equiv & \bar{v}^*_{i-n} & n \geq i \geq 1. \end{array}$$

The costs for the derivative generation by reverse mode grow linearly with the number of independent variables.

For symmetry reasons the complexity of the reverse propagation of gradients

$$OPS(\bar{y}^* F'(x)) \leq const^r \cdot OPS(F(x)) \tag{4.6}$$

is also bounded above by a constant $const^r \in [3, 5]$.

Geometrically spoken the function $\bar{F}$ maps the tangent $\bar{x}$ to the tangent $\bar{y}$, whereas the function $\hat{F}^*$ maps the normal vector $\bar{x}^*$ to the normal vector $\bar{y}^*$. The fundamental relation

$$\bar{x}^* \bar{x} = [\bar{y}^* F'] \bar{x} = \bar{y}^* [F' \bar{x}] = \bar{y}^* \bar{y}$$

shows the symmetry between the two strategies. For a more detailed description, refer to [Gri00].

Two fundamentally different implementations of automatic differentiation algorithms are widely used. One is based on source-code transformation [TPS00], the other one on operator overloading techniques. In general the forward mode is used in the context of source code transformation. Due to linguistical complexity, C/C++ models can until today not be derived using source code transformation techniques. Source code transformation codes like ADIFOR ([BCKM94], [BCC+92]) usually restrict to the forward mode[‡].
Operator overloading strategies can be applied efficiently to models of almost every language whereas the automatic differentiation software itself has to build on a language allowing operator overloading. A famous example is ADOL-C [GJM+99], which we apply in our work.

A complete overview of the available tools can be found at [Bis04].

[‡]An exception is the commercial code TAF.

4.2.1.1 Advantages of AD for ill-conditioned systems

Due to the stiffness of the systems we require implicit integrators. A linear equation $\mathfrak{J}x = b$, which will be presented in the next section, has to be solved. The *condition number* of the model Jacobian $\kappa(\mathfrak{J}) = \|\mathfrak{J}\|\|\mathfrak{J}^{-1}\|$ of a specific model discriminates well-conditioned from ill-conditioned problems. For invertible $\mathfrak{J}$ and $b \neq 0$ we obtain with the perturbation $\tilde{\mathfrak{J}}\tilde{x} = \tilde{b}$ of the linear system $\mathfrak{J}x = b$ the error bound

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \kappa(\mathfrak{J}) \left(\frac{\|\mathfrak{J} - \tilde{\mathfrak{J}}\|}{\|\mathfrak{J}\|} + \frac{\|\tilde{b} - b\|}{\|b\|} \right) \frac{1}{1 - \kappa(\mathfrak{J}) \frac{\|\tilde{\mathfrak{J}} - \mathfrak{J}\|}{\|\mathfrak{J}\|}}. \tag{4.7}$$

Thus the relative error due to a small perturbation of relative size ϵ is of size $\mathcal{O}(\kappa(\mathfrak{J})\epsilon)$. If we calculate the Jacobian $\mathfrak{J}$ via finite differences we can expect loosing approximately half of the significant digits. For condition numbers of approximately $1/\sqrt{eps}$ the resulting error is of order one.

4.3 Generating sensitivity information

In this section we study the generation of the derivatives of the solution of the initial value problem $z(t; \tau_j, s_j^z, q, p)$ (1.7) with respect to initial values s_j^z, control parameters q and parameters p.

4.3.1 External numerical differentiation

Let $\mathfrak{w} = (s_j^z, q, p)$ consist of the initial values $s_j^z \in \mathbb{R}^{n_z}$ of the relaxed differential algebraic equation (1.7), the vector of control values $q \in \mathbb{R}^{n_u}$ and the parameter vector $p \in \mathbb{R}^{n_p}$.

The classical approach for obtaining derivative information of the solution $z(t; \tau_j, \mathfrak{w})$ with respect to initial values, control values and parameters is to calculate a nominal trajectory and in addition a trajectory with expediently perturbed initial values, control values and parameters. Adding a perturbation term $\delta_{\mathfrak{w}}$ to $\mathfrak{w}$ and calculating a perturbed trajectory allows us to compute an approximation to $\frac{\partial z}{\partial \mathfrak{w}}(t; \tau_j, \mathfrak{w})$ by the unsymmetric difference quotient (Equation 4.1). In general already small perturbations of the initial values lead to different grids and orders of the BDF-integrator using step size and order control. Gear and Vu ([GV83]) chose a BDF method of fixed step size and fixed order to obtain reliable derivative information. The approach to simply increase accuracy of nominal and varied trajectory leads to dramatically rising computation costs[§] especially for stiff systems. The problems become even more severe when treating problems with implicit discontinuities. For perturbed initial conditions (and possibly perturbed controls and parameters) additional switching point searches have to be performed.

This method for obtaining sensitivity information is called *external numerical differentiation (END)* because of the fact that the differentiation is performed outside the discretization scheme.

One can regard the discretization of the nominal trajectory as a sequence of maps. Perturbing initial values, parameters and control values will likely, as mentioned before, lead to different step sizes. This sequence of maps does not always differentiably depend on initial values, parameters and control values. The *internal numerical differentiation (IND)* avoids this effect.

4.3.2 Internal numerical differentiation

The *internal numerical differentiation* [Boc81] of the discretization scheme in combination with accurate derivatives of the model functions provides sufficiently precise sensitivity information. The central idea behind the internal numerical differentiation is to differentiate the adaptively generated sequence of maps but not the adaptive components like order and step size control or error control. This also includes Newton's method for solving the implicit corrector equation, like freezing the number of Newton steps for the solution of the implicit equation (3.23), approximation of the Jacobian and its decomposition. The reutilization of the linear algebra components makes the IND extraordinarily advantageous concerning computational effort.

The name *internal differentiation* is explained by the fact that the differentiation is carried out within the discretization scheme.

[§]As a rule of thumb one can assume that half of the significant digits of the accuracy of nominal and varied trajectory remain valid in the derivative approximation of the sensitivity.

For the variational DAE we obtain the system

$$
\begin{aligned}
A(z,u,p)\cdot \dot{\mathfrak{W}}^x &= \begin{pmatrix} -A_x\dot{x}+f_x \\ -A_y\dot{x}+f_y \\ -A_q\dot{x}+f_q \\ -A_p\dot{x}+f_p \end{pmatrix}^T \times \begin{pmatrix} \mathfrak{W}^z_{s_j^z} & \mathfrak{W}^z_q & \mathfrak{W}^z_p \\ & \mathbb{1} & \\ & & \mathbb{1} \end{pmatrix} \\
0 &= \begin{pmatrix} g_z \\ g_q \\ g_p \end{pmatrix}^T \times \begin{pmatrix} \mathfrak{W}^z_{s_j^z} & \mathfrak{W}^z_q & \mathfrak{W}^z_p \\ & \mathbb{1} & \\ & & \mathbb{1} \end{pmatrix} - \alpha(t) \begin{pmatrix} g_z(\tau_j) \\ g_q(\tau_j) \\ g_p(\tau_j) \end{pmatrix}^T,
\end{aligned}
\tag{4.8}
$$

where $\alpha(t)$ is the relaxation factor (3.25). In general we only require n_d directions which can be accumulated in a matrix $\mathfrak{D} = (\mathfrak{D}_z^T, \mathfrak{D}_q^T, \mathfrak{D}_p^T)^T \in \mathbb{R}^{(n_z+n_u+n_p)\times n_d}$. The Wronskian matrices $\mathfrak{W}^{\mathfrak{q}}_{\mathfrak{r}}$ are given by the matrices

$$
\mathfrak{W}^{\mathfrak{q}}_{\mathfrak{r}} = \frac{\partial \mathfrak{q}}{\partial \mathfrak{r}}(t;\tau_j)
$$

with $\mathfrak{q} \in \{x,z\}$ and $\mathfrak{r} \in \{s_j^z, q, p\}$. $\mathfrak{W}$ denotes the Wronskian $\left(\frac{\partial z}{\partial s_j^z, q, p}\right)$. With (4.8) we obtain the new variational DAE with only n_d directions

$$
\begin{aligned}
A(z,u,p)\cdot \mathfrak{D}\cdot \dot{\mathfrak{W}}^x &= \begin{pmatrix} -A_x\dot{x}+f_x \\ -A_y\dot{x}+f_y \\ -A_q\dot{z}+f_q \\ -A_p\dot{z}+f_p \end{pmatrix}^T \times \begin{pmatrix} \mathfrak{D}\cdot\mathfrak{W} \\ \mathfrak{D}_q \\ \mathfrak{D}_p \end{pmatrix} \\
0 &= \begin{pmatrix} g_z \\ g_q \\ g_p \end{pmatrix}^T \times \begin{pmatrix} \mathfrak{D}\cdot\mathfrak{W} \\ \mathfrak{D}_q \\ \mathfrak{D}_p \end{pmatrix} \\
&\quad -\alpha(t) \begin{pmatrix} g_z(\tau_j) \\ g_q(\tau_j) \\ g_p(\tau_j) \end{pmatrix}^T \times \begin{pmatrix} \mathfrak{D}\cdot\mathfrak{W} \\ \mathfrak{D}_q \\ \mathfrak{D}_p \end{pmatrix}.
\end{aligned}
\tag{4.9}
$$

At the presence of discontinuities the update matrices explained in Section 3.3 have to be taken into account.

IND is also stable for low integration accuracies [Boc87]. The nominal and the perturbed solution are very similar which makes it possible to reuse iteration matrices for the perturbed trajectory from the nominal one.
For the sensitivity calculation by usage of IND two fundamentally different approaches can be applied. One is to calculated the Wronskians via finite differences,

the other one is to directly solve the variational differential equation. In the first approach we solve an IVP (3.25) for the nominal trajectory and the varied trajectories with perturbation $\mathfrak{w}$. Because of the drawback that an appropriate choice of the perturbation to set up the differential quotient can hardly be given we do not use this method. For a detailed discussion of this approach refer to [Bau99].

In order to calculate the sensitivities, we directly treat the variational differential equation along with the nominal system when using automatic differentiation to calculate the sensitivity matrices. Bock [Boc83] also calls this approach the *analytical limit* of the IND.

To simplify presentation, we set $A = \mathbb{1}$ in what follows.

For the solution of the discretized equations for the variational DAEs (4.9) we have to solve the linear system

$$\begin{pmatrix} \alpha_0 \mathbb{1} + h\frac{\partial f}{\partial x} & h\frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} \cdot \left.\frac{\partial z}{\partial \mathfrak{w}}\right|_{n+1} = \begin{pmatrix} -h\frac{\partial f}{\partial q}\mathfrak{D}_q - h\frac{\partial f}{\partial p}\mathfrak{D}_p - \mathbb{1}c(\mathfrak{W}_{\mathfrak{w}}) \\ -\frac{\partial g}{\partial \mathfrak{w}} + \alpha(t_{n+1})\frac{\partial g}{\partial \mathfrak{w}}(\tau_j)\mathfrak{D} \end{pmatrix} \tag{4.10}$$

where $c(\mathfrak{W}_{\mathfrak{w}})$ represents the constant part of the derivative of the corrector polynomial, n is the active integrator step (see Chapter 3). For large scale systems as we treat them in the thesis on hand the efficient calculation of the derivative matrices is crucial. In the following section we develop techniques particularly adapted to large scale sparse systems as they usually arise in chemical engineering.

4.4 Exploiting sparsity structures

In practice the models describing application processes are mostly sparse. Consequently substantial savings can be expected by exploiting the model sparsity structure. In the first part we concentrate on compression techniques for forward mode, i.e., compression from the right side. We later extend these techniques to reverse mode.

When speaking about model functions, we refer to the right hand sides of the DAE as f and g. F stands for either the differential or the algebraic functions.

4.4.1 Compression techniques

As seen in the previous section the forward mode allows the computation of a product

$$\mathfrak{J} \cdot \mathfrak{S} \in \mathbb{R}^{m \times \iota} \tag{4.11}$$

in running time proportional to $\text{eval}(F) \cdot \iota$, where $\text{eval}(F)$ is the cost for evaluating the vector function F, m and ι are the numbers of rows and columns of the system

(4.11). The question is how to find a matrix $\mathfrak{S}$ for a given sparsity structure of the model Jacobians $\mathfrak{J}$, i.e., the derivative matrices in (4.10) in such a way that the resulting system (4.11) contains as few columns as possible. If we speak about model Jacobians in the following, we speak about these derivative matrices. The sparsity structure of the model is obtained by a bit pattern propagation.
Coleman and Moré in 1984 ([CM84]) proposed an approach for easy computation of the matrix $\mathfrak{S}$, Averick et al. in 1993 ([AMB93]) combined these ideas with automatic differentiation techniques.
In the following we will focus on a sparsity structure exploitation of the model Jacobians by compression techniques.
Let $\prec$ describe the dependency relation, i.e., $l \prec m$ means that v_m depends on v_l with the arguments v described in Subsection 4.2.1. Let $\prec^*$ denote the transitive closure of $\prec$ which is a partial ordering of all indices $i = 1-n, \ldots, l$. The index set $\mathcal{X}_k \equiv \{j \leq n : j-n \prec^* k\},\ k = 1-n, \ldots, l$ signifies which elemental relations v_k (see 4.3) depend nontrivially on the independent variable x_j. The *index domains* can be obtained by

$$\mathcal{X}_k = \bigcup_{j \prec k} \mathcal{X}_j \quad \text{for} \quad j \leq n, \tag{4.12}$$

starting from $\mathcal{X}_{j-n} = \{j\}$. Analogously the index ranges

$$\mathcal{Y} = \{i \leq m : k \prec^* (l-m+i)\} \quad \text{for} \quad k = l, \ldots, n-1$$

contain the indices of all dependent variables y_i non-trivially influenced by the intermediate v_k. They can be computed in complete analogy to Equation 4.12 according to their backward recurrence. We call

$$w = \max_{1-n \leq i \leq l} |\mathcal{X}_i|$$

the *width of the Jacobian matrix* $\mathfrak{J}$ and

$$\hat{w} = \max_{1-n \leq i \leq l} |\mathcal{Y}_j|$$

the width of $\mathfrak{J}^T$.
Choosing a so-called *seed matrix* $\mathfrak{S}$ consisting of n_s directions $\mathfrak{s} \in \mathbb{R}^\iota$, we obtain the matrix equation

$$\mathfrak{J}^{\text{comp}} = \mathfrak{J} \cdot \mathfrak{S} \in \mathbb{R}^{m \times \iota}. \tag{4.13}$$

The ith row of the compressed matrix $\mathfrak{J}^{\text{comp}}$ is given by

$$\mathfrak{j}_i^{\text{comp}^T} = \mathfrak{J}_i \cdot \mathfrak{S} \in \mathbb{R}^m \tag{4.14}$$

and thus

$$\mathfrak{j}_i^{\mathrm{comp}^T} = \mathfrak{j}_i^T \mathfrak{S}_i \tag{4.15}$$

with

$$\mathfrak{S}_i = (e_j^T \mathfrak{S})_{j \in \mathcal{X}_{l-m+1}} \in \mathbb{R}^{\iota_i \times \iota}. \tag{4.16}$$

$\mathfrak{j}_i^T$ is given by the gradient of the ith model function F_i. The entries of $\mathfrak{j}_i$ can be retrieved if the number of computed values ι is not smaller than the number $p\iota_i = |\mathcal{X}_{l-m+i}|,\ \ 1 \leq i \leq m$ of unknown non-zero entries in the ith row. Since we need all rows of $\mathfrak{J}^{\mathrm{comp}}$, we require the number of rows of the seed matrix $\mathfrak{S}$ to be greater than or equal to the matrix width of the Jacobian $\mathfrak{J}$:

$$p \geq \max_{1 \leq i \leq m} p_i = w.$$

The compression relation (4.13) can be illustrated as follows

$$m\left\{\underbrace{\begin{bmatrix} \\ \\ \mathfrak{J}_{comp} \\ \\ \times\quad\times\quad\times\quad\dots\quad\times \\ \\ \end{bmatrix}}_{p} \equiv \underbrace{\begin{bmatrix} \\ \\ \mathfrak{J} \\ \\ 0\times00\dots00\times\times0 \\ \\ \end{bmatrix}}_{n}\underbrace{\begin{bmatrix} \\ \\ \mathfrak{S} \\ \\ \times\times\times\dots\times \\ \\ \end{bmatrix}}_{p}\right\}n$$

where n denotes the number of independent variables. The sparsity structure of the ith row of the Jacobian $\mathfrak{J}$ is given by the index set $\mathcal{X}_{l-m+i}$.
Regarding the columns $\mathfrak{j}_i$ as vertices $V = \{\mathfrak{j}_1, \mathfrak{j}_2, \dots, \mathfrak{j}_n\}$ of the Jacobian $\mathfrak{J}$ and $(i,j) \in E$ as edges for $\mathcal{Y}_{i-n} \cap \mathcal{Y}_{j-n} \neq \emptyset$ we obtain the graph $\mathcal{G}(\mathfrak{J})$. The task of coloring the column incidence graph $\mathcal{G}(\mathfrak{J})$

$$\phi : V \mapsto \{1, 2, \dots, p\}$$

with the least possible number of colors such that adjacing edges do not have the same color is the famous graph coloring problem known to be NP-hard (see [Pot88], [BT00]). The minimum number of colors required for the coloring of $\mathcal{G}(\mathfrak{J})$ is called the *chromatic number* $\chi(\mathcal{G}(\mathfrak{J}))$. In short the coloring instances arise from a matrix partitioning problem [HS02].
Several heuristical approaches leading to approximations of the chromatic number - and thus heuristics for the choice of $\mathfrak{S}$ - have been proposed in the last decades.

In the following we will present two of them and adapt them to our requirements.

Crucial for the performance of these approaches is, aside from the remaining number of columns in the compressed Jacobian, also the computational expense for reobtaining the original Jacobian $\mathfrak{J}$. The methods for setting up the seed-matrix $\mathfrak{S}$ can be divided into groups given by the operations necessary to reobtain the original Jacobian. Powell and Toint [PT79] distinguished between

- direct
- substitution
- elimination

methods for approximating Jacobians. In direct methods the m $\iota_i \times \iota_i$ square matrices $\mathfrak{S}_i^{\square}$ (first ι_i rows of $\mathfrak{S}_i$) are permutation matrices ($\mathbb{1}^{p_i \times p_i}$). Substitution methods allow $\mathfrak{S}_i^{\square}$ to be a unitary $0-1$-matrix, in case of elimination methods $\mathfrak{S}_i^{\square}$ is a general non-singular $\iota_i \times \iota_i$ square matrix. These classes inherently differ in the algorithms needed for the reconstruction of the full Jacobian. For direct methods no arithmetic operations are necessary, substitution methods possibly need subtractions, elimination methods in general need all arithmetic operations.

Curtis-Powell-Reid seeding (CPR)

One very famous heuristical approach we implemented in the thesis at hand is the so called Curtis-Powell-Reid strategy [CPR74] with

$$\mathfrak{S} = \left[e_{\phi(j)}^T\right]_{j=1,\dots,n} \in \mathbb{R}^{n\times\iota}$$

and Cartesian basic vectors $e_i \in \mathbb{R}^p$. The submatrices

$$\left[e_{\phi(j)}^T\right]_{j\in\mathcal{X}_{l-m+i}} \in \mathbb{R}^{\iota_i\times\iota} \tag{4.17}$$

of $\mathfrak{S}$ build up the seed matrix $\mathfrak{S}$ itself. To sum up:

- Each column contains no more than one 1.
- Each row contains exactly one 1.

$\mathfrak{J}^{\text{comp}}$ can be interpreted as a sparse data structure of the original Jacobian $\mathfrak{J}$. No arithmetic operations are needed in this direct method for approximating the Jacobian $\mathfrak{J}$, when building it from the compressed form $\mathfrak{J}^{\text{comp}}$.

Obviously a lower bound for the number of colors in the column incidence graph is the maximal number of non-zero entries per row. For left hand side compression of the Jacobian $\mathfrak{J}$ in combination with the reverse mode, the lower bound is the maximal number of non-zero entries per column.
The chromatic number $\chi(\mathcal{G}(\mathfrak{J}))$ is of course independent of the ordering of the columns but the coloring and thus the approximation obtained by a sequential algorithm to the chromatic number does depend on the ordering! The resulting width w of the matrix is bounded below by the chromatic number $\chi(\mathcal{G}(\mathfrak{J}))$ of the column incidence graph.

Newsam-Ramsdell seeding (NR)

Another approach to efficiently compute sparse Jacobians is the method of Newsam-Ramsdell [NR83].
Often the algorithm proposed by Newsam and Ramsdell provides a more efficient way to calculate the sparse Jacobian matrix. For at most p non-zero elements per row of the Jacobian $\mathfrak{J}$ we construct a seed matrix $\mathfrak{S}^{NR} \in \mathbb{R}^{n\times p}$ such that $\mathfrak{J}$ can be reconstructed from

$$\mathfrak{J}^{\text{comp}} = \mathfrak{J} \cdot \mathfrak{S}^{\text{NR}}, \;\; \mathfrak{J}^{\text{comp}} \in \mathbb{R}^{m\times p}$$

by row-wise solution of

$$\mathfrak{j}_i^{\text{comp}} = \mathfrak{j}_i \cdot \mathfrak{S}^{\text{NR}} \in \mathbb{R}^{1\times p}, \;\; i = 1, \ldots, m.$$

The seed matrix $\mathfrak{S}^{\text{NR}}$ is chosen to be composed of Vandermonde matrices, which guarantees a full rank for every $p_i \times p_i$ submatrix

$$\mathfrak{S}_i^{\text{NR}^\square} = \left[\lambda_j^{k-1}\right]_{j\in\mathcal{X}_{l-m+1}}^{k=1,\ldots,p_i} \in \mathbb{R}^{p_i\times p_i}.$$

The Vandermonde matrix choice enables us to solve the arising linear systems in $\mathcal{O}(p^2)$ floating point operations [BP70] or [GL96] compared to the costs for the classical LU decomposition being of the order $\mathcal{O}(p^3)$. For a more detailed discussion of the algorithm see [GUG96].
Unfortunately the condition number increases exponentially with the dimension p of the Vandermonde matrix. This drawback is less severe for automatic differentiation than for derivatives provided by finite differences, but makes it sometimes favorable to prefer the Curtis-Powell-Reid method to Newsam-Ramsdell's algorithm. The possibility of circumventing this drawback by constructing the seed matrix based on Chebychev polynomials rises the floating point operation costs for solving the linear systems to $\mathcal{O}(p^3)$.

Until now we restricted our attention to a compression of the model Jacobian in the context of the automatic differentiation forward mode. Nevertheless, a compression of the original matrix multiplying from the left side in combination with the adjoint mode of AD may additionally reduce the actual number of derivative evaluations. Considering the submatrix of the distillation column model given in Section 6.1 (and in some detail in the Appendix A) we see that we obtain a substructure for the Jacobian concerning the molar fractions of the liquid phase of the structure

$$\mathfrak{J}_{\text{subsys}} = \begin{pmatrix} * & * & * & * \\ * & & & \\ & * & & \\ & & * & \\ & & & * \end{pmatrix} \in \mathbb{R}^{4\times 5}, \tag{4.18}$$

where the first row represents the molar balance on the specific tray and the other rows give the equilibrium conditions with the vapor phase of the individual component. Neither of the two previously explained compression strategies would compress the sub-Jacobian using a seed-matrix from the right side in forward AD mode. The costs for evaluating the derivative of the submatrix would be proportional to the product of the matrix' column width and the costs for evaluating the model function (4.5). Since $w \leq \chi(\mathfrak{G}(\mathfrak{J}))$, the Newsam-Ramsdell algorithm in general produces a smaller width of $\mathfrak{J}^{\text{comp}}$.
Nevertheless, we observed that for our applications with significant high condition numbers of the Jacobians themselves, the Curtis-Powell-Reid algorithm performs better, despite the fact that the Newsam-Ramsdell algorithm guarantees the width of $\mathfrak{J}^{\text{comp}}$ to be equal to w.
This especially becomes valid for row and column compression, discussed in the following.

Since we do not restrict ourselves to forward mode of AD it is by no means necessary to restrict matrix compression techniques to column compressions. The matrix $\mathfrak{J}_{\text{subsys}}$ (4.18) cannot be compressed by any of the presented compression techniques in combination with forward mode of AD, i.e., with the above presented techniques it is impossible to compute a derivative matrix $\mathfrak{J}^{\text{comp}}_{\text{subsys}} = \mathfrak{J}_{\text{subsys}} \cdot \mathfrak{S}_{\text{subsys}}$ with fewer columns. Finding a seed-matrix $\mathfrak{S}^{\text{rev}}$ to compress the original Jacobian from below reduces the derivative evaluation costs from $m \times \text{eval}(F)$ to possibly $\chi(\mathcal{G}(\mathfrak{J}^T)) \times \text{eval}(F)$. In the case of the matrix $\mathfrak{J}_{\text{subsys}}$ (4.18) derivative evaluation costs reduce from $5 \times \text{eval}(F)$ to $2 \times \text{eval}(F)$. In many situations it is advantageous to split matrices into submatrices since matrices of the form

$$\mathfrak{J}_{\text{subsys}} = \begin{pmatrix} * & * & * & * & * \\ * & * & & & \\ * & & * & & \\ * & & & * & \\ * & & & & * \end{pmatrix}$$

can not directly be compressed using the compression techniques of the antecedent section. Based on the ideas of Coleman and Verma [CV98] and [HS02] we apply a bipartitioning strategy on the CPU time intensive generation of the derivative matrices, with a priori giving a column respectively row splitting.

4.4.2 Conserving sparsity pattern information

In the following we focus on the use of automatic differentiation in the context of the thesis on hand.

On every subdomain $\mathfrak{R}_i$ (see Chapter 5) uniquely given by the actual switching vector σ the sparsity pattern $\mathcal{X}(\sigma_i)$ remains unchanged. Traversing one area boundary may change the sparsity pattern, nevertheless, in *real-world applications* mostly only few rows of the Jacobian change. In view of the exploitation of sparsity structures we therefore a priori divide the model equations into those which remain unchanged after potential changing in the signs of the switching vectors, and those that may change:

$$\mathfrak{J} = \left(\begin{array}{ccccccc} * & \dots & * & \dots & * & \dots & * \\ \vdots & & \vdots & & \vdots & & \vdots \\ * & \dots & * & \dots & * & \dots & * \\ \hline \triangle & \dots & \triangle & \dots & \triangle & \dots & \triangle \\ \hline * & \dots & * & \dots & * & \dots & * \\ \hline \triangle & \dots & \triangle & \dots & \triangle & \dots & \triangle \\ \hline * & \dots & * & \dots & * & \dots & * \\ \vdots & & \vdots & & \vdots & & \vdots \\ * & \dots & * & \dots & * & \dots & * \end{array}\right). \tag{4.19}$$

The triangles signify the columns with changing sparsity structure due to transition to other areas in space (see 5).

We apply a hierarchical storing and exploitation strategy:

1. Calculation of the bit patterns $\mathcal{X}^i$ for the area i (see (5.5)) for the Jacobian submatrices in this area, i.e., derivatives of differential and algebraic equations with respect to differential and algebraic variables, controls/parameters (see Equation 4.10).

2. After transition of an area boundary the new patterns $\mathcal{X}^j$ for the area j are obtained by calculation of the new columns arising from the implicit model change.

4.4.3 AD Application

For the differential algebraic equation (1.1) we obtain the model Jacobians needed to solve the linear system (4.10).

In the complex distillation process application, an intermediate tray (neither reboiler nor condenser) is described dynamically by usage of 5 differential and 81 algebraic variables and equations. Since to $n_y \gg n_x$ the effective computation of

$$\frac{\partial g}{\partial y} \in \mathbb{R}^{n_y \times n_y}$$

turns out to be crucial. To calculate the submatrix $\frac{\partial g}{\partial x} \in \mathbb{R}^{n_y \times n_x}$ we apply forward AD in combination with CPR respectively NR compression ($\frac{\partial g}{\partial x}^{\text{comp}} = \mathfrak{J} \cdot \mathfrak{S}$), for the submatrix $\frac{\partial f}{\partial y} \in \mathbb{R}^{n_x \times n_y}$ reverse AD to calculate the derivatives $\frac{\partial f}{\partial y}^{\text{comp}} = \mathfrak{S}^{\text{rev}} \cdot \mathfrak{J}$, with $\mathfrak{S}^{\text{rev}}$ obtained via CPR and NR, respectively. Since the number of control functions n_u and parameters n_p is small at least with respect to n_y (but also w.r.t. n_x if we focus on the complete column), the pairs $\left(\frac{\partial(f,g)}{\partial u}, \mathfrak{S}_u\right)$ and $\left(\frac{\partial(f,g)}{\partial p}, \mathfrak{S}_p\right)$ are computed using forward AD with $\mathfrak{S}_u \in \mathbb{R}^{n_u \times n_z}$ and $\mathfrak{S}_p \in \mathbb{R}^{n_u \times n_z}$.

4.4.4 Exploitation of model characteristics

Rigorous models of distillation processes are composed of comparatatively precise descriptions of the individual trays via e.g. thermodynamical models. In our case every individual tray (see Appendix A) is built up by a complex model, nevertheless the trays are interacting only via out- and in-streams, making the blocks loosely coupled. Discarding the switches in the trays would except for reboiler and condenser give a common sparsity pattern for every individual tray. For the Jacobian $\mathfrak{J} = \frac{\partial g}{\partial y}$

we obtain the following structure

$$\left(\begin{array}{cccccccccccccccccc}
\ulcorner & \cdots & \cdots & \urcorner & 0 & \blacktriangle & 0 & \cdots & & & & & & & & & \cdots & 0 \\
\vdots & & & \vdots & 0 & \cdots & & & & & & & & & & & \cdots & 0 \\
\vdots & & & \vdots & 0 & \cdots & & & & & & & & & & & \cdots & 0 \\
\llcorner & \cdots & \cdots & \lrcorner & 0 & \cdots & & & & & & & & & & & \cdots & 0 \\
0 & 0 & \boxtimes & 0 & \ulcorner & \cdots & \cdots & \urcorner & 0 & \blacktriangle & 0 & \cdots & & & & & \cdots & 0 \\
 & & & & \vdots & & & \vdots & 0 & \cdots & & & & & & & \cdots & 0 \\
 & & & & \vdots & & & \vdots & 0 & \cdots & & & & & & & \cdots & 0 \\
0 & \cdots & \cdots & 0 & \llcorner & \cdots & \cdots & \lrcorner & 0 & \cdots & & & & & & & \cdots & 0 \\
 & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & \\
0 & \cdots & & & & & & & & & \cdots & 0 & \boxtimes & 0 & \ulcorner & \cdots & \cdots & \urcorner \\
0 & \cdots & & & & & & & & & & & \cdots & 0 & \vdots & & & \vdots \\
0 & \cdots & & & & & & & & & & & \cdots & 0 & \vdots & & & \vdots \\
0 & \cdots & & & & & & & & & & & \cdots & 0 & \llcorner & \cdots & \cdots & \lrcorner
\end{array}\right). \tag{4.20}$$

The zeros in matrix (4.20) represent vanishing submatrices. $\boxtimes$ and $\blacktriangle$ give the coupling between the individual trays in the algebraic equations. In our context they only consist of liquid and vapor outflow. The blocks signify tray specific patterns which are still sparse themselves.

Since for implicit discontinuities the sparsity patterns may change (see 4.19) it is not directly possible to restrict the pattern determination to one tray, augment by the interconnecting patterns, and reuse the pattern for the rest of the trays. Nevertheless in practice only few equations change. This allows for an a priori bit pattern determination of those parts which remain unchanged in case of zero crossings of switching functions. The varying parts are calculated after the first evaluation of the switching vector and so determine the valid area $\mathfrak{R}$ (see Chapter 5).

Chapter 5

Solution strategy for the implicitly discontinuous DAE constrained optimization problem

In the last years much effort has been spent to cope with implicit discontinuous dynamical models in the context of optimal control of mechanical and mechatronical systems (see e.g. [MBSL03]). In so called hybrid models [EFS02] in addition to continuous variables discrete variables occur. At time points where the discrete variables change their value, switching events may occur. In contrast to switching points in our models, these points are either explicitely given or their chronology and number can a priori be uniquely determined and thus can be taken into account by combining discrete and continuous optimization methods.

In the following we present an algorithm we developed to find local optima of optimization problems with underlying differential algebraic equations (DAE) with implicitly defined discontinuities. Our strategy builds on the multiple shooting discretization of the dynamic model ([Pli81] and [BP84]). With an adequate control parameterization this leads to the highly structured NLP which can be solved by and adapted SQP method (see Chapter 2).

As shown in Chapter 3 switching events in the right hand sides of the DAE can be hidden in the integrator as long as they neither vanish nor change their chronology (see Theorem 9) due to changing control profiles or changing initial values for the differential equation in consecutive SQP steps. For comparatatively few discontinuities it is also possible to introduce additional switching phases. Since the size of the quadratic program (QP) is significantly increased by this approach it can not be applied in processes with large numbers of switches. Nevertheless this approach allows of the usage of integrators which can not treat state dependent discontinuities. We compare local optima on digraphs $\mathfrak{C}$ given by the switching chronology of the active trajectory.

This chapter is organized as follows: In the first section we recall the formulation of the optimization problem with implicitly defined discontinuities; the second section describes our trajectory monitor surveying the area chronology which we will interpret as a digraph in the fourth section. The third section explains our strategy for updating matching conditions if the ending point of a trajectory in one multiple shooting interval and the initial values of the trajectory of the succeeding interval do not have the same sign structure of the switching vector and thus do not lie in the same area $\mathfrak{R}$. The fourth section explains our strategy for controlling trajectories with differing digraphs $\mathfrak{C}$.

5.1 Problem formulation

The problem already introduced in Chapter 1 is of the form

$$\min_{t_f,z,u,p} \Xi(t_f, z(t_f), p) \tag{5.1a}$$

subject to

$$\begin{aligned}
A(t,z(t),u(t),p)\cdot\dot{x}(t) &= f(t,z(t),u(t),p,\operatorname{sgn}(\sigma(t))), && t\in[t_i,t_f] && (5.1b)\\
0 &= g(t,z(t),u(t),p,\operatorname{sgn}(\sigma(t))), && t\in[t_i,t_f] && (5.1c)\\
z(t_s^+) &= z(t_s^-)+\hat{\Delta}^z(t_s,z(t_s),p) && && (5.1d)\\
0 &\le h(t,z(t),u(t),p), && t\in[t_i,t_f] && (5.1e)\\
0 &= r(z(t_f),p) && && (5.1f)\\
0 &\le r(z(t_f),p) && && (5.1g)
\end{aligned}$$

where several switching points t_s may appear in $[t_i, t_f]$.

5.2 Monitoring the accessed area

In Figure 5.1 we schematically show a two dimensional cut through the phase space of a discontinuous dynamical model (5.1) fragmented into areas $\mathfrak{R}_m$ where the right

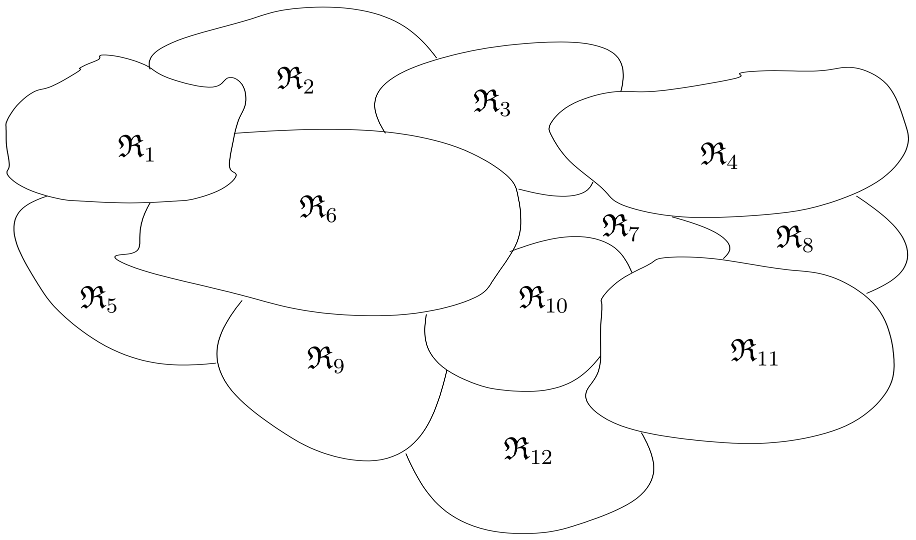

Figure 5.1: Two-dimensional schematic cut through the phase space. The $\mathfrak{R}_m$ symbolize the areas characterized by a unique sign structure of the switching vector $\sigma(t)$ (the area identification is given by the global monitor $\mathfrak{M}_{global}$).

hand sides of the DAE are sufficiently smooth. Each area $\mathfrak{R}_m$ is uniquely defined by the sign structure of the switching vector $\operatorname{sgn}\sigma(t)$.

Since by using multiple shooting the integration on the individual multiple shooting intervals is a priori decoupled we apply a two level discontinuity monitor. We introduce

- local monitors $\mathfrak{M}^{\hat{n}}_{local}, \ \ \hat{n} = 1, \ldots, n_{shoot}$,

- a global monitor $\mathfrak{M}_{global}$,

where n_{shoot} is the number of multiple shooting intervals.

The local monitors $\mathfrak{M}^{\hat{n}}_{local}$ assign a temporary area descriptor $\hat{\mathfrak{R}}_{\hat{n}}$ to the individual sign structures of the switching vector $\sigma(t)$, which occur while integrating in the multiple shooting interval $\hat{n}$.
After integration of each of the n_{shoot} intervals, the global monitor $\mathfrak{M}_{global}$ synchronizes the area numbering and stores the trajectory information as explained in Chapter 3.

Sparsity patterns $\mathcal{X}$ of the Jacobians $\mathcal{J}$ (see Section 4.4) in general vary from area $\mathfrak{R}_m$ to another area $\mathfrak{R}_k, k \neq m$. Hence, we apply the following strategy

- Preparation phase: Distinguish between invariant and variant model parts with respect to bit pattern changes of the Jacobians of the right hand sides of the DAE.
- Initialization phase: Initialize the global monitor with the sign structure of the switching vector $\sigma(t)$ and the sparsity structure based on the valid bit pattern and calculation of the seeding matrix $\mathcal{S}$.
- Integration phase: Perform the integration in the individual multiple shooting intervals and update the local monitor $\mathfrak{M}^{\hat{n}}_{local}$.
- Post-integration phase: Synchronize area information.

Remark 5.2. The sufficiently accurate detection of switching points requires a multi-level detection scheme (see section 3.3.2).

Freezing the adaptive components instead of the application of varied trajectories is strongly advisable since no additional zero searches of the switching vector $\sigma(t)$ are needed. The usage of Bock's IND (see section 4.3.2) directly builds on this freezing strategy to obtain sensitivity information.

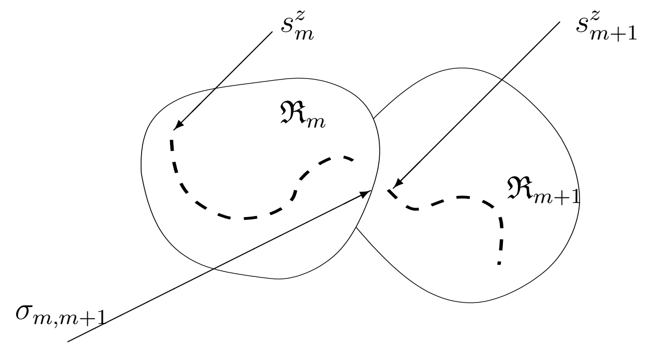

Figure 5.2: Schematical representation of the situation explained in Example 5.3. The end point of the trajectory in the first multiple shooting interval ends in a different area than the starting point of the following multiple shooting interval lies in.

5.2.1 Switching points at multiple shooting points

As long as discontinuities only appear within multiple shooting intervals and neither the chronology nor the number of accessed areas changes, the discontinuity events can in principle be hidden in the integrator which makes them invisible for the optimization algorithm. For the above mentioned efficiency reasons, e.g. with respect to reoccuring sparsity patterns of the Jacobians, this internal handling (see e.g. [Sch99b]) can not be recommended. Moreover discontinuity events may be - a priori unknown[*] - situated on multiple shooting nodes.

Example 5.3. : In the context of the distillation column application, let the column (see Section 6.1.1 or Appendix A) initially be floated with inert gas in an overheated state. Assume that due to reduced feed stream by the optimization the state remains overheated when the end of the multiple shooting interval $\hat{n} = 0$ is reached. This leads to

$$\psi_{\hat{n}=0}^{therm} > 0, \quad t \in [t_i, t_1].$$

Nevertheless let the new start value s_1 of the following multiple shooting interval contain a configuration with

$$\psi_{\hat{n}=1}^{therm} < 0, \quad t = t_1.$$

The situation is schematically shown in Figure 5.2. Let this implicit discontinuity be described by the component $\mathfrak{j}$ of the switching function vector. In this case the zero-crossing of the switching function $\sigma_{\mathfrak{j}} = \psi^{therm}$ describing this event occurs at the multiple shooting node since

$$\begin{aligned} \sigma_{\mathfrak{j}}(t_1, z(t_1), p)^- &> 0 \\ \sigma_{\mathfrak{j}}(t_1, z(t_1), p)^+ &< 0. \end{aligned}$$

Two fundamentally different cases for switching events which appear on multiple shooting nodes (between interval $\hat{n}$ and interval $\hat{n}+1$) have to distinguished:

In the first case the switching event lies on the end point of the interval $\hat{n}$ or on the start point of the subsequent interval $\hat{n}+1$. Theorem 9 is not valid in this case since the assumption that t_s lies in an interval $t_s \in (t_i^{\hat{n}}, t_f^{\hat{n}})$ is violated ($t_s = t_i^{\hat{n}+1} \notin (t_i^{\hat{n}+1}, t_f^{\hat{n}+1})$ or $t_s = t_f^{\hat{n}} \notin (t_i^{\hat{n}}, t_f^{\hat{n}})$). This is indicated by a zero-crossing of a switching function σ_j at the end point $t = t_s$. In the integrator we always calculate the trajectory until we reach a final point $\hat{t} > t_f^{\hat{n}}$. Thus Bock's Theorem 9 becomes valid for $t_s = t_f^{\hat{n}}$ and guarantees differentiability of the solution with respect to initial values and control parameters. For $t_s = t_i^{\hat{n}+1}$ one could in principle reverse integrate for a short time step. Instead we assume that the surrounding interval

[*]A priori knowledge of the time point of a switching point allows e.g. to introduce a stage transition phase of zero time length and therefore to adapt the consistency conditions.

exists to prove differentiability. Since in the final SQP step consistency is obtained differentiability can be guaranteed in the solution of the algorithm.

In the second case the switching event has to take place at the multiple shooting node since start point of interval $\hat{n}+1$, i.e., point $t_i^{\hat{n}+1}$, and end point $t_f^{\hat{n}}$ of interval $\hat{n}$ do not lie in the same areas but treated independently neither at $t_f^{\hat{n}}$ nor at $t_i^{\hat{n}+1}$ any of the switching function in the switching vector shows zero-crossing. In this case the matching condition at the multiple shooting nodes have to be modified as described in the next section.

5.3 Updating consistency conditions

In this section we present our strategy for updating matching conditions if the sign structure of the switching vector $\text{sgn}(\sigma) \in \{-1, 1\}^{n_\sigma}$ at the state vector at the end of multiple shooting interval $\hat{n}$ deviates from the sign structure of the switching vector $\text{sgn}(\sigma)$ at the state vector at the start point of the succeeding interval $\hat{n}+1$.

To simplify presentation we assume for the moment that the discontinuities due to changes of controls and initial values change multiple shooting intervals but preserve their chronology and number. Nevertheless, this restriction is by no means necessary for our approach.

As pointed out in the preliminary example (Example 5.3), zero-crossings in the switching vector may be situated on multiple shooting nodes.

In Chapter 3 we gave the embedding theorem (Theorem 9). Under little restrictive assumptions it guarantees the differentiability of the solution of the IVP

$$\mathcal{Q} \cdot \dot{z} = \begin{cases} (f,g)^-(t,z,u,p) & t \in [t_i, t_s] \\ (f,g)^+(t,z,u,p) & t \in (t_s, t_f] \end{cases}$$

with

$$\mathcal{Q} = \begin{pmatrix} \mathbb{1}_{n_x} & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{(n_x+n_y)\times(n_x+n_y)}$$

and

$$x(t_i) = s_i^x$$

with respect to initial values and parameters.

The sensitivities $\mathfrak{S}_z(t_f, t_i)$ and $\mathfrak{S}_p(t_f, t_i)$ are given by the equations (3.47) and (3.48), respectively.

For the switching points

$$t_{s,0} < t_{s,1} < \ldots < t_{s,n_{ms}}$$

detected in the first integration run we obtain a sequence of entered areas. The global monitor $\mathfrak{M}^{global}$ labels these areas and detects points where modified matching conditions have to be applied.

The continuity conditions (1.12b) in explicit vector form without implicit discontinuity updates at the multiple shooting points but allowing explicit transition formulations can be written in the form

$$\begin{pmatrix} x(t_1; s_0^z, \hat{q}_0) & - & s_1^x \\ x(t_2; s_1^z, \hat{q}_1) & - & s_2^x \\ & \vdots & \\ x(t_{n_{ms}}; s_{n_{ms}-1}^z, \hat{q}_{n_{ms}-1}) & - & s_{n_{ms}}^x \end{pmatrix} \in \mathbb{R}^{n_{ms}}. \tag{5.4}$$

For the corresponding Jacobian with additionally introduced conditions arising from the algebraic conditions

$$g(s_{\hat{n}}^z, \hat{q}_{\hat{n}}, \tau_{\hat{n}}) = 0$$

in (1.12b), we obtain

$$\begin{pmatrix} X_0^x & X_0^y & X_0^u & -\mathbb{1} & & & & & & & \\ G_0^x & G_0^y & G_0^u & & & & & & & & \\ & & & X_1^x & X_1^y & X_1^u & -\mathbb{1} & & & & \\ & & & G_1^x & G_1^y & G_1^u & & & & & \\ & & & & & & \ddots & & & & \\ & & & & & & X_{n_{\text{ms}}}^x & X_{n_{\text{ms}}}^y & X_{n_{\text{ms}}}^u & -\mathbb{1} \\ & & & & & & G_{n_{\text{ms}}}^x & G_{n_{\text{ms}}}^y & G_{n_{\text{ms}}}^u & \end{pmatrix}, \tag{5.5}$$

where n_{ms} is the number of multiple shooting intervals. For a multistage formulation of n_{sta} model stages the matrix structure reappears on every stage with the dedicated number of multiple shooting intervals. For simplicity of presentation we consider one model stage in the following. The generalization to more model stages is straightforward, and has been implemented.

$X^x_{\hat{n}}$, $X^y_{\hat{n}}$ and $X^u_{\hat{n}}$ are the Jacobian matrices of $x_{\hat{n}}(t^{\hat{n}+1}; s^z_{\hat{n}}, q_{\hat{n}})$ with respect to $s^x_{\hat{n}}$, $s^y_{\hat{n}}$ and $s^u_{\hat{n}}$, respectively, i.e.,

$$\begin{aligned} X^x_{\hat{n}} &= \frac{\partial x(t^{\hat{n}+1}; s^z_{\hat{n}}, \hat{q}_{\hat{n}})}{\partial s^x_{\hat{n}}}, \\ X^y_{\hat{n}} &= \frac{\partial x(t^{\hat{n}+1}; s^z_{\hat{n}}, \hat{q}_{\hat{n}})}{\partial s^y_{\hat{n}}}, \\ X^u_{\hat{n}} &= \frac{\partial x(t^{\hat{n}+1}; s^z_{\hat{n}}, \hat{q}_{\hat{n}})}{\partial s^u_{\hat{n}}}. \end{aligned}$$

The consistency condition Jacobians are given by

$$\begin{aligned} G^x_{\hat{n}} &= \frac{\partial g(t^{\hat{n}+1}; s^z_{\hat{n}}, \hat{q}_{\hat{n}})}{\partial s^x_{ij}}, \\ G^y_{\hat{n}} &= \frac{\partial g(t^{\hat{n}+1}; s^z_{\hat{n}}, \hat{q}_{\hat{n}})}{\partial s^y_{ij}}, \\ G^u_{\hat{n}} &= \frac{\partial g(t^{\hat{n}+1}; s^z_{\hat{n}}, \hat{q}_{\hat{n}})}{\partial s^u_{ij}}. \end{aligned}$$

For discontinuities a priori known to appear, an additional model stage of zero duration can be introduced by using generalized continuity conditions of the form

$$\hat{c}(\hat{q}_{\hat{n}}) + s_{\hat{n}} - s_{\hat{n}+1} = 0, \quad t^{\hat{n}+1} = (t^{\hat{n}})^+. \tag{5.6}$$

However, the appearance of the discontinuities is implicitly given and thus a modification of the consistency conditions is merely necessary if the sign structure of the switching vector σ_j changes from the final point of the preceding interval $\hat{n}$ to the following interval indexed $\hat{n}+1$. At the price of a significantly growing QP we could a priori introduce additional generalized continuity conditions (5.6) at every multiple shooting point and control its actual entries by the global switching monitor $\mathfrak{M}_{global}$ which would change the matrix of the Jacobian of the matching conditions (5.5) in the above described form.

If the area $\mathfrak{R}_k$ given by the switching vector at the ending point $x(t^{\hat{n}+1}; s_{\hat{n}}, \hat{q}_{\hat{n}})$ and the area $\mathfrak{R}_l$ given by start vector in the preceding multiple shooting interval are not the same or in short if

$$\operatorname{sgn}(\sigma(t^{\hat{n}+1})^-) \neq \operatorname{sgn}(\sigma(t^{\hat{n}+1})^+),$$

the consistency conditions have to be modified. Here $\sigma(t^{\hat{n}+1})^-$ denotes the switching vector corresponding to $z(t^{\hat{n}+1}; s_{\hat{n}}, \hat{q}_{\hat{n}})$, $\sigma(t^{\hat{n}+1})^+$ is the switching vector at $z(t^{\hat{n}+1}; s_{\hat{n}+1}, \hat{q}_{\hat{n}+1})$ and $\neq$ describes a componentwise comparison.

We replace (5.4) by

$$\begin{pmatrix} x(t_1; s_0^z, \hat{q}_0) & - & s_1^x \\ x(t_2; s_1^z, \hat{q}_1) & - & s_2^x \\ & \vdots & \\ {}^x\mathcal{T}^{\hat{n}+1} x(t^{\hat{n}+1}; s_{\hat{n}}^z, \hat{q}_{\hat{n}}) & - & s_{\hat{n}+1}^x \\ & \vdots & \\ x(t_{n_{ms}}; s_{n_{ms}-1}^z, \hat{q}_{n_{ms}}) & - & s_{n_{ms}}^x \end{pmatrix} \in \mathbb{R}^{n_{ms}} \tag{5.7}$$

where the operator ${}^x\mathcal{T}^{\hat{n}}$

$${}^x\mathcal{T}^{\hat{n}+1} x(t^{\hat{n}+1}; s_{\hat{n}}^z, \hat{q}_{\hat{n}})^- = x(t^{\hat{n}+1}; s_{\hat{n}}^z, \hat{q}_{\hat{n}})^+$$

maps the left hand limit of the state to the right.

The limits are given by

$$x(t^{\hat{n}+1}; s_{\hat{n}}^z, \hat{q}_{\hat{n}})^{\pm} = \lim_{\epsilon \mapsto 0} x(t^{\hat{n}+1} \pm \epsilon; s_{\hat{n}}^z, \hat{q}_{\hat{n}}).$$

The discontinuity update is of the form

$$\left(\frac{\partial x(t^{\hat{n}+1}; s_{\hat{n}}, q_{\hat{n}})^+}{\partial s_{\hat{n}}}\right)^T = \underbrace{\left(\frac{\partial x(t^{\hat{n}+1}; s_{\hat{n}}, q_{\hat{n}})^+}{\partial x(t^{\hat{n}+1}; s_{\hat{n}}, q_{\hat{n}})^-} \cdot \frac{\partial x(t^{\hat{n}+1}; s_{\hat{n}}, q_{\hat{n}})^-}{\partial s_{\hat{n}}}\right)^T}_{\mathfrak{X}_{\hat{n}+1}^s}$$

with

$$\mathfrak{X}_{\hat{n}+1}^z = \nabla_{s_{\hat{n}+1}^z} {}^x\mathcal{T}^{\hat{n}+1} x(t^{\hat{n}+1}; s_{\hat{n}}^z, \hat{q}_{\hat{n}}) \in \mathbb{R}^{n_x \times n_z}.$$

The matrix

$$\frac{\partial x(t^{\hat{n}+1}; s_{\hat{n}}, q_{\hat{n}})^+}{\partial x(t^{\hat{n}+1}; s_{\hat{n}}, q_{\hat{n}})^-} \tag{5.8}$$

describes an approximation to a discontinuity update.

In complete analogy we replace $G_{\hat{n}+1}^z$ in the consistency conditions by $\mathfrak{G}_{\hat{n}+1}^z$, again using the update operator

$${}^z\mathcal{T}^{\hat{n}+1} y(t^{\hat{n}+1}; s_{\hat{n}}^z, \hat{q}_{\hat{n}})^- = y(t^{\hat{n}+1}; s_{\hat{n}}^z, \hat{q}_{\hat{n}})^+$$

at the end point of the interval $\hat{n}$, i.e.,

$$\mathfrak{G}^z_{\hat{n}+1} = \nabla^z_{s_{\hat{n}+1}} {}^y\mathcal{T}^{\hat{n}+1} y(t^{\hat{n}+1}; s^z_{\hat{n}}, \hat{q}_{\hat{n}}) \in \mathbb{R}^{n_y \times n_z}.$$

Assumption 5.9. *We make the assumption of sufficiently wide overlapping areas $\mathfrak{R}_m$ and $\mathfrak{R}_k$ in the sense that the right hand sides remain evaluable in an area beyond their core validity since discontinuity updates are performed at the point $t^{\hat{n}+1}$ which does not have to coincide with the zero-crossing of the switching function (see Figure 5.3).*

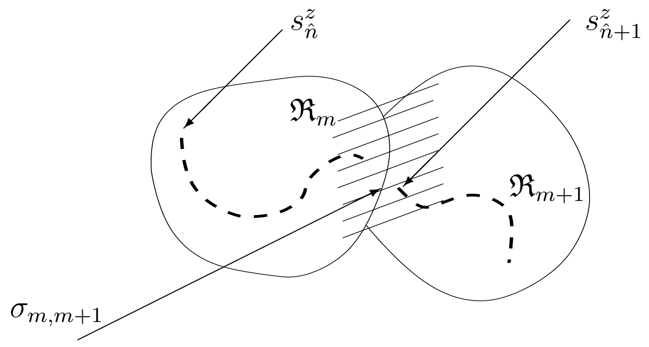

Figure 5.3: In addition to Figure 5.2 the shaded part signifies the overlap region of the two areas.

Remark 5.10. For vanishing discontinuities in states and derivatives, the operator ${}^x\mathcal{T}^{\hat{n}}$ is the identity mapping and the matrix (5.8) transforms into the identity matrix.

Remark 5.11. For a zero-crossing of the respective switching function at $t_s = t^{\hat{n}+1}$, the update turns out to be equivalent to the additional introduction of an interval of zero length with merely giving the jump-conditions.

One of the main advantages of the multiple shooting method is the possibility to a priori feed process information to the integrator via intermediate start values. Since the Assumption 5.9 turns out to be almost always valid in practice, for carefully chosen initial values, the right hand sides of the DAE system in general remain evaluable.
Nevertheless, if Assumption 5.9 does not hold, i.e., if the execution of the switching causes the right hand side to be inevaluable, we locally choose the initial values of the following interval via integration, i.e., interrupt the SQP process and reintegrate

the multiple shooting interval $\hat{n}+1$ starting from the end point of the preceding interval.
This can be interpreted as a temporary reduction of the number of multiple shooting intervals. In this case the inherent parallelity of multiple shooting is broken at the interval where the new start values are obtained via integration.

Since the updates do not destroy the specific structure of the problem, the search space decomposition (see 2.30) and thus the decomposition of the NLP especially adapted to DAE optimization can be transferred to Leineweber's approach (see [Lei99]).

5.4 Controlling trajectories with differing switching structures

The local discontinuity monitor $\mathfrak{M}^{l,\hat{n}}_{local}$ in the l-th SQP step of multiple shooting interval $\hat{n}$ provides a local area chronology $\mathfrak{C}^{l,\hat{n}}$ which can be interpreted as a digraph $(\mathcal{V}^l, \mathcal{E}^l)$ with the vertices

$$\mathcal{V}^l = \{1, 2, \ldots, n_{\mathfrak{C}^{l,\hat{n}}}\},$$

where $n_{\mathfrak{C}^{l,\hat{n}}}$ gives the number of the locally entered areas in interval $\hat{n}$ and $\mathcal{E}^{l,\hat{n}}$ describes the edges given by transitions to other areas discarding self-loops[†].

Since Theorem 9 does not hold if new switching points arise or existing switching points vanish - briefly if the digraph $\mathfrak{C}$ changes - the sensitivities

$$\frac{\partial z(t_f^{\hat{n}})}{\partial z(t_i^{\hat{n}})}(\mathfrak{C}^{l,\hat{n}})$$

in the area $\mathfrak{R}_m$ need not change continuously for $\mathfrak{C}^{l,\hat{n}} \neq \mathfrak{C}^{l+1,\hat{n}}$.

Example 5.12. Let the single tray equipped with a heating (3.3.6) be fed with feed stream F. Due to the objective an increased heat supply may cause the bi-phase state to turn temporarily to an overheated mono-phase state (see Example 5.3, schematically in the right picture of Figure 5.4).

[†]In principle every integrator run leaving the switching vector unchanged could be interpreted as a *self-loop*.

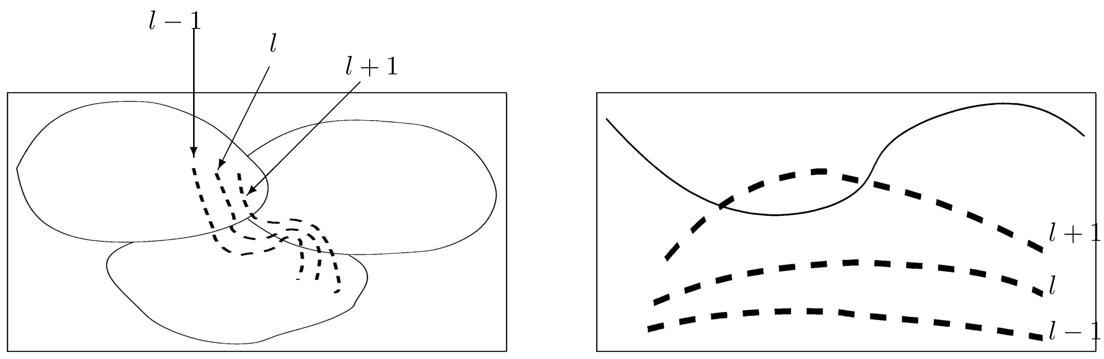

Figure 5.4: Schematic 2D illustration of changes in the digraph $\mathfrak{C}$ in consecutive steps l. The dashed lines symbolize 2D projections of the trajectories in the phase space. In the right graphics step $l+1$ leads to two newly arising switchings. The left picture shows a vanishing switching point in step $l+1$.

The local digraphs $\mathfrak{C}^{l,\hat{n}}$ of the individual multiple shooting intervals $\hat{n}$ in SQP step l are coupled to a global digraph $\mathfrak{C}^{l}_{global}$.

Starting from the initial in general infeasible trajectory the optimization procedure is performed as long as the digraph $\mathfrak{C}^{l}_{global}$ does not change in successive steps. If the monitor $\mathfrak{M}_{global}$ reports a change from the digraph $\mathfrak{C}^{l}_{global}$ to $\mathfrak{C}^{l+1}_{global}$, a feasible trajectory with respect to the validity of $\mathfrak{C}^{l}_{global}$ is searched while still keeping the trajectory information for the trajectory with the digraph $\mathfrak{C}^{l+1}_{global}$ if possible.

Feasibility search with active $\mathfrak{C}^{l}_{global}$ if $\mathfrak{C}^{l+1}_{global} \neq \mathfrak{C}^{l}_{global}$:

1. *Start.*
   Restart SQP-step l with the old Hessian approximation.

2. *Search.*
   Locally weight the Lagrange and/or Mayer term

   $$F(w) \rightarrow \beta_k F(w)$$

   and correspondingly the l_1 penalty function (2.27)

   $$\begin{aligned}\Psi_k(w,\varpi,\tau) =& \beta_k F(w) \\ &+ \sum_{i=1}^{n_G} \varpi_i |G_i(w)| + \sum_{j=1}^{n_H} \tau_j |\min(0, H_j(w))|.\end{aligned} \tag{5.13}$$

   Perform line search for the current factor β_k.

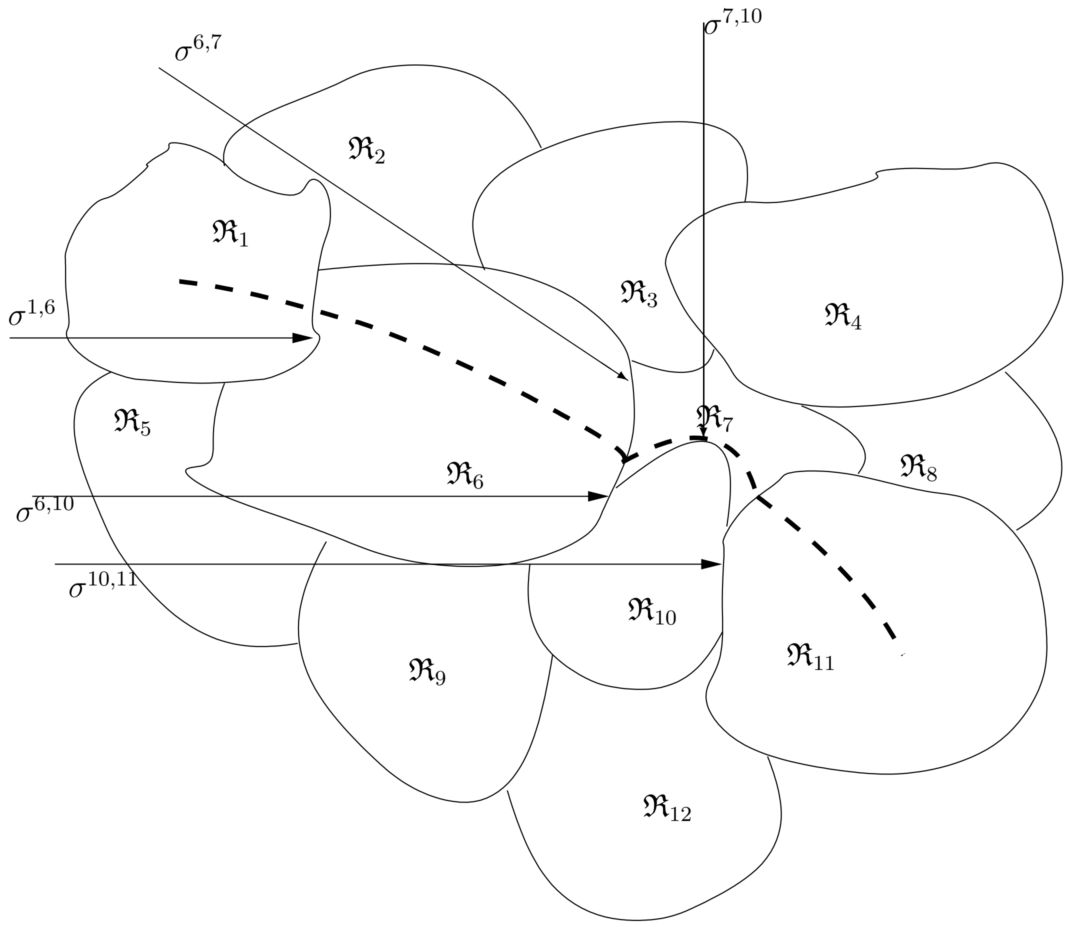

Figure 5.5: Schematic representation of a trajectory (dashed) in a 2D cut through the phase space. The components of the switching vector σ, the switching functions $\sigma^{i,j}$, show zero-crossings at the area boundaries.

3. *Convergence check.*

   (a) *Convergence: Save upper bound on the solution.*
   When feasibility is obtained the local solution represents an upper bound on the solution.

   Start a line search with $\beta = 1$ allowing steps violating the digraph $\mathfrak{C}^l_{global}$. Go to 4..

   (b) *No Convergence: Go back to SQP step $l+1$. Restart with digraph $\mathfrak{C}^{l+1}_{global}$.*
   If no convergence can be obtained within the active switching structure given by the digraph $\mathfrak{C}^l_{global}$, the algorithm continues with SQP-step $l+1$.

4. *Termination.*
   If a local optimum (KKT-point) (2.29) is obtained, the local solution is com-

pared with the active upper bound if available (see 3.(a)). The lower of these is a local solution.

If no local optimum (KKT-point) (2.29) is obtained, restart with trajectory adjacent to the graph $\mathfrak{C}^{l+1}_{global}$.

If a feasible trajectory can be found in such a way that it represents an upper bound for the local optimum search, the trajectory corresponding to the digraph $\mathfrak{C}^{l+1}_{global}$ is used as a new starting point for the SQP-algorithm. Local solutions are valued with respect to the stored feasible solution corresponding to digraph $\mathfrak{C}^{l}_{global}$. The algorithm ends when a local solution is found without modification of the objective. Either this solution is regarded as a local optimum or if the preceding local solution stored as an upper bound on the solution is better with respect to the objective, this upper bound is regarded to be the solution, respectively.

The factor β is chosen to decrease in consecutive feasibility search steps according to

$$\beta_{k+1} = \frac{\beta_k}{4},$$

and we restrict to a search depth $k = k_{max}$.

After digraph changes we reinitialize the Hessian approximation (2.2.1.2) which in this step leads to a steepest descent step ([Bau01]).

In Figure 5.5 we schematically present a trajectory by the 2D cut through the phase space. If jumps occur, i.e., if a jump function $\hat{\Delta}_j$ is given for one switching function, the start point after the switch does not coincide with the point before the switch. The schematic trajectory shown symbolizes a situation as it is typically obtained as a local feasible solution with a small β_k.

The algorithm has proven to be applicable to large scale optimal control problems. It builds on the a priori intelligent choice of initial values for the intermediate starting points at the multiple shooting nodes. In this case only few subsequent line searches have to be performed for differing β_k within the respective digraphs $\mathfrak{C}_{global}$. The local search for feasible trajectories is crucial in the sense that consecutively diminished upper bounds for the objective are calculated.

Since the quasi-Newton approximation has to be reinitialized with the identity matrix for changing digraphs $\mathfrak{C}_{global}$ in consecutive SQP steps a superlinear convergence rate can in general not be expected. Looking towards real-time applications it could be interesting to extend our integrator in such a way, that it provides second derivative information.

Chapter 6

Applications

In this chapter we present several applications of the algorithms introduced in the preceeding chapters. In the first section we focus on a multi-tray distillation process described by a rigorous model which was developed in cooperation with BASF AG, Ludwigshafen.

A more detailed description of the equation can be found in the appendix (see Chapter A).

Due to the ill-condition of the Jacobian of the right hand sides w.r.t. the state vector (see Equation (4.10) of the rigorous model of this multi-phase process the treatment requires accurate derivative information.
Since it is hardly possible to exploit sparsity structures in right hand side Jacobians via symbolic differentiation, automatic differentiation techniques (see Chapter 4) have to be used. The implicitly defined discontinuities in the states require an appropriate discontinuity treatment both in the integrator and the optimization algorithm.

The second section deals with on the optimization of biotechnological models. The development of these models was also accomplished in cooperation with BASF AG, Ludwigshafen, based on existing processes. We first present a comparatively easy biotechnological model which was in its original form presented by Kühn ([Küh02]). In this thesis, it is for the first time treated with a derivative based algorithm which allows pointwise non-smooth trajectories.

Afterwards, we present a more complex biotechnological process. It was originally modeled by King ([Kin94]). Schäfer in his diploma thesis ([Sch99a]) smoothed the discontinuities an treated and used a classical SQP optimal control algorithm. We compare our optimization results with those obtained via optimization of the smoothed model Schäfer proposed for this process.

Finally we apply our algorithms in the feed back context. The optimization of the second biotechnological process which we present in the last section is performed using feed back on shrinking horizons. We take model deficiencies into account by characteristically perturbing model states after certain a priori given sampling times.

6.1 Optimization of a distillation process

The distillation process [Kis92] physically separates a mixture into two or more products that have different boiling points. If a liquid mixture of two volatile materials is heated, the vapor coming out has a higher concentration of the more volatile material than the liquid from which it was evolved. Nowadays many distillation columns are composed of multiple trays which are vertically arranged above each other.

Distillation processes play a crucial role in chemical engineering. Since about 80% of the overall energy consumed in chemical industry is in fact consumed in distillation processes, optimization is both ecologically and economically of utmost importance. To mention another number, the US portion of the world's annual energy consume amounts to one fourth, 7% of this is spent in the chemical industry.

The system as a whole (see Figure 6.1) can for very high numbers of trays be regarded as a discretization of a partial differential equation where the spatial discretization is naturally given by the individual trays*. We exploit this observation in [BPSB$^+$04] to generate start values for the rigorous model. Another approach are the so called *Wave models* (Kienle et al. [Kie98]) which can be interpreted as homogenization of the complete distillation column and thus are restricted in their applicability to columns with very many trays or to porous media columns.

Due to its complex dynamics the rack-in of the overall process is usually not taken into account when modeling distillation processes. For this task homogenization

*Some modern distillation columns do not show the classical tray structure, but are filled up with porous media given the column a continuous structure.

techniques can hardly be applied due to complex physical phase equilibria on the individual trays. Rigorous models of the processes are required. In the following we propose a distillation column model which explicitely treats the complex dynamics of the liquid vapor mixture on the individual trays.

6.1.1 Modeling

Let us describe the modeling of the multi-tray distillation column. We propose a thermostatic model which can be used to simulate and optimize rack-in processes. Nevertheless we restrict to *ideal* trays in that sense, that equilibria in the individual trays are reached instantaneously. We assume ideal mixtures and *perfect gases.* Spatial fluctuations or variations in the concentrations etc. one the individual trays are discarded, which allows us to work with a differential algebraic equation (DAE) instead of a partial differential equation. A tabulated overview of the model can be found in the appendix (see Appendix A). In Figure (6.1) a scheme of a distillation column is shown.

The component molar balances are described by n_C differential equations per tray

$$\begin{aligned}\frac{dn_{i,j}}{dt} =& L^{\text{in}} X_i^{\text{in}} \delta_{j,n_{\text{feed}}} - L_j^{\text{out}} X_{i,j}^{\text{out}} - V_j^{\text{out}} Y_{i,j}^{\text{out}} \\ &+ L_{j+1}^{\text{out}} X_{i,j+1}^{\text{out}} \Theta(n_{\text{tray}} - j - 1) \\ &+ V_{j-1}^{\text{out}} Y_{i,j-1}^{\text{out}} \Theta(j-1).\end{aligned} \tag{6.1}$$

Here $\Theta(\cdot)$ is the Heaviside function (3.54) which gives the equations for reboiler and condenser a special structure. The *Kronecker* δ distinguishes the feed tray from the others. The indices $i = 1, \dots, n_C$ stand for the components in the mixture in the respective tray and $j = 1, \dots, n_{\text{tray}}$ describe the different trays starting from the reboiler $j = 1$ and ending with the condenser $j = n_{\text{tray}}$. The $n_{i,j}$ are the molar masses, the vectors Y_i describe the molar fraction of components i in the gas-phase and x_i describe the molar fraction of components i in the liquid phase. L_j^{out} and V_j^{out} are the liquid and vapor outstreams off the j-th tray, respectively. The energy balance on the trays is given by

$$\begin{aligned}\frac{dE_j}{dt} =& L^{\text{in}} h^{L,\text{in}} \delta_{j,n_{\text{feed}}} - L_j^{\text{out}} h_j^{L,\text{out}} - V_j^{\text{out}} h_j^{V,\text{out}} \\ &+ L_{j+1}^{\text{out}} h_{j+1}^{L,\text{out}} \Theta(n_{\text{tray}} - j - 1) \\ &+ V_{j-1}^{\text{out}} h_{j-1}^{V,\text{out}} \Theta(j-1) \\ &+ Q\delta_{j,n_1} \\ &- Q^{cool} \delta_{j,NTray},\end{aligned} \tag{6.2}$$

where the $h_j^{L/V}$ denotes the individual enthalpy on the tray j, $h^{L,in}$ stands for the feed enthalpy on the feed tray. Q and Q^{cool} stand for the heating and cooling and the reboiler and condenser, respectively.

The liquid feed is usually added in the middle of the column, i.e., fed to the tray $(NTray + 1)/2$ (for $NTray$ odd). This is due to the fact that we want the composition of the feed to be closest to the composition of the mixture at the respective tray since this leads to the least possible entropy growth. This is assumed to be the case in the feed tray[†].
We explicitely take heat capacities of the column into account (see Appendix A).

Based on Dalton's law stating that the overall pressure of a gas is composed of the individual pressures of the components, we transform Henry's law (see e.g. [AdP02]) to

$$Y_{i,j} = k_{i,j} X_{i,j}, \qquad i = 1, \ldots, n_C, \;\; j = 1, \ldots, n_{tray} \tag{6.3}$$

which we use as equilibrium conditions subject to

$$\sum_i Y_{i,j} = 1. \tag{6.4}$$

The k-values are determined by

$$k_{i,j} p_j = p_{j,i}^{sat}, \qquad i = 1, \ldots, n_C, \;\; j = 1, \ldots, n_{tray}, \tag{6.5}$$

where p_j are the pressures on the individual trays (see Appendix A).
In our application we focus on a mixture of four components. The inert gas ($i = 4$) is assumed to be thousand times as volatile as component $i = 1$

$$k_{4,j} = 1000 \cdot k_{1,j} \tag{6.6}$$

[†]For other compositions of the feed stream it can for energy conservation reasons be advantageous to feed in other tray instead.

instead of (6.5). The thermodynamic vapor fraction ψ_j is calculated via

$$\psi_j \cdot (h_j^{V,sat} - h^{L,sat_j}) = (h_j - h^{L,sat}), \tag{6.7}$$

allowing of overheated vapor causing an implicit model change and a jump in the first derivative of the state functions.
Additionally, overpressure valves at the top of each tray let the vapor stream out when the ambient pressure is exceeded. These valves are technically realized by a porous medium. These implicit changes on every tray also cause implicit discontinuous changes of the right hand sides. The third group of implicit discontinuities are due to the weir functions. These functions signify, whether the current fluid level in the respective tray reaches the weir height.

To describe the dependency of the vapor pressure of each of the components on the temperature T, we use the modified Antoine equations of the form

$$\log p_i = A_i + \left(\frac{B_i}{C_i + T} \right) + D_i \log T + E_i T^2 \tag{6.8}$$

with parameters A_i, B_i, C_i, D_i and E_i $(i = 1, 2, 3)$.
The specific density is given by the equation

$$\log \left(\frac{\rho_i^{liq}}{F_i} \right) = - \left(1 + \left| 1 - \frac{T}{H_i} \right|^{J_i} \right) \cdot \log G_i, \tag{6.9}$$

where F_i, G_i, H_i and J_i again denote parameters for $i = 1, 2, 3$. For the inert gas we assume

$$\rho_4^{liq} \equiv 1. \tag{6.10}$$

Since the term $1 - \frac{T}{H_i}$ in (6.9) remains positive the argument's direct evaluation via the formula (6.9) is uncritical despite of J_i being of order of magnitude of 10^{-1}. Nevertheless, as can be seen in e.g. (6.8) and (6.9), the model describing the distillation process is of very high nonlinearity.

6.1.1.1 Implicit discontinuities in the model

Our modeling leads to three implicitly given discontinuities on every tray arising from

- changes from a mono-phase state of uniquely vapor flooded trays to the coexistence of a liquid and a vapor phase, i.e., we obtain the switching function

$$\psi_j - 1 = 0,$$

- weir-overflows of the liquid phase to lower trays, i.e., we obtain the switching function
$$\Delta h_j^{weir} = 0,$$
- on- and off-switching of outstream of vapor phase to above lying trays due to overpressure, i.e., we obtain the switching function
$$p_j - p^{ref} = 0$$
with a system parameter p^{ref}.

6.1.2 Rack-in process of an instationary distillation process

Distillation columns are often flooded with the inert gas nitrogen N_2 before new start-up.
The aim of the optimization under consideration is to find an energy optimal way of running the process from the initial mono-phase state of the column being flooded with inert gas to an operation state, given by a certain purity of the mixture in reboiler and condenser under diverse constraints.

The algebraic equations (6.6) which describe the property of the inert gas on the respective tray of being highly volatile compared to the other substances strongly contributes to the high condition number

$$\kappa(\mathcal{J}_g) = \|\mathcal{J}_g\| \cdot \|\mathcal{J}_g^{-1}\|$$

of the model Jacobian $\mathcal{J}_g = \frac{\partial g}{\partial y} \in \mathbb{R}^{n_y \times n_y}$ of the algebraic equations. Thus condition numbers

$$\kappa(\mathcal{J}_g) \approx 2 \cdot 10^9 \tag{6.11}$$

with respect to the Hilbert norm for the described configuration arise.

Due to the bad condition (6.11) of the right hand side Jacobian, the simulation of the process briefly described in the last section is only possible, when accurate derivative information of the right hand side of the DAE is provided. This is done by using the strategies presented in Chapter 4.

The classical way of starting a distillation process is to completely flood the column initially flooded with the inert gas with liquid feed and to set the distillate stream

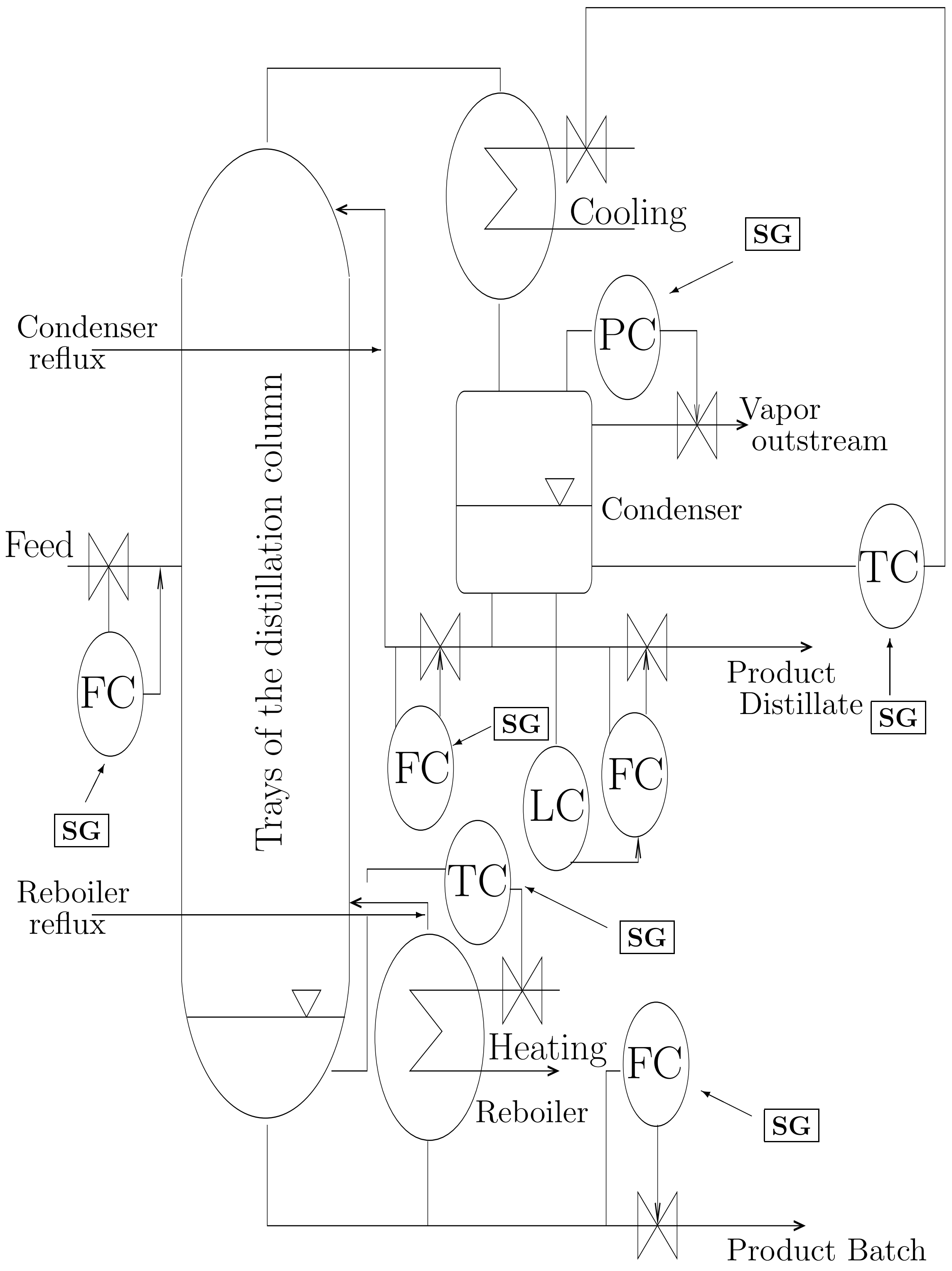

Figure 6.1: Schema of the modeled distillation column. TC, PC, LC and FC stand for the controllers, $\boxed{\textbf{SG}}$ for the control functions

D (see Appendix A) to almost zero. We use this as an initial trajectory leading to a digraph $\mathfrak{C}$ (see Chapter 5) which gives the actually valid switching chronology and thereby obtain initial guesses for the states at the multiple shooting nodes.

For an objective of the form[‡]

$$\min_{t_f,z,u} \Xi(t_f, z(t_f), p) = \lambda_1 \int_0^{t_f} \left(X_0^{cond} - X_0^{condref}\right)^2 dt \tag{6.12a}$$

$$+\lambda_2 \int_0^{t_f} \left(X_0^{reb} - X_0^{rebref}\right)^2 dt \tag{6.12b}$$

$$+\lambda_3 \int_0^{t_f} Q\, dt \tag{6.12c}$$

$$+\lambda_4 \int_0^{t_f} Q^{cool}\, dt \tag{6.12d}$$

$$-\lambda_5 \frac{\int_0^{t_f} DF_{cond}^{l,out}\, dt}{\int_0^{t_f} F_{in}\, dt} \tag{6.12e}$$

$$+\lambda_6 \int_0^{t_f} \left(\sum_{j=1}^{3} X_{j,reb}^{v,out}\right) F_{reb}^{v,out}\, dt \tag{6.12f}$$

$$+\lambda_T t_f, \tag{6.12g}$$

with the control vector u subsuming the added energy Q in the reboiler, the cooling Q^{cool}, the feed control F_{in} and the controls not explicitly involved in the objective, the reflux ratio in the condenser R^{cond} and the reboiler R^{reb} and the outflow control in the condenser, we obtain the optimal control profiles shown in the upper two rows in Figure 6.2. The first part of the objective (6.12a) guarantees the purity of the condenser product, $X_0^{condref}$ stands for a reference purity of component indexed 0 in the condenser. The second part of the objective (6.12b) penalizes the integral quadratic deviation from the advised purity X_0^{rebref}, which is chosen to be small since we want the fraction of the most volatile component in the liquid holdup to be sufficiently small. The variables X_0^{cond} and X_0^{reb} stand for the liquid molar fraction of component indexed 0 in reboiler and condenser, respectively. The weighted

[‡]motivated by our project partner BASF AG, Ludwigshafen

rate of yield is described by (6.12e). The term (6.12f) penalizes vapor outflow from the condenser. The term (6.12f) weighted with λ_6 measures the lost vapor outflow. Since the inert gas has to be stripped off it is weighted with the sum of the partial sums of the non-inert gas components so that for pure inert gas in the condenser this term does not contribute to the penalty function. The last term (6.12g) penalizes the overall time needed.

Inequality constraints are given with respect to temperatures and pressures on the trays. The initial configuration is fixed to the state of a inert gas flooded column. As one end point constraint we require an amount

$$M = \int_0^{t_f} DF_{cond}^{l,out}\, dt$$

of distillate. We demand a sufficiently high purity of the separated substance in the condenser and a sufficiently low concentration in the reboiler which is obtained because of the formulation of the objective (6.12).

The optimization of the above (and in Appendix A) described process leads to a reduction of the objective function (6.12) of 21.6% compared to the classical strategy of flooding the column completely with the liquid feed. The comparative result is obtained by performing a simulation with hand chosen control values after flooding the column and letting the inert gas stream out. The strategy in conform to the classical way to rack-in such a process.

In Figure 6.2 we present the optimal control profiles for piecewise constant controls and six characteristic state profiles of the solution trajectory. The feed (first row, first column, Figure 6.2) is not piped in a constant profile. First a comparatively little amount of feed is fed into the column; just enough (see the plot of Δ_{reb}^{weir}, third row third column, Figure 6.2) for a sufficient amount of liquid holdup to flow to the reboiler. After the start phase of the rack-in process the heating is kept almost constant (first row, second column, Figure 6.2). This - in combination with the profiles of the feed and the reflux in the reboiler leads to a wave like profile in the liquid holdups as can be seen in the plot of the liquid holdup in the reboiler (Δh_{reb}^{weir}) and in the plot of the liquid holdup in the tray above the feed tray (third row, first column). The thermodynamic vapor fraction in the condenser ψ_{cond}^{therm} (third row, second column) shows that the condenser after an initial phase condenses almost the whole vapor holdup. This can also be seen in the integral over the vapor outstream (fourth row, second column) which remains constant after the initial period where especially the inert gas is drawn off. The plot of the vapor holdup of the inert gas in the reboiler (fourth row, third column) shows that the total condensing phase and the vanishing of the vapor holdup of the inert gas coincide. k_0^3 gives the k-value of the inert gas (fourth row, first column) in the reboiler varying by several orders of magnitude.

Crucial for the optimization is that trajectories with differing digraphs can be compared (see Chapter 5). The digraph of the starting step, described in the beginning of this chapter, strongly differs from the digraph valid in our solution.

All plots, variables and parameters in the context of the distillation process are given in arbitrary units, since the precise process control underlies nondisclosure in the context of the BMBF project 03-BOM1HD [§].

Every tray the column is composed of contains 81 algebraic and 5 differential variables. Additional inequality constraints are formulated with respect to pressures and temperatures on the trays. Also the levels of the liquid phase in reboiler and condenser are restricted by inequality constraints. Additional constraints arise for the volumes in the individual trays.
We used 28 multiple shooting intervals of equal length, and piecewise constant controls. We need 76 SQP steps to obtain the solution of an accuracy of $1.0 \cdot 10^{-4}$ with an integrator tolerance of $1.0 \cdot 10^{-6}$. The CPU time was 2709.6 seconds on a Linux machine (Intel Pentium IV, 2.53 GHz with a cache of 512 KB). For the solution trajectory we obtain 109 switching events.

[§] German ministry for education and science (in german "**B**undes**m**inisterium für **B**ildung und **F**orschung")

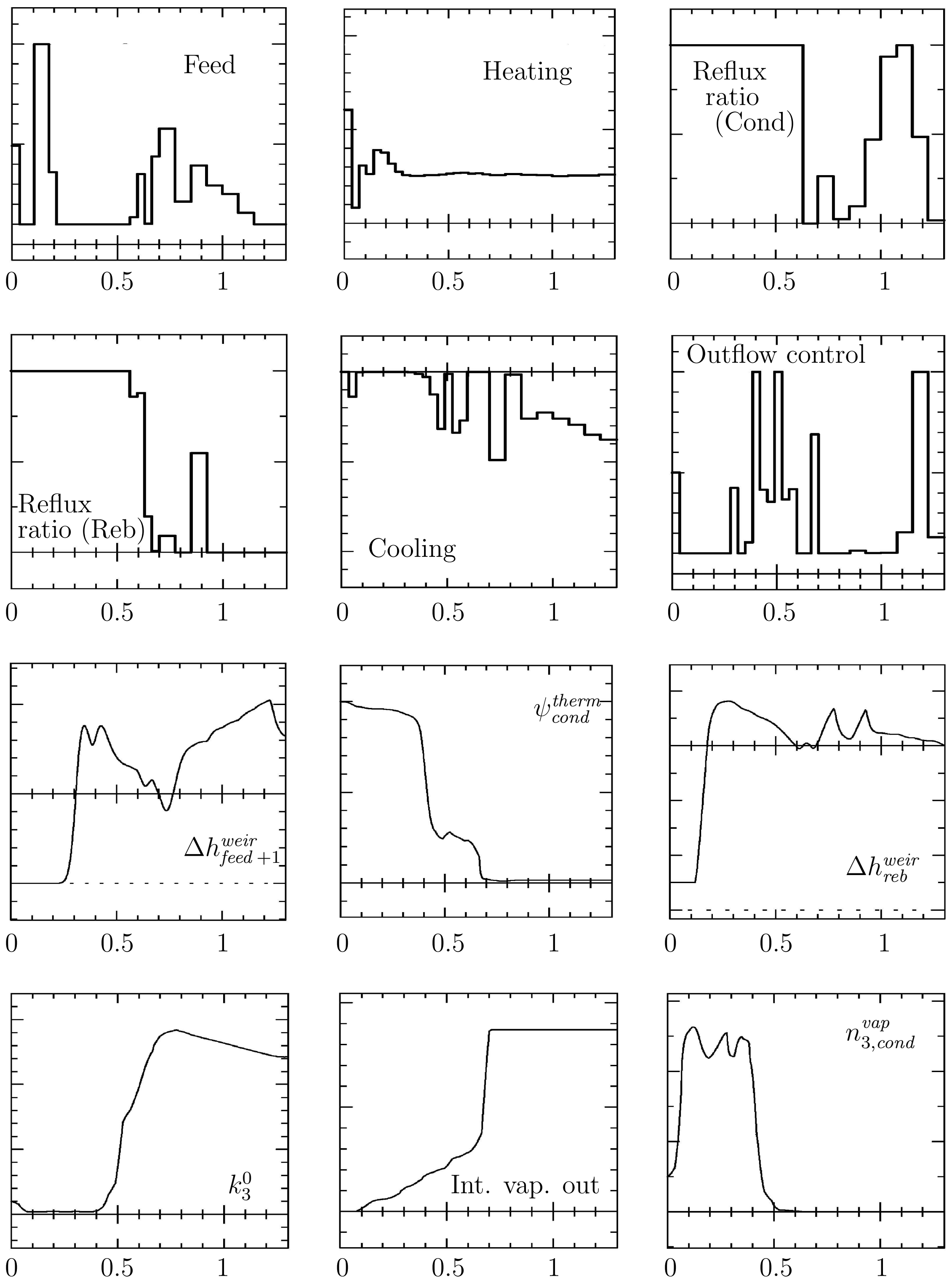

Figure 6.2: First two rows: Plots of optimal control profiles w.r.t. the objective (6.12) (*Cond* and *Reb* stand for condenser and reboiler). The lower two rows show plots of six states. For nondisclosure reasons the plots are not labeled on the ordinate.

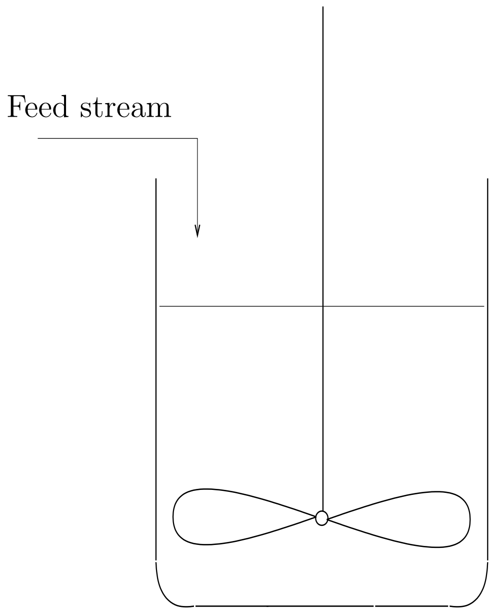

Figure 6.3: Diagram of a batch fermenter

6.2 Optimization of a biotechnological batch process

Due to their inherent complexity biotechnological processes were rarely modeled in the last years, in that sense that bio-mathematical models were developed. Often neural networks are applied to circumvent the in general highly complex modeling process.

In this section we first present a smaller application describing a fermentation process used for the production of antibiotics. The second subsection focuses on a more complex biotechnological process describing the fed-batch fermentation with *Streptomyces tendae.*

6.2.1 Optimization of a simple batch fermentation process

The results given in this subsection are rescaled due to disclosure obligation with BASF AG Ludwigshafen. The plots in Figures (6.4, 6.5) are labeled in arbitrary units.

In 3.3.7.1 we gave the equations for the small biotechnological fermentation (6.13) which we repeat here

$$f_0 \quad = \dot{X} \quad = r_X X - \frac{X}{V}\dot{V} \tag{6.13a}$$

$$f_1 \quad = \dot{P} \quad = r_P X - \frac{P}{V}\dot{V} \tag{6.13b}$$

$$\begin{aligned} f_2 \quad = \dot{S}_1 \quad = & -\left(\frac{1}{Y_{X/S_1}} r_X X + \frac{1}{Y_{P/S_1}} r_P X\right) \cdot \Theta(S_1) \\ & -\frac{S_1}{V}\dot{V} + \frac{1}{V}\left(F_A S_{F_A} + F_B S_{F_B}\right) \end{aligned} \tag{6.13c}$$

$$f_3 \quad = \dot{S}_2 \quad = -\left(\frac{1}{Y_{X/S_1}} r_X X\right) \cdot \Theta(S_2) - \frac{S_2}{V}\dot{V} \tag{6.13d}$$

$$f_4 \quad = \dot{V} \quad = F_A + F_B. \tag{6.13e}$$

where X denotes the biomass, P the product. The substrates are described by the variables S_1 and S_2 with the reaction rates

$$r_X = \mu_{max}\left(\frac{S_1}{S_1 + K_{S_1}} + \frac{S_2}{S_2 + K_{S_2}}\right) \tag{6.14}$$

and

$$r_P = \alpha\mu_{max}\left(\frac{S_1}{S_1 + K_{S_1}} + \frac{S_2}{S_2 + K_{S_2}}\right) X + \beta X. \tag{6.15}$$

This empirical Monod approach assumes a maximum growth rate μ_{max}. Y_i[¶] are the rate of yield coefficients. The coefficients α and β stand for the growth and non-growth associated product building. K_j[‖] are the product limitation coefficients ([Köh02]). The model parameters are obtained by a heuristical parameter estimation process based on measurement data obtained at BASF AG ([Küh02]).

Objective in the presented process is to maximize the product yield. Penalized is the integral over the feed stream

$$\max\left(\Phi = c_p(t_f) - \lambda \int_{t_i}^{t_f} F_1 + F_2 \, dt\right) \tag{6.16}$$

where λ is a weighting factor. We apply a piecewise constant control parameterization. Köhler sets the coefficients S_{F_A} and S_{F_B} describing the relative contribution of the feed streams F_A and F_B to one, only substrate S_1 is fed to the process.

[¶] $Y_i = Y_{X/S_1}, Y_{X/S_2}$ and $Y_{X/P}$

[‖] $j = S_1, S_2$

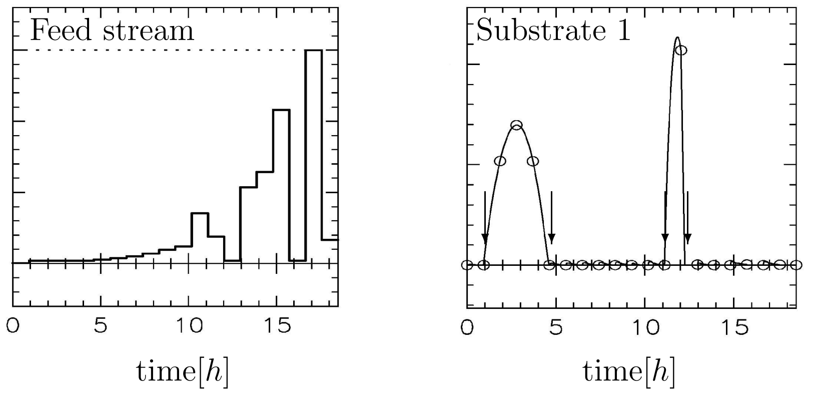

Figure 6.4: In the left plot the optimal feeding strategy with respect to the penalty term (6.16) is plotted for the Kühn model (6.13). The right plot shows the evolution of substrate 1. The ordinate of the plots is unlabeled for nondisclosure reasons. The arrows point to the time points where implicit discontinuities and thus model changes arise.

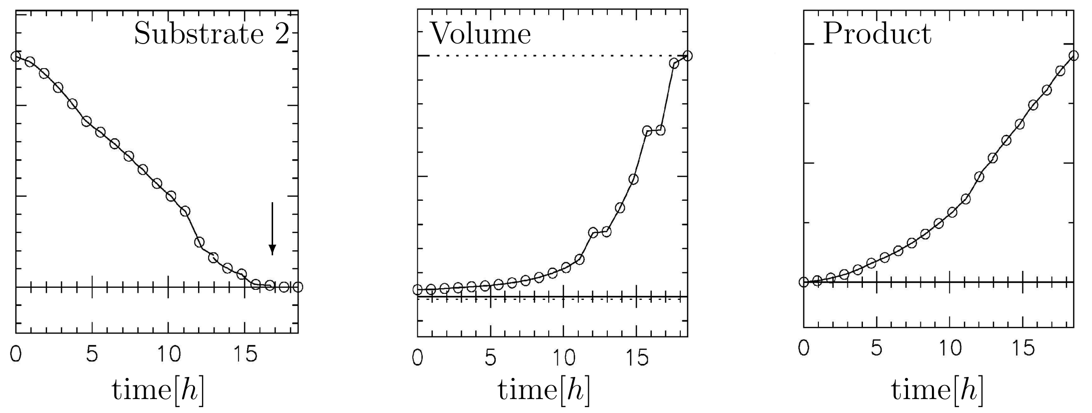

Figure 6.5: Solution trajectories for the Kühn model (6.13). The ordinate of the plots is unlabeled for nondisclosure reasons. The arrows point to the time points where implicit discontinuities and thus model changes arise.

Bounds on controls

$$0 \leq u(t) \leq u_{max}, \quad t \in [t_i, t_f] \tag{6.17}$$

are given by the maximum throughput that can be fed into the batch fermenter. The state constraints arise from the fact that each of the masses is required to be positive

$$\{S_{1,2}, P, V, X\}(t) \geq 0, \quad \text{for all } \; t \in [t_i, t_f].$$

In the left plot in Figure 6.4 we show the optimal feeding strategy and the resulting trajectory for substrate 1 in the right. In Figure 6.5 we present the trajectories of substrate 2, the overall volume and the product.

As already explained in the context of our implementation of the BDF integrator in Chapter 3, implicit discontinuities occur at points of vanishing substrates. As discussed the non classical continuation leads to a three valued switching structure. The initial trajectory for substrate 1 is of the form as plotted in Figure 3.6. As local solution we obtain, as can easily be seen a trajectory containing for substrate 1 four instead of one switches.

We used 20 multiple shooting intervals of equal length, and piecewise constant controls. The running time is 17.3 seconds on a Linux machine (Intel Pentium IV, 2.53 GHz with a cache of 512 KB).

6.2.2 Optimization of a biotechnological batch process

In this part we present the optimization of a biotechnological batch fermentation process of similar characteristics to the preceding one but equipped with three control functions and three implicit discontinuities.

The quite well studied process model describes the fed-batch fermentation with *Streptomyces tendae* for the antibiotics production (e.g. [Kin94], [KWB+95]). We restrict ourselves to giving the equations with implicit discontinuities and discussing the results. For a detailed model description refer to [Kin94].

The growth rate of amino acids is given by

$$\begin{aligned}\mu_{As} = &\left(\mu_{As1m} \frac{c_A}{c_A + K_{As1}} + \mu_{As2m} \frac{K_{As2}}{K_{As2} + c_A} \right) \cdot \Theta(c_A) \\ &\cdot \frac{c_C}{c_C + K_{AsC}} \cdot \rho(g_{Nu}) g_{Pr}\end{aligned} \tag{6.18}$$

and the formation of nucleotides by

$$\begin{aligned}\mu_{Nu} = &\left(\mu_{Nu1m}\frac{c_{Ph}}{c_{Ph}+K_{Nu1}} + \mu_{Nu2m}\frac{K_{Nu2}}{K_{Nu2}+c_{Ph}}\right)\cdot\Theta(c_{Ph})\\ &\cdot\frac{g_{As}}{g_{As}+K_{NuAs}}\cdot\rho(g_{Nu})g_{Pr},\end{aligned} \tag{6.19}$$

where Θ is the Heaviside function 3.54. Values for the constants K_X (K_{As1}, K_{As2}, K_{AsC}, K_{Nu1}, K_{Nu2}, K_{NuAs}) and for the constants μ_{As1m}, μ_{As2m} and μ_{Nu1m}, μ_{As2m} are taken from [Kin94]. c_A stands for the ammonia concentration, c_C for the glucose concentration and c_{Ph} for the phosphate concentration. The g_X stand for relative concentrations ($g_x = \frac{m_X}{V_X}$).

Again we observe, that locally no classical solution exists (see Chapter 3). Therefore we choose

$$\begin{aligned}\epsilon =&\frac{\mathcal{D}\sigma_1^+}{\mathcal{D}\sigma_1^+ - \mathcal{D}\sigma_1^-}\\ =&\frac{-\left(\frac{1}{Y_{X/S_1}}r_X X + \frac{1}{Y_{P/S_1}}r_P X\right) - \frac{S_1}{V}\dot{V} + \frac{1}{V}\left(F_A S_{F_A} + F_B S_{F_B}\right)}{-\left(\frac{1}{Y_{X/S_1}}r_X X + \frac{1}{Y_{P/S_1}}r_P X\right)}.\end{aligned} \tag{6.20}$$

and calculate the continuation for the solution. Again we obtain, as is characteristic for switchings due to vanishing concentration (see Figure 3.5) in the model on hand (see Figure 6.6), that the convex combination leads to locally vanishing right hand sides for the corresponding component. Termination criterion is an a priori fixed broth volume.

In the upper part (upper six plots) of Figure (6.6) optimization results of the batch fermentation process are shown, which we obtained using a smoothing of the discontinuities with functions of the form

$$\frac{c_X^4}{const + c_X^4}$$

arising from model changes when concentrations c_X - modeled by usage of the heaviside functions - vanish with an adequately chosen constant *const* for each of the three components.
The lower part (lower six plots) of Figure (6.6) shows the optimization result obtained upon explicitly taking discontinuities into account.
The intention is to maximize the amount of product in the optimization process. With our algorithm, we obtain an amount of $75.961g$ compared to $72.600g$ as found

in Schäfer ([Sch99a]), which amounts to an increase of 4.63%.

The gain coming from explicitely allowing implicit discontinuities can be explained in the sense that parts which contribute to the objective are shrunk when applying the smoothing. Nevertheless it can not be asserted in general that leaving the model unchanged (in the sense that discontinuities are taken into account) causes better optimization results.

However, assuming that the biotechnological model with implicit discontinuities correctly describes the system, then smoothings of the system inherent discontinuities always changes the model qualitatively and so may change the optimization results and even cause them to be incorrect. Our approach thus broadens the applicability of derivative based optimization methods.
We used 15 multiple shooting intervals of equal length, and piecewise constant controls for the three control variables. The running time is 97.1 seconds on a Linux machine (Intel Pentium IV, 2.53 GHz with a cache of 512 KB).

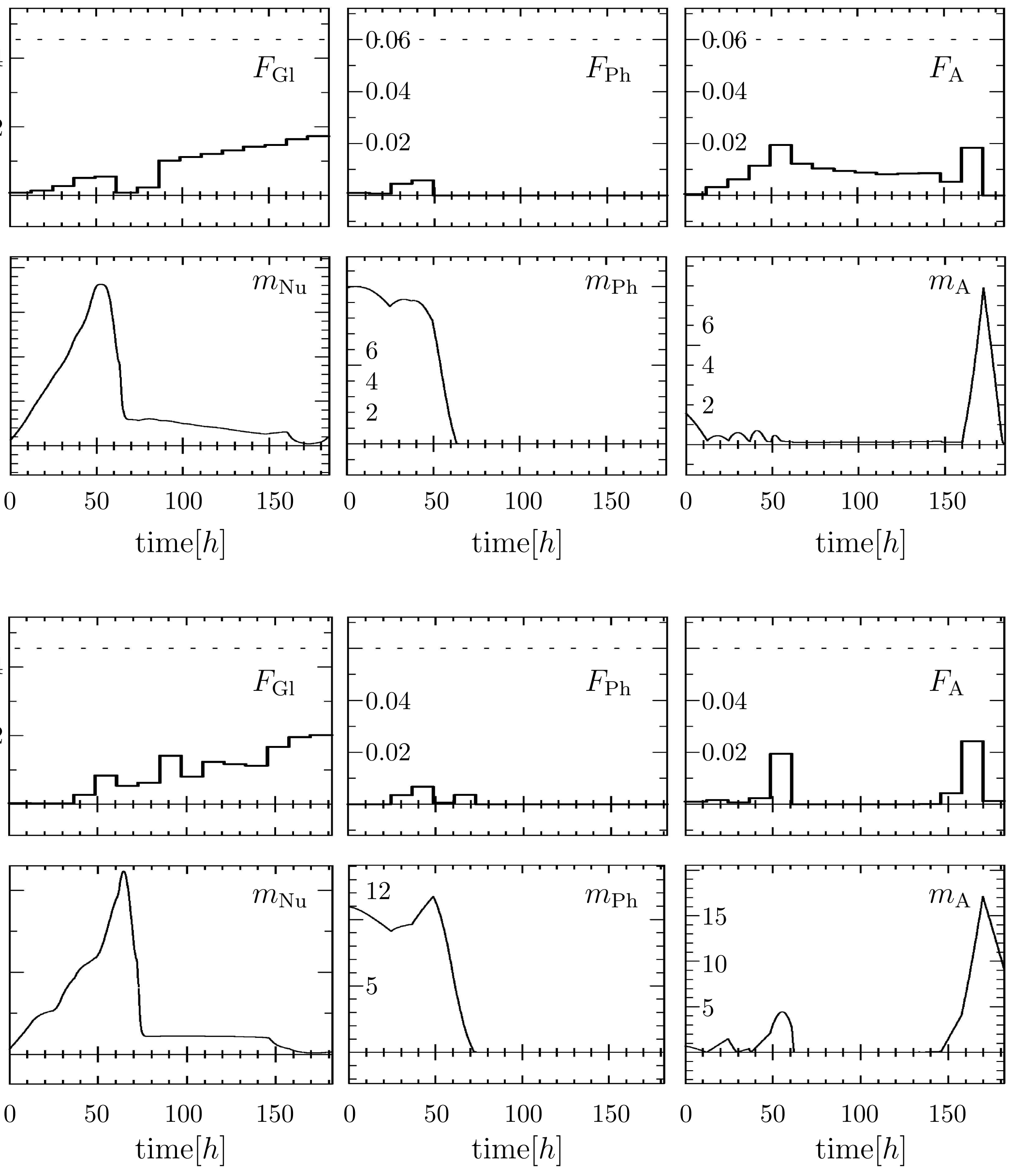

Figure 6.6: Find plotted above the optimal feed profiles (row one and row three) and the three optimal state profiles for the biotechnological process with respect to the maximization of the product mass at the end time. In the upper two rows the results for the smoothed version without implicit discontinuities are presented, the lower two rows show the results for the model explicitely taking implicit discontinuities into account. F_{Gl}, F_{Ph} and F_{A} stand for the glucose, phosphate and ammonia feeds, respectively. m_{Nu}, m_{Ph} and m_{A} represent the masses of the nucleotid mass, the phosphate mass and the ammonia mass.

6.3 Real-time optimization of discontinuous processes

Since real world processes rarely completely coincide with the mathematical model used for its description, a priori off-line optimization of the processes has limited applicability. Real-time optimization techniques have recently attracted increasing interest ([AZ00]). Bock et al. [BDS$^+$00] and [BDLS00] proposed a new approach to deal we real-time processes based on an efficient initial value embedding strategy, that exploits solution information in subsequent optimization problems. Instead of applying a fast off-line algorithm successively solving the arising problems [BBB$^+$01] completely and returning the converged solution to the real-time process they directly send back the first QP solution of a full step exact Hessian SQP algorithm. Under some restriction Diehl in 2002 ([Die02]) was able to give a proof for contractivity of the real-time iterations.

Due to the extraordinary complexity of biotechnological processes the models available are often subject to severe simplifications. Feedback methods like the ones proposed by Diehl et al. ([DBS04]) in real-time can react to deviation between the model proposed states and the measurements.

We again focus on the model of King [Kin94] explained in the last section in some detail.
We simulate perturbation to the process in the following form: After every sampling time step we add a normally distributed error weighted by a factor $weight = 0.04$ with mean 0, variance 1 and deviation 1 to the actual state vector. Additionally, we simulate a model shortcoming by subjoining a drift to a larger concentration in every sampling step. We regard this assumption to be proximate since despite of the stirring (see Figure 6.3) local inhomogeneities are likely to appear.

Every state is assumed to be measurable so that we can perform a measurement after every single sampling interval.

We combined the approach of Diehl et al. ([Die02]) with our BDF integrator for implicit discontinuities. Since the trajectory memory techniques proposed in the thesis on hand can not directly be transferred to real-time applications, we choose an embedded strategy. Because the process model is a priori known we perform an off-line optimization of the process to determine accurate initial values and a

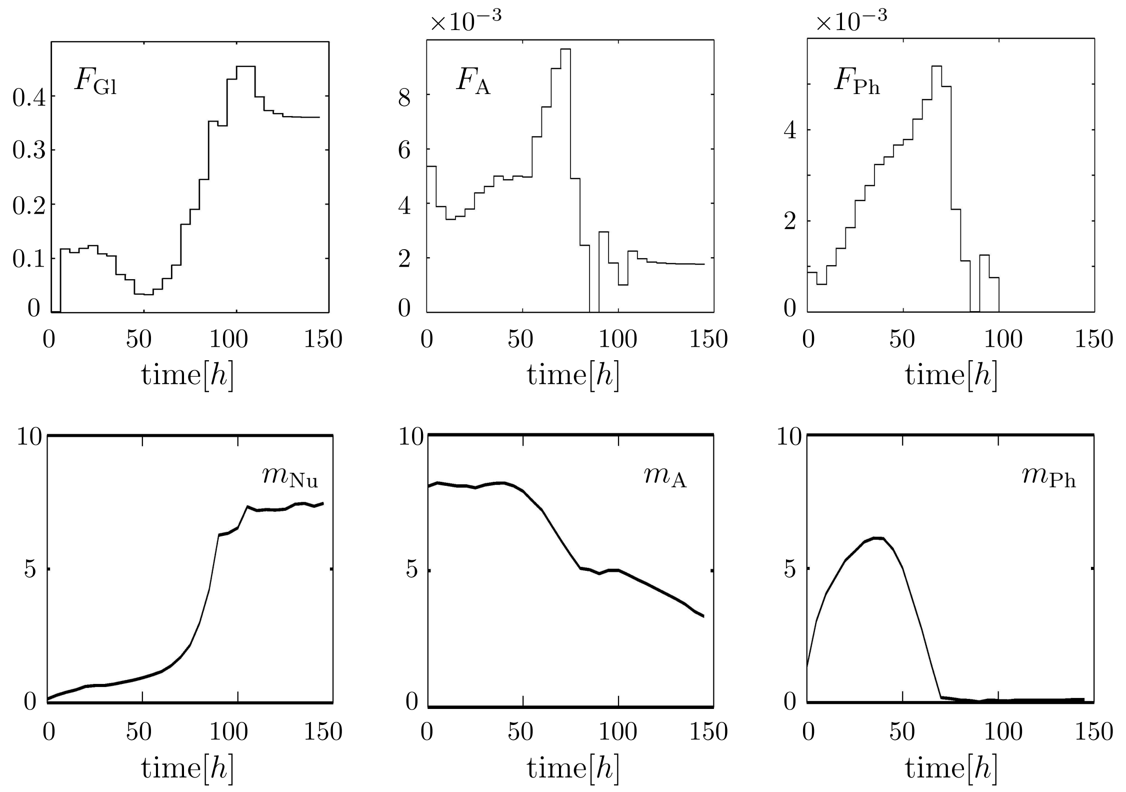

Figure 6.7: In the first row the feed profiles for the feedback optimization on shrinking horizons with implicit discontinuities are presented. The second row shows the adjacent state profiles for the process.

chronology graph $\mathfrak{C}$. For the drift assumption mentioned above we obtain optimal offline trajectories significantly differing from those obtained under the assumption that the model correctly describes the process. Based on these offline trajectories we start the feedback algorithm on shrinking horizons within the digraph $\mathfrak{C}$, which in contrary to the results in the preceding section only shows vanishing ammonia mass.

The controls and three state variables are plotted in Figure 6.7. The introduced perturbation as a normal distributed error and the drift significantly shift the trajectories in comparison to the off-line optimization of the unperturbed case to higher substrate concentrations. Qualitatively the trajectory of the phosphate mass remains almost unchanged.

Since feedback optimization techniques become increasingly important for industrial applications and the accurate modeling of real-world processes often includes implicitly defined discontinuities the combination of our approach comparing different digraphs with the above mentioned feedback techniques. One possibility would be to perform the feedback optimization in parallel on different digraphs.

Chapter 7

Conclusion and Outlook

In this last chapter we give a brief summary of the methods developed in this thesis for optimal control of implicitly discontinuous processes.
Moreover, we indicate possible extensions and how they can be built on our methods and results, including the presentation of concrete ideas and perspectives.

7.1 Conclusion

In the thesis on hand we presented strategies for the efficient treatment of large scale optimal control problems with implicit discontinuities. The main achievements are:

- We implemented a state-of-the-art BDF integrator based on the code DAESOL which can efficiently treat implicit discontinuities in derivatives and states in the model and provides the sensitivity information (see Chapter 3) needed for the application of the solver in direct optimization.

- Since bi-phase dynamic process models are often severely ill-conditioned highly accurate derivate information is needed. Because of the large scale of the models of often more than thousand state variables, efficient derivative generation becomes crucial. We have presented a highly efficient strategy to generate derivatives needed for the sensitivity generation (see Chapter 4) specifically adapted to rigorous multi-tray distillation models (see 6.1) with implicit discontinuities. The sparsity patterns of the right hand sides of the differential algebraic equation are partitioned into those which are invariant with respect to area changes indicated by sign changes of the switching vector and those which vary. Additionally, the similarity of patterns of the right hand side Jacobians in the model parts arising from different trays is exploited (see Chapter 4).

- We have developed an algorithm to survey and treat implicit discontinuities beyond the integrator since implicit discontinuities can only be hidden in the integrator if the digraph $\mathfrak{C}_l$ (see Chapter 5) remains unchanged within the SQP step. We give strategies for the treatment of switching points which coincide with multiple shooting points and of digraph changes due to changing chronology, newly appearing or vanishing switching points.
- Chapter 6 is devoted to applications. We here presented the modeling of a highly complex multi-phase distillation column as well as results for the optimization of the complex distillation process. Moreover, we here applied our algorithms to two biotechnological processes showing the characteristic of inconsistent switching. We also give optimization results and compare them with a smoothed treatment.

7.2 Outlook

The techniques presented to exploit the sparsity patterns in the derivative matrices have shown to be highly efficient for differential algebraic equation (DAE) models. They explicitely take account of the special structure arising from rigorous models of distillation columns.
Since bio-mathematical systems - often described by partial differential equations - attract increasing attention the special adaptation of the linear algebra involved in the DAE solver seems obvious.

In our implementation partitionings of the Jacobians of the right hand sides are performed in advance, what requires an a priori adequate ordering of the model equations. For a wider applicability it can be interesting to perform the graph partitioning automatically.

For systems with a high number of differential variables but comparatively few degrees of freedom in the controls, Schlöder ([Sch88]) proposed a highly efficient approach originally developed for parameter estimation problems transferred to optimal control by Schäfer ([Sch04]). For future use it seems auspicious - with regard to state discritized partial differential equation - to combine this technique with our treatment of implicit discontinuities.

For the DAE system (1.1) it is - based on the trajectory memory implemented in this thesis - possible to exploit the stored trajectory information to evaluate adjoint derivative information for the BDF method using again Bock's Internal Numerical Differentiation (IND). We obtain as recursion formula which can be easily evaluated based on the stored trajectory information (see Chapter 3 and 5). This strategy seems to be promising for applications from process engineering for which comparatatively few components have to differentiated in the solution point.

Our hierarchical memory technique for storing trajectory information can be employed to reuse this information in multiple successive SQP steps (ad hoc implementation available). This idea can be regarded as a natural extension of Bock's IND (see Chapter 3) from the classical sensitivity generation within the integrator to the SQP algorithm. Steps generated on the basis of this freezing technique are available extraordinarily fast since the expensive sensitivity generation is dramatically accelerated at the price of a temporary switch off of the error control.

We presented (see Chapter 5) an algorithm which explicitely takes switching events into account of the optimization algorithm. In the approach we proposed an optimum search is performed within a digraph uniquely given by the active switching structure. For future implementations it will be interesting to fix the active digraph by usage of temporarily active inequality constraints - at the cost of breaking the complete QP structure.

In Chapter 6 we presented the Feedback optimization on shrinking horizons of a batch fermentation process. Since the process itself is comparatatively slow, after every time step the entire offline algorithm could be restarted. For the model describing the rack-in process of the distillation column this surely can't be accomplished. Nevertheless it should be possible to allow several digraphs within which the optimization could be in parallel be performed.

Appendix A

Distillation column

In this Chapter we present the modeling details of the distillation column treated in Chapter 6 (see Figure 6.1). The parameters are subject to nondisclosure, we therefore do not give parameter values for the parameters $P_1, P_2, \ldots, P_{45}$.

In the following we suppress the index i from all variables of the dimension *NTray* for clarity reasons. Index j stands for one of the *NComp* components of the mixture.

A.1 Problem formulation

In the following we briefly describe the components of the distillation column model of Chapter 6 in a tabulated form. *Math.* stands for the mathematical symbol, *Program* gives the in the code used variable name, *Dimension* gives the dimension of the anent variable.

A.1.1 Differential variables

The overall number of differential variables NXD in the distillation column is given by

$$\mathrm{NXD} = (\mathit{NComp} + 1) * \mathit{NTray} + \Sigma_1$$

where Σ_1 stands for the additional variables arising in reboiler and condenser.
The M additional variables in reboiler and condenser describe the integrals over the control expenses like integrated feed, heating and cooling energy etc..

Description	Math.	Program	Dimension
Component mol	n_j	n[j]	$NTray * NComp$
Energy per tray	E	Energy	$NTray$

Table A.1: $(NComp + 1) * NTray$ differential variables which occur on every tray.

A.1.2 Algebraic variables

We collect the algebraic variables in groups as follows

- In- and out-streams (see Table A.2)
- Vapor-liquid equilibrium and miscellaneous (see Table A.3)
- Material variables (see Table A.4)
- Volume constraints (see Table A.5)
- Thermal variables (see Table A.6)
- Geometric variables (see Table A.7)
- Energy variables (see Table A.8)

Description	Math.	Program	Dimension
Liquid product	L^{out}	F_l_out	$NTray$
Temperature liquid product	$t^{l,out}$	t_l_out	$NTray$
Pressure liquid product	$p^{l,out}$	p_l_out	$NTray$
Molar enthalpy liquid product	$h^{l,out}$	h_l_out	$NTray$
Vapor product	V^{out}	F_v_out	$NTray$
Temperature vapor product	$t^{v,out}$	t_v_out	$NTray$
Pressure vapor product	$p^{v,out}$	p_v_out	$NTray$
Molar enthalpy vapor product	$h^{v,out}$	h_v_out	$NTray$
Mole fraction liquid product	$X_j^{l,out}$	x_l_out[j]	$NTray * NComp$
Mole fraction vapor product	$X_j^{v,out}$	x_v_out[j]	$NTray * NComp$

Table A.2: In- and out-streams

Description	Math.	Program	Dimension
Temperature	T	temp	$NTray$
Dew temperature	T^{dew}	TDew	$NTray$
Saturated temperature	T^{sat}	TSat	$NTray$
Pressure	p	press	$NTray$
Saturated pressure component	p_j^{sat}	psat_i[j]	$NTray *3$
k-value	k_j	k[j]	$NTray * NComp$
Vapor fraction	ψ	vap_frac	$NTray$
Thermodynamic vapor fraction	ψ^{therm}	vap_frac_therm	$NTray$

Table A.3: Vapor-liquid equilibrium and miscellaneous

Description	Math.	Program	Dimension
Holdup mol	n^{tot}	n_tot	$NTray$
Holdup mol liq	n^{liq}	n_liq	$NTray$
Holdup mol vap	n^{vap}	n_vap	$NTray$
Mole fraction liquid	X_j	x[j]	$NTray * NComp$
Mole fraction liquid dew	X_j^{dew}	x_dew[j]	$NTray * NComp$
Mole fraction vapor	Y_j	y[j]	$NTray * NComp$
Mole fraction mixture	z_j	z[j]	$NTray * NComp$

Table A.4: Material variables

Description	Math.	Program	Dimension
Component molar enthalpy	ρ^{liq}	rho_liq	$NTray$
Component molar enthalpy	ρ^{vap}	rho_vap	$NTray$
Component molar enthalpy component	ρ_j^{liq}	rho_liq_i[j]	$NTray * NComp$

Table A.5: Volume constraints

Description	Math.	Program	Dimension
Molar enthalpy	h	h	$NTray$
Molar enthalpy liquid	h^l	hl	$NTray$
Molar enthalpy liquid saturated	$h^{l,sat}$	hl_sat	$NTray$
Molar enthalpy vapor	h^v	hv	$NTray$
Molar enthalpy vapor saturated	$h^{v,sat}$	hv_sat	$NTray$
Molar enthalpy liquid component	h^l_j	hl_i[j]	$NTray * NComp$
" saturated	$h^{L,sat}_j$	hl_i_sat[j]	$NTray * NComp$
Molar enthalpy vapor component	h^V_j	hv_i[j]	$NTray * NComp$
" saturated	$h^{V,sat}_j$	hv_i_sat[j]	$NTray * NComp$

Table A.6: Thermal variables

Description	Math.	Program	Dimension
Liquid volume	V^{liq}	V_liq	$NTray$
Vapor volume	V^{vap}	V_vap	$NTray$
Height above weir	Δh^{weir}	delta_h_weir	$NTray$
Liquid level	$level$	level	$NTray$

Table A.7: Geometric variables

Description	Math.	Program	Dimension
Energy of the content	E^{fluid}	E_fluid	$NTray$
Energy of the vessel	E^{vessel}	E_vessel	$NTray$

Table A.8: Energy variables

A.1.3 Parameters and Controls

The number of design parameters is given by

$$\mathrm{NP} = 4\,\mathit{NTray} + \mathit{NComp} + \Sigma_2$$

where Σ_2 subsumes parameters specific for condenser and reboiler.

Description	Math.	Program	Dimension
Mole fraction liquid feed	X_j^{in}	X_in[j]	*NComp*
Molar enthalpy liquid feed	h^{in}	h_in	1
Volume	V	V	*NTray*
Area	A	A	*NTray*
Height of weir	h^{weir}	h_weir	$NTray$
Valve parameter	Y^{valve}	Y_valve	$NTray$
Heat capacity for $p = const$	C_p^{steel}	cp_steel	1
Mass of the tray	m^{steel}	m_steel	1
Outside temperature	T^{ref}	T_referenz	1

Table A.9: Parameters

Description	Math.	Program	Dimension
Heating in reboiler	Q	Q	1
Cooling in condenser	Q^{cool}	Q_cool	1
Reflux ratio	R	ratio	1
In-Feed control	u^{feed}	feed_control	1
Reboiler reflux ratio	R_{reboil}	ratio_reb	1
Vapor outflow control	Y_{cond}^{cont}	Y_cont	1

Table A.10: Control functions

The outstream of the condenser

$$\hat{L} = L + D$$

is composed of the reflux L and the distillate D.
The reflux ratio is given by

$$R = \frac{L}{D}.$$

and thus

$$L = RD = R\frac{R+1}{R+1}D = \frac{R}{R+1}\hat{L}. \tag{A.1}$$

The reflux L is piped to tray $NTray - 1$. The obtained condenser product is $D > 0$. A respective relationship holds for the reboiler (see Figure 6.1).

A.1.3.1 Switching functions

Three switching function $\sigma(t)$ are associated to every tray so that the overall number of switching functions is

$$NSWT = 3 * NTray\,.$$

The switching functions arise from

- Mono-phase bi-phase transition

  $$\psi_i^{therm} - 1 \left\{ \begin{matrix} > \\ \leq \end{matrix} \right\} 0$$

  where ψ_i^{therm} denotes the thermodynamic vapor fraction,

- the outstream of vapor due to overpressure

  $$p_i \left\{ \begin{matrix} > \\ \leq \end{matrix} \right\} 1,$$

  with the pressure p_i on tray i and

- weir overflows

  $$\Delta h_i^{weir} \left\{ \begin{matrix} > \\ \leq \end{matrix} \right\} 0$$

  caused be the amount of liquid holdup which exceeds the weir height on the respective tray.

A.1.4 Differential equations

In the following subsection we list the differential equations of the DAE process model.

Component balance:
The $NTray * NComp$ component balances are given by

$$\begin{aligned}\dot{n}_{ij} =& - X^{l,out}_{i,j} L^{out}_{j} - Y^{out}_{i,j} V^{out}_{j} \\ &+ X^{l,out}_{j+1,j} L^{out}_{j+1} \Theta(n_{\text{tray}} - j - 1) \\ &+ Y^{out}_{j-1,j} V^{v,out}_{j-1} \Theta(j-1) \\ &+ \ \delta_{j,NFeed}\, X^{in}_{i} L^{in}.\end{aligned} \tag{A.2}$$

Energy balance:
We use for the $NTray$ energy balances

$$\begin{aligned}\dot{E}_j =& - h^{L,out}_{j} L^{out}_{j} - h^{V,out}_{j} V^{out}_{j} \\ &+ \ h^{V,out}_{j-1} V^{out}_{j-1} + h^{L,out}_{j+1} L^{out}_{j+1} \\ &+ \ \delta_{j,NFeed}\, h^{L,in} L^{in} \\ &+ \ \delta_{j,NTray}\, Q^{cool} \\ &+ \ \delta_{j,1}\, Q.\end{aligned} \tag{A.3}$$

A.1.5 Algebraic equations

To adequately model the dynamics of the liquid-vapor mixture which explicitely allows a vanishing liquid phase we formulated the following algebraic equations:

Mass balance:
The $NTray + NComp * NTray + NTray + (NComp + 3) * NTray$ equations have the following form (depending on the valid sign structure of the switching vector)

$$n^{tot}_{j} = \sum_i n_{i,j} \tag{A.4}$$

$$n_{i,j} = Y_{i,j} n^{vap}_{j} + X_{i,j} n^{liq}_{j} \tag{A.5}$$

$$n^{tot}_{j} h_j = n^{vap}_{j} h^{v}_{j} + n^{liq}_{j} h^{l}_{j}. \tag{A.6}$$

For $\psi_j^{therm} < 1$ we obtain:

$$\begin{aligned} n_j^{vap} &= \psi_j \cdot n_j^{tot} \ , && \text{(A.7)} \\ \sum_i X_{i,j} &= 1 \ , && \text{(A.8)} \\ Y_{i,j} &= k_{i,j} X_{i,j} \ , && \text{(A.9)} \\ \sum_i Y_{i,j} &= 1 \ . && \text{(A.10)} \end{aligned}$$

For overheated gas $\psi_i^{therm} \geq 1$ we instead assume:

$$\begin{aligned} n_j^{vap} &= n_j^{tot} \ , && \text{(A.11)} \\ n_j^{liq} &= 0 \ , && \text{(A.12)} \\ X_{i,j} &= X_{i,j}^{dew} \ , && \text{(A.13)} \\ \psi_j &= 1 \ . && \text{(A.14)} \end{aligned}$$

Energy balance:
The $3 * NTray$ have the form:

$$\begin{aligned} E_j &= E_j^{vessel} + E_j^{fluid} && \text{(A.15)} \\ E_j^{vessel} &= C_p^{steel} m^{steel} (T_j - T^{referenz}) && \text{(A.16)} \\ E_j^{fluid} &= n_j^{tot} h_j - 10^{-4} p_j V_j . && \text{(A.17)} \end{aligned}$$

Thermodynamic vapor fraction:
On every tray the thermodynamic vapor fraction is given by:

$$\psi_j^{therm} \cdot (h_j^{v,sat} - h_j^{l,sat}) = h_j - h_j^{l,sat} \qquad \text{(A.18)}$$

Liquid product:
The $5 * NTray + NComp * NTray + NTray$ liquid product equations are of the

following form:

$$\begin{aligned} A\ level_j &= V_j^{liq} && \text{(A.19)} \\ \Delta h_j^{weir} &= level_j - h_j^{weir} && \text{(A.20)} \\ t_j^{l,out} &= T_j && \text{(A.21)} \\ p_j^{l,out} &= p_j && \text{(A.22)} \\ X_{i,j}^{l,out} &= X_{i,j} && \text{(A.23)} \\ h_j^{l,out} &= h_j^l. && \text{(A.24)} \end{aligned}$$

For $\Delta h_j^{weir} > 0$:

$$L_j^{out} = 10^3 |\Delta h_j^{weir}|^{\frac{3}{2}}, \tag{A.25}$$

For $\Delta h_j^{weir} \leq 0$:

$$L_j^{out} = 0. \tag{A.26}$$

Vapor product:
The $3 * NTray + NComp * NTray + NTray$ equations are of the form

$$\begin{aligned} t_j^{v,out} &= T_j, && \text{(A.27)} \\ p_j^{v,out} &= p_j, && \text{(A.28)} \\ h_j^{v,out} &= h_j^v, && \text{(A.29)} \\ X_{i,j}^{v,out} &= Y_{i,j}. && \text{(A.30)} \end{aligned}$$

For overpressure $p_j > 1$ in tray i we obtain

$$V_j^{out} = 10 Y_j^{valve} (p_j - 1), \tag{A.31}$$

otherwise, for pressure $p_j \leq 1$

$$V_j^{out} = 0. \tag{A.32}$$

and for the condenser:

For overpressure $p_{cond} > 1$ in the condenser we obtain

$$V_{cond}^{out} = Y_{cond}^{cont} (p_{cond} - 1), \tag{A.33}$$

otherwise, for pressure $p_{cond} \leq 1$:

$$V_{cond}^{out} = 0. \tag{A.34}$$

Specific enthalpy liquid:
The $NTray + NComp * NTray$ equations are of the form

$$h_j^l = \sum_j X_{i,j} h_{i,j}^l \tag{A.35}$$

and for $\psi_j^{therm} < 1$:

$$\begin{aligned}
h_{i,1}^l &= P_1 - P_2 + P_3(T_j - 25) && \text{(A.36)}\\
h_{i,2}^l &= P_4 - P_5 + P_6(T_j - 25) && \text{(A.37)}\\
h_{i,3}^l &= P_7 - P_8 + P_9(T_j - 25) && \text{(A.38)}\\
h_{i,4}^l &= P_{10} - P_8 + P_9(T_j - 25) && \text{(A.39)}
\end{aligned}$$

otherwise for $\psi_j^{therm} \geq 1$:

$$\begin{aligned}
h_{i,1}^l &= P_1 - P_2 + P_3(T_j^{dew} - 25) && \text{(A.40)}\\
h_{i,2}^l &= P_4 - P_5 + P_6(T_j^{dew} - 25) && \text{(A.41)}\\
h_{i,3}^l &= P_7 - P_8 + P_9(T_j^{dew} - 25) && \text{(A.42)}\\
h_{i,4}^l &= P_{10} - P_8 + P_9(T_j^{dew} - 25) && \text{(A.43)}
\end{aligned}$$

with the parameters $P_1, P_2, \ldots, P_{10}$.

Specific enthalpy vapor:
The $NTray + NComp * NTray$ equations for the vapor enthalpies are given by

$$h_j^v = \sum_j Y_{i,j} h_{i,j}^v, \tag{A.44}$$

$$\begin{aligned}
h_{i,1}^v &= P_1 + P_{11}(T_j - 25), && \text{(A.45)}\\
h_{i,2}^v &= P_4 + P_{12}(T_j - 25), && \text{(A.46)}\\
h_{i,3}^v &= P_7 + P_{13}(T_j - 25), && \text{(A.47)}\\
h_{i,4}^v &= P_{14}(T_j - 25), && \text{(A.48)}
\end{aligned}$$

with process parameters P_k.

k-values:
For the $NComp * NTray + 3 * NTray$ k-value equations, we obtain

$$k_{i,1}\, p_j = p^{sat}_{i,1}, \tag{A.49}$$
$$k_{i,2}\, p_j = p^{sat}_{i,2}, \tag{A.50}$$
$$k_{i,3}\, p_j = p^{sat}_{i,3}, \tag{A.51}$$
$$k_{i,4} = 1000\, k_{i,1} \tag{A.52}$$

and

$$\begin{aligned} p^{sat}_{i,1} &= \Big(\exp(P_{15} + \frac{P_{16}}{\hat{T}_j + 273.15} + (P_{17}) \log|\hat{T}_j + 273.15| \\ &\quad + P_{18}(\hat{T}_j + 273.15)^2 10^{-6})\Big) 10^{-5} \end{aligned} \tag{A.53}$$

$$\begin{aligned} p^{sat}_{i,2} &= \Big(\exp(P_{19} + \frac{P_{20}}{\hat{T}_j + 273.15} + (P_{21}) * \log(|273.15 + \hat{T}_j|) \\ &\quad + (P_{22}) * (\hat{T}_j + 273.15)^1)\Big) 10^{-5} \end{aligned} \tag{A.54}$$

$$\begin{aligned} p^{sat}_{i,3} &= \Big(\exp(P_{23} + \frac{P_{24}}{\hat{T}_j + 273.15} + (P_{25}) * \log(|273.15 + \hat{T}_j|) \\ &\quad + P_{26} * (\hat{T}_j + 273.15)^2 10^{-6})\Big) 10^{-5} \end{aligned} \tag{A.55}$$

with

$$\hat{T}_j = \begin{cases} T_j & \psi^{therm}_j > 1 \\ T^{dew}_j & \psi^{therm}_j \le 1 \end{cases} \tag{A.56}$$

and the parameters P_k.

Specific density liquid phase:
The $NTray + NComp * NTray$ equations are given by

$$\rho^{liq}_j = \sum_j \rho^{liq}_{ij} \tag{A.57}$$

$$\rho_{i,1}^{liq} = P_{27} / \left(P_{28}^{1+\left|\left(1-\frac{\hat{T}_j+273.15}{P_{29}}\right)\right|^{P_{30}}} \right), \tag{A.58}$$

$$\rho_{i,2}^{liq} = P_{31} / \left(P_{32}^{1+\left|\left(1-\frac{\hat{T}_j+273.15}{P_{33}}\right)\right|^{P_{34}}} \right), \tag{A.59}$$

$$\rho_{i,3}^{liq} = P_{35} / \left(P_{36}^{1+\left|\left(1-\frac{\hat{T}_j+273.15}{P_{37}}\right)\right|^{P_{38}}} \right), \tag{A.60}$$

$$\rho_{i,4}^{liq} = 1.0 \tag{A.61}$$

with

$$\hat{T}_j = \begin{cases} T_j & \psi_j^{therm} > 1 \\ T_j^{dew} & \psi_j^{therm} \leq 1. \end{cases} \tag{A.62}$$

Specific density vapor phase:
For the $NTray$ equation we formulate

$$\rho_j^{vap} \left(P_{39} \left(T_j + 273.15 \right) \right) = 100\, p_j. \tag{A.63}$$

Bubble temperature liquid:
The relation between the $NTray$ boiling point temperatures and the individual pressures on the trays is as follows

$$\begin{aligned} p_j =& (z_{i,1} + 1000\, z_{i,4}) \Big(\exp(P_{15} - \frac{P_{40}}{T_j^{sat} + 273.15} \\ &- P_{41} \log |T_j^{sat} + 273.15| + (P_{18}\, 10^{-6}) \, (T_j^{sat} + 273.15)^2) \Big) \, 10^{-5} \\ &+ z_{i,2} \Big(\exp(P_{19} - \frac{P_{42}}{T_j^{sat} + 273.15} \\ &- P_{44} \log |T_j^{sat} + 273.15| + (P_{22}) \, (T_j^{sat} + 273.15)) \Big) \, 10^{-5} \\ &+ z_{i,3} \Big(\exp(P_{23} - \frac{P_{43}}{T_j^{sat} + 273.15} \\ &- P_{45} \log |T_j^{sat} + 273.15| + (P_{26}\, 10^{-6}) \, (T_j^{sat} + 273.15)^2) \Big) \, 10^{-5}. \end{aligned} \tag{A.64}$$

Dew point temperature vapor:
For the $NComp * NTray + NTray$ dew point relationships we obtain

$$\begin{aligned} p_j\ z_{i,1} &= X_{i,1}^{dew} \Big(\exp(P_{15} - \frac{P_{40}}{T_j^{dew} + 273.15} - P_{41} \log(|T_j^{dew} + 273.15|) \\ &+ P_{18}\ 10^{-6} (T_j^{dew} + 273.15)^2) \Big) 10^{-5}, \qquad (A.65) \\ p_j\ z_{i,2} &= X_{i,2}^{dew} \Big(\exp(P_{19} - \frac{P_{42}}{T_j^{dew} + 273.15} - P_{44} \log(|T_j^{dew} + 273.15|) \\ &+ P_{22} (T_j^{dew} + 273.15)) \Big) 10^{-5}, \qquad (A.66) \\ p_j\ z_{i,3} &= X_{i,3}^{dew} \Big(\exp(P_{23} - \frac{P_{43}}{T_j^{dew} + 273.15} - P_{45} \log(|T_j^{dew} + 273.15|) \\ &+ P_{26}\ 10^{-6} (T_j^{dew} + 273.15)^2) \Big) 10^{-5}, \qquad (A.67) \\ p_j\ z_{i,4} &= X_{i,4}^{dew} \Big(\exp(P_{15} - \frac{P_{40}}{T_j^{dew} + 273.15} - P_{41} \log(|T_j^{dew} + 273.15|) \\ &+ P_{18}\ 10^{-6} (T_j^{dew} + 273.15)^2) \Big) 10^{-5}, \qquad (A.68) \end{aligned}$$

and

$$\sum_j X_{i,j}^{dew} = 1. \qquad (A.69)$$

Specific enthalpy saturated vapor:
The $NComp * NTray + NTray$ equations for specific enthalpies in the saturated vapor are

$$\begin{aligned} h_{i,1}^{v,sat} &= P_1 + P_{11}\ (T_j^{dew} - 25), \qquad (A.70) \\ h_{i,2}^{v,sat} &= P_4 + P_{12}\ (T_j^{dew} - 25), \qquad (A.71) \\ h_{i,3}^{v,sat} &= P_7 + P_{13}\ (T_j^{dew} - 25), \qquad (A.72) \\ h_{i,4}^{v,sat} &= P_{14}\ (T_j^{dew} - 25), \qquad (A.73) \end{aligned}$$

$$h_j^{v,sat} = \sum_j z_{i,j}\ h_{i,j}^{v,sat}. \qquad (A.74)$$

Specific enthalpy saturated liquid:
The $NComp * NTray + NTray$ equations for specific enthalpies in the saturated liquid are

$$\begin{aligned}
h_{i,1}^{l,sat} &= P_1 - P_2 + P_3\ (T_j^{sat} - 25), && \text{(A.75)}\\
h_{i,2}^{l,sat} &= P_4 - P_5 + P_6\ (T_j^{sat} - 25), && \text{(A.76)}\\
h_{i,3}^{l,sat} &= P_7 - P_8 + P_9\ (T_j^{sat} - 25), && \text{(A.77)}\\
h_{i,4}^{l,sat} &= P_{10} - P_8 + P_9\ (T_j^{sat} - 25), && \text{(A.78)}
\end{aligned}$$

$$h_j^{l,sat} = \sum_j z_{i,j}\ h_{i,j}^{l,sat}. \qquad \text{(A.79)}$$

Appendix B

Practical usage of the automatic differentiation in C/C++ in MUSCOD-II

In this appendix we briefly explain how automatic differentiation can be used in our tools.

For the generation of derivatives we rely on Griewank's tool *ADOL-C* ([GJM$^+$99]), the sparsity compression techniques (see Chapter 4) are automatically applied based on the bit patterns obtained from the right hand side of the differential algebraic equation. It is possible to choose between the different sparsity pattern exploitation techniques presented in Chapter 4.

B.1 Usage of Automatic differentiation in MUSCOD-II

To provide derivative information of the compressed Jacobian or to calculated directional derivatives we use the the package *ADOL-C* ([GJM$^+$99]).
The package utilizes operator overloading in *C++* (even if the source files are implemented in *C*). Overloading does not generate intermediate source code and only requires minor changes to the users evaluation program. The key ingredient of AD by overloading is the concept of an *active variable*. All variables that are considered differentiable quantities must be declared as active types called `adouble` in *ADOL-C*. All calculations involving active variables must occur between the void function calls `trace_on(tag,keep)` and `trace\_off(file)` marking the active section of

the code.

B.1.1 A tool for the generation of derivative files

We provide a Perl script that takes care of the creation of the derivative functions which *ADOL-C* needs.

The script uses the model source file as input. It looks for the three model functions of which derivatives are needed - usually named `iffcn`, `igfcn`, `iswitch` - and writes them to temporary files. These files are then altered to create the required derivative files `function_variable.cc`, where `function` stands for `iffcn`, `igfcn` or `iswitch` and `variable` for the type of variables with respect to which we are differentiating.

B.1.1.1 Using the script

Opon each call of `extract.pl`, you have to tell the script which file you want to process. You can either do this by passing the name as a command line argument, e.g. `extract.pl`, `mymodel.c`, or by changing the line

```
FILENAME_defines = "mymodel.c";
```

inside the script. Experienced users may modify the behavior, too. For example you may suppress the creation of derivatives for the function describing the algebraic equations by setting the value of `gfcn_create` to zero at the top of the script. For detailed information, please read the comments in the script.

In this context we to some extent stick to the general XML-Standard by using tags indicating the beginning and the ending of certain passages. These tags are C comments with certain keywords. In detail:

- You may want to use preprocessor directives like includes or defines in your C source code which are needed by the model functions (usually `iffcn`, `igfcn`, `iswitch`) and so also in the appropriate derivative files. Everything in between the tags

  ```
  /* begin_preprocessor */  ...  /* end_preprocessor */
  ```

  will be copied to the derivative files headers. Make sure all directives are between these tags, otherwise you may obtain errors when compiling or linking the code.

- Every function (`iffcn, igfcn, iswitch`) of the problem has to be encapsulated by

```
/* begin_ffcn */ ... /* end_ffcn */
/* begin_gfcn */ ... /* end_gfcn */
/* begin_switch */ ... /* end_switch */    ,
```

  respectively.

- The block of equations in each of these model functions has to be encapsulated by

```
/* begin_equations */ ... /* end_equations */
```

- You may want to use your own names for the variables you use, to keep the code in a readable and understandable form. For example you may define and set the local variable

```
double temperature;
temperature = xd[0];
```

  and use `temperature` instead of `xd[0]` later on in the code. To handle this implicit formulation the user has to indicate which local variables are differential variables, derivative of a differential variables, algebraic variables, a parameters or a controls by writing the *definitions* of the variables between the fitting tags:

```
/* begin_differential */ ... /* end_differential */
/* begin_diffdiff */ ... /* end_diffdiff */
/* begin_algebraic */ ... /* end_algebraic */
/* begin_parameters */ ... /* end_parameters */
/* begin_controls */ ... /* end_controls */
```

  This has to be done inside each of the three model functions.

List of Figures

Bibliography

[AdP02] P. W. Atkins and J. de Paula. *Atkins' Physical Chemistry.* Oxford University Press, Oxford, 2002.

[AMB93] B.M. Averick, J.J. Moré, and C.H. Bischof. Computing large sparse Jacobian matrices using automatic differentiation. *Preprint MCS-P348-0193, Mathematics and Computer Science Division*, 1993.

[AZ00] F. Allgöwer and A. Zheng, editors. *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory.* Birkhäuser, Basel, 2000.

[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`.

[Bau00] J. Bausa. *Dynamische Optimierung energie- und verfahrenstechnischer Prozesse*, volume 667 of *Fortschr.-Ber. VDI Reihe 3, Verfahrenstechnik.* VDI Verlag, Düsseldorf, 2000.

[Bau01] A. Baud. *Mont Blanc et Aiguilles Rouges à ski.* Nevicata, 2001.

[BBB$^+$01] T. Binder, L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J.P. Schlöder, and O.v. Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Grötschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 295–340. Springer, 2001. download at: http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/Preprint-01-15.html.

[BBS99] I. Bauer, H.G. Bock, and J.P. Schlöder. DAESOL – a BDF-code for the numerical solution of differential algebraic equations. Internal report, IWR, SFB 359, University of Heidelberg, 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.

[BCKM94] C.H. Bischof, A. Carle, P.M. Khademi, and A. Mauer. The ADIFOR 2.0 system for the automatic differentiation of Fortran 77 programs. Preprint MCS–P481–1194, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1994. To appear in IEEE Computational Science & Engineering.

[BCKM96] C.H. Bishof, A. Carle, P. Khademi, and A. Mauer. Adifor 2.0: Automatic differentiation of Fortran 77 programs. *IEEE Computational Science & Engineering*, 18(3):18–32, 1996.

[BDLS00] H.G. Bock, M. Diehl, D.B. Leineweber, and J.P. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 246–267, Basel, 2000. Birkhäuser.

[BDS$^+$00] H.G. Bock, M. Diehl, J.P. Schlöder, F. Allgöwer, R. Findeisen, and Z. Nagy. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. In *ADCHEM2000 - International Symposium on Advanced Control of Chemical Processes*, volume 2, pages 695–703, Pisa, 2000.

[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.

[BGS88] R.W. Brankin, I. Gladwell, and L.F. Shampine. Starting BDF and Adams codes at optimal order. *J. Comput. Appl. Math.*, 21(3):357–368, 1988.

[BHS02] M. Buss, M. Hardt, and O.v. Stryck. Numerical solution of hybrid optimal control problems with applications in robotics. In *Proc. 15th*

IFAC World Congress on Automatic Control, Barcelona, 2002. Elsevier Science.

[Bie84] L.T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Comput. Chem. Engng.*, 8:243–248, 1984.

[Bis04] C.H. Bischof. `www.autodiff.org`, 2004.

[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, Bonn, 1986.

[BNPS91] R. Bulirsch, E. Nerz, H.J. Pesch, and O.v. Stryk. Combining direct and indirect methods in nonlinear optimal control: Range maximization of a hang glider. Technical Report 313, Schwerpunktprogramm der Deutschen Forschungsgemeinschaft, Technical University Munich, 1991.

[Boc74] H.G. Bock. Numerische Optimierung zustandsbeschränkter parameterabhängiger Prozesse. Master's thesis, University of Cologne, Cologne, 1974.

[Boc77] H.G. Bock. Zur numerischen Behandlung zustandsbeschränkter Steuerungsprobleme mit Mehrzielmethode und Homotopieverfahren. *Z. Angew. Math. Mech.*, 57:T266–T268, 1977.

[Boc78] H.G. Bock. Numerische Berechnung zustandsbeschränkter optimaler Steuerungen mit der Mehrzielmethode. Technical report, Carl-Cranz-Gesellschaft, Report, 1978.

[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*. Springer, Heidelberg, 1981.

[Boc83] H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, Boston, 1983.

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

[BP70] A. Bjørk and V. Pereyra. Solutions of Vandermonde systems of equations. *Math. Comp.*, 24:893–903, 1970.

[BP84] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proc. 9th IFAC World Congress Budapest*. Pergamon Press, 1984.

[BP02] U. Brandt-Pollmann. Optimization of discontinuous dynamical models. Technical report, First International Conference on Optimization and Software, Hangzhou, China, December 2002.

[BP03] U. Brandt-Pollmann. Nicht-Standard Probleme der optimalen Steuerung in Chemie und Biotechnologie. Technical report, DECHEMA Jahrestagung der Biotechnologen, München, April 2003.

[BPDL$^+$03] U. Brandt-Pollmann, M. Diehl, D. Leineweber, S. Sager, and A. Schäfer. MUSCOD-II users' manual, 2nd ed. 2003.

[BPDLP04] U. Brandt-Pollmann, M. Diehl, D. Lebiedz, and A. Potschka. A parallel optimal control algorithm based on direct multiple shooting for differential algebraic equations. 2004.

[BPSB$^+$04] U. Brandt-Pollmann, S. Sager, H. G. Bock, M. Diehl, D. Lebiedz, and J. P. Schlöder. Generating start values for chemical engineering optimal control problems by polynomial interpolation. *submitted to AIChE*, 2004.

[BPSDL04] U. Brandt-Pollmann, S. Sager, M. Diehl, and D. Lebiedz. A fast method for optimal control using automatic differentiation for sensitivity generation. *submitted to Opt. Meth. and Software*, 2004.

[Bro67] C.G. Broyden. Quasi-Newton methods and their application to function minimization. *Maths. Comp.*, 21:368–381, 1967.

[BS81] H.G. Bock and J.P. Schlöder. *Numerical solution of retarded differential equations with state-dependent time lags*, volume 61. Z. angew. Math. Mech., 1981.

[BT00] M. Benzi and M. Tuma. Orderings for factorized sparse approximate inverse preconditioners. *SIAM Journal on Scientific Computing*, 21(5):1851–1868, 2000.

[Bul71] R. Bulirsch. Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung. Technical report, Carl-Cranz-Gesellschaft, Report, 1971.

[CC82] T.F. Coleman and A.R. Conn. Mathematical programming. *Nonlinear programming via an exact penalty function: Asymptotic Analysis*, 24:123–136, 1982.

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

[Cha79] R. M. Chamberlain. Some examples of cycling in variable metric methods for constrained minimization. *Math. Prog.*, 16:378–383, 1979.

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

[CLPP82] R.M. Chamberlain, C. Lemaréchal, H.C. Pedersen, and M.J.D. Powell. The watchdog technique for forcing convergence in algorithms for constrained optimization. *Math. Prog. Study*, 16:1–17, 1982.

[CM84] T.F. Coleman and J.J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20:187–209, 1984.

[CPR74] A.R. Curtis, M.J.D. Powell, and J.K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974.

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

[CV98] T.F. Coleman and A. Verma. The efficient computation of sparse Jacobian matrices. *SIAM J. Sci. Comput.*, 19(4):1210–1233, 1998.

[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.

[Dav68] W.C. Davidon. Variance algorithms for minimization. *Computer J.*, 10:406–410, 1968.

[Dav91] W.C. Davidon. Variable metric method for minimization. *SIAM J. Optim.*, 1:1–17, 1991.

[DBS04] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Contr. Optim.*, accepted, 2004.

[Deu83] P. Deuflhard. Order and stepsize control in extrapolation methods. *Numer. Math.*, 41:399–422, 1983.

[DHZ87] P. Deuflhard, E. Hairer, and J. Zugck. One step and extrapolation methods for differential-algebraic systems. *Numer. Math.*, 51:501–516, 1987.

[Die02] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschr.-Ber. VDI Reihe 8, Meß-, Steuerungs und Regelungstechnik.* VDI Verlag, Düsseldorf, 2002.

[DS83] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Computational Mathematics. Prentice Hall, 1983.

[DS89] J.E. Dennis and R.B. Schnabel. A view of unconstrained optimization. In G. L. Nemhauser, A. H. G. Rinnooy, and M. J. Todd, editors, *Handbooks in Operations Research and Management Science*, volume 1 (Optimization), Amsterdam, 1989. Elsevier.

[DV84] K. Dekker and J.G. Verwer. *Stability of Runge-Kutta methods for stiff non-linear differential equations.* North Holland Publ. Co., Amsterdam, 1984.

[EFS02] S. Engell, G. Frehse, and E. Schnieder, editors. *Modeling, Analysis and Design of Hybrid Systems.* Number 279 in Lecture Notes in Control and Information Sciences (LNCIS). Springer, 2002.

[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 I mit BDF Verfahren. Master's thesis, University of Bonn, Bonn, 1987.

[Eic92] E. Eich. *Projizierte Mehrschrittverfahren zur numerischen Lösung von Bewegungsgleichungen technischer Mehrkörpersysteme mit Zwangsbedingungen und Unstetigkeiten*, volume 109 of *Fortschr.-Ber. VDI Reihe 18, Mechanik/Bruchmechanik*. VDI Verlag, Düsseldorf, 1992.

[EN00] R. Ehrig and U. Nowak. Limex. `http://elib.zib.de/pub/elib/codelib/limex/`, 2000.

[Fee99] W. F. Feehery. Dynamic optimization with path constraints. *Ind. Eng. Chem. Res.*, 38:2350–2363, 1999.

[Fil64] A.F. Filippov. Differential equations with discontinuous right hand side. *AMS Transl.*, 42:199–231, 1964.

[Fle71] R. Fletcher. A general quadratic programming algorithm. *J. Inst. Math. Appl.*, 7:76–91, 1971.

[Fle87] R. Fletcher. *Practical Methods of Optimization.* Wiley, Chichester, 2nd edition, 1987.

[Fle95] R. Fletcher. An optimal positive definite update for sparse Hessian matrices. *SIAM J. Optim.*, 5:192–218, 1995.

[FM90] A.V. Fiacco and G.P. McCormick. Nonlinear programming: Sequential unconstrained minimization techniques. *SIAM publications*, 1990.

[Fre98] C. Freddebeul. A-BDF: A generalization of the backward differentiation formula. *SIAM J. Numer. Anal.*, 35(5):1917–1938, 1998.

[GB94] J.V. Gallitzendörfer and H.G. Bock. Parallel algorithms for optimization boundary value problems in DAE. In H. Langendörfer, editor, *Praxisorientierte Parallelverarbeitung.* Hanser, München, 1994.

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

[Gea88] C.W. Gear. Differential-algebraic 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. Updated version of the paper published in *ACM Trans. Math. Software* 22, 1996, 131–167.

[GL96] G. Golub and C. van Loan, editors. *Matrix computations.* The Johns Hopkins University Press, London, 3rd edition, 1996.

[GO84] C.W. Gear and O. Osterby. Solving ordinary differential equations with discontinuities. *ACM Trans. Math. Softw.*, 10(1):23–44, 1984.

[GP83] R.-P. Ge and M.J.D. Powell. The convergence of variable metric matrices in unconstrained optimization. *Mathematical Programming*, 27:123–143, 1983.

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

[Gri00] A. Griewank. *Evaluating Derivatives.* Frontiers in applied mathematics. Society for Industrial and Applied Mathematics (SIAM), 2000.

[GUG96] U. Geitner, J. Utke, and A. Griewank. Automatic computation of sparse Jacobians by applying the method of Newsam and Ramsdell. In M. Berz et al. eds., editor, *Computational Differentiation, Proceedings of the Second International Workshop*, pages 161–172, Philadelphia, 1996. SIAM.

[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.

[Han94] M. Hanke. Asymptotic expansions for regularization methods of linear fully implicit differential-algebraic equations, 1994.

[HBSC97] P. Hovland, C. Bishof, D. Spiegelman, and M. Casella. Efficient derivative codes through automatic differentation and interface contraction: An application in biostatistics. *SIAM J. Sci. Comp.*, 18(4):1056–1066, 1997.

[Hin80] A.C. Hindmarsh. LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM-SIGNUM Newsletter*, 15:10–11, 1980.

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

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

[HS02] S. Hossain and T. Steihaug. Graph coloring in the estimation of mathematical derivatives, 2002.

[Ins04] Texas Instruments. *Derive*. Texas Instruments, Inc., Housten, Texas, 6.0 edition, 2004.

[Int04] Intel. Ia-32 Intel Architecture Software Developer's Manual, 2004.

[Kar39] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[Kie98] A. Kienle. Reduced models for multicomponent separation processes using nonlinear wave propagation theory. In *13th International Congress of Chemical and Process Engineering*, Praha, Czech Republic, 1998.

[Kin94] R. King. *Mathematische Modelle mycelförmig wachsender Mikroorganismen*, volume 103 of *Fortschrittberichte VDI, Reihe 17*. VDI-Verlag, Düsseldorf, 1994.

[Kis92] H. Z. Kister. *Distillation Design*. McGraw-Hill, New York, 1992.

[Köh02] J. Köhler. Optimierung biokinetischer Systeme. Technical report, BASF AG, 2002.

[Kör02] S. Körkel. *Numerische Methoden für Optimale Versuchsplanungsprobleme bei nichtlinearen DAE-Modellen*. PhD thesis, University of Heidelberg, Heidelberg, 2002.

[Kra04] Ch. Kraus. *Efficient Object–Oriented Modelling, Simulation and Parameter Estimation for Biomechanical Problems.* PhD thesis, University of Heidelberg, Heidelberg, 2004.

[Kro79] F.T. Krogh. Recurrence relations for computing with modified divided differences. *Math. Comp.*, 33(148):1265–1271, 1979.

[KT51] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, 1951. University of California Press.

[Küh02] A. Kühn. Modellierung und Optimierung einer Aminosäurefermentation. Master's thesis, University of Freiberg, 2002.

[KWB$^+$95] Th. Kendlbacher, W. Waldraff, G. Breuel, R. Biener, R. King, and E. D. Gilles. Application of model-predictive control to fermentations with streptomyces. In R. D. Schmid, editor, *Biochemical Engineering 3 - Stuttgart*, Stuttgart, 1995. E. Kurz & Co.

[LBBS03] D.B. Leineweber, I. Bauer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: Theoretical aspects. *Comp. & Chem. Eng.*, 27:157–166, 2003.

[LBP03] D. Lebiedz and U. Brandt-Pollmann. Manipulation of Self-Aggregation Patterns and Waves in a Reaction-Diffusion System by Optimal Boundary Control Strategies. *Phys. Rev. Lett.*, 91(20), 2003.

[LBP04a] D. Lebiedz and U. Brandt-Pollmann. Dynamic control and information processing in chemical reaction systems by tuning self-organization behavior. *Chaos*, (to appear), 2004.

[LBP04b] D. Lebiedz and U. Brandt-Pollmann. Manipulation of surface reaction dynamics by global pressure and local temperature control - a model study. *Chaos*, 2004.

[LBP04c] D. Lebiedz and U. Brandt-Pollmann. Nonlinear feedback control of pattern formation in bacterial chemotoxis. 2004.

[LBS97] D.B. Leineweber, H.G. Bock, and J.P. Schlöder. Fast direct methods for real-time optimization of chemical processes. In *Proc. 15th IMACS*

World Congress on Scientific Computation, Modelling and Applied Mathematics Berlin, Berlin, 1997. Wissenschaft- und Technik-Verlag.

[Lei96] D.B. Leineweber. The theory of MUSCOD in a nutshell. IWR-Preprint 96-19, University of Heidelberg, 1996.

[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.

[LNP98] M. Lalee, J. Nocedal, and T. Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM J. Optim.*, 8(3):682–706, 1998.

[LP86] P. Lötstedt and L.R. Petzold. Numerical solution of nonlinear differential equations with algebraic constraints I: Convergence results for backward differentiation formulas. *Math. Comp.*, 46(174):491–516, 1986.

[LSBS03] D.B. Leineweber, A. Schäfer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part II: Software aspects and applications. *Comp. & Chem. Eng.*, 27:167–174, 2003.

[LW97] P. Li and G. Wozny. Dynamische Optimierung großer chemischer Prozesse mit Kollokationsverfahren am Beispiel Batch-Destillation. *Automatisierungstechnik*, 45(3):136–143, 1997.

[Map04] Maplesoft. *Maple*. Maple Inc., Waterloo, 9.0 edition, 2004.

[Mar78] N. Maratos. *Exact penaly function algorithms for finite-dimensional and control optimization problems*. PhD thesis, IMperial College, London, 1978.

[MBSL03] K.D. Mombaur, H.G. Bock, J.P. Schlöder, and R.W. Longman. Open-loop stable solution of periodic optimal control problems in robotics. *submitted to Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 2003.

[Mic76] M.L. Michelsen. Semi-implicit Runge-Kutta methods for stiff systems, program description and application examples. *Inst. f. Kemiteknik, Danmarks tekniske Højskole, Lynby*, 1976.

[Min04] Hoang Duc Minh. *Modeling, simulation and optimization of complex processes in catalytic monoliths, in preparation.* PhD thesis, University of Heidelberg, 2004.

[Mom02] K.D. Mombaur. *Stability Optimization of Open-Loop Controlled Walking Robots.* VDI Verlag, Düsseldorf, 2002. Download at: `http://www.ub.uni-heidelberg.de/archiv/1796/`.

[Nil97] S. Nilchan. *The Optimisation of Periodic Absorption Processes.* PhD thesis, Department of Chemical Engineering and Chemical Technology, Imperial College of Science, London, 1997.

[Noc92] J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 1:199–242, 1992.

[NR83] G.N. Newsam and J.D. Ramsdell. Estimation of sparse Jacobian matrices. *SIAM J. Alg. Disc. Meth.*, 4(3):404–417, 1983.

[NW99] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, 1999.

[Osb69] M.R. Osborne. On shooting methods for boundary value problems. *J. Math. Anal. Appl.*, 27:417–433, 1969.

[PB93] C.C. Pantelides and P.I. Barton. Equation oriented dynamic simulation: Current status and future perspectives. *Comp. Chem. Eng.*, pages 263–285, 1993.

[Pet82] L.R. Petzold. A Description of DASSL: A Differential-Algebraic System Solver. In *Proc. 10th IMACS World Congress*, Montreal, 1982.

[Pet91] L.R. Petzold. Dassl. `http://www.engineering.ucsb.edi/~cse/ddassl.tar.gz`, June 1991.

[Pli81] K.J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Master's thesis, University of Bonn, 1981.

[Pot88] A. Pothen. The complexity of optimal elimination trees. Technical Report CS-88-13, Pennsylvania State University, 1988.

[Pow76] M.J.D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In R. W. Cottle and C. E. eds. Lemke, editors, *SIAM-AMS Proceedings*, volume IX of *SIAM publications*, pages 53–72, 1976.

[Pow77] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. ed. Watson, editor, *Numerical Analysis Dundee*, volume 3 of *Springer Verlag*, pages 144–157, Berlin, 1977.

[Pow78a] M.J.D. Powell. Algorithms for nonlinear constraints that use Lagrangian functions. *Math. Prog*, 14(3):224–248, 1978.

[Pow78b] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, editor, *Numerical Analysis, Dundee 1977*, volume 630 of *Lecture Notes in Mathematics*, Berlin, 1978. Springer.

[Pow84] M.J.D. Powell. Convergence properties of algorithms for nonlinear optimization. *SIAM Review*, 28:487–500, 1984.

[Pow85] M.J.D. Powell. On the quadratic programming algorithm of Goldfarb and Idnani. *Math. Progr. Study*, 25:46–61, 1985.

[PT79] M.J.D. Powell and Ph.-L. Toint. On the estimation of sparse Hessian matrices. *SIAM J. Numer. Anal.*, 16:1060–1074, 1979.

[Roc88] M. Roche. Rosenbrock methods for differential algebraic equations. *Numer. Math.*, 52:46–63, 1988.

[SBPD$^+$03] A.A.S. Schäfer, U. Brandt-Pollmann, M. Diehl, H.G. Bock, and J.P. Schlöder. Fast optimal control algorithms with application to chemical engineering. In D. Ahr, R. Fahrion, M. Oswald, and G. Reinelt, editors, *Operations Research Proceedings*, pages 300–307, Heidelberg, 2003. Springer.

[SBS98] V.H. Schulz, H.G. Bock, and M.C. Steinbach. Exploiting invariants in the numerical solution of multipoint boundary value problems for DAEs. *SIAM J. Sci. Comp.*, 19:440–467, 1998.

[Sch88] J.P. Schlöder. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*, volume 187 of *Bonner Mathematische Schriften*. University of Bonn, Bonn, 1988.

[Sch96] V.H. Schulz. Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots. IWR-Preprint 96-12, University of Heidelberg, 1996. Ph.D. thesis.

[Sch99a] A.A.S. Schäfer. Numerische Verfahren für große nichtlineare beschränkte Optimierungsprobleme und ihr Einsatz bei Optimalsteuerungsproblemen. Master's thesis, University of Heidelberg, 1999.

[Sch99b] R. von Schwerin. *MultiBody System SIMulation. Numerical Methods, Algorithms, and Software.* Lecture Notes in Computational Science and Engineering Vol. 7. Springer, 1999.

[Sch04] A.A.S. Schäfer. *Ein effizientes reduziertes SQP-Verfahren zur Parameterschätzung und Optimalen Steuerung von (bio-)chemischen Prozessen mit wenigen Freiheitsgraden.* PhD thesis, University of Heidelberg, 2004.

[SG75] L.F. Shampine and M.K. Gordon. *Computer Solution of Ordinary Differential Equations.* W.H. Freeman and Co., San Francisco, CA, 1975.

[Spe80] B. Speelpenning. *Compiling fast partial derivatives of functions given by algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, 1980.

[Ste95] M.C. Steinbach. *Fast recursive SQP methods for large-scale optimal control problems.* PhD thesis, University of Heidelberg, 1995.

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

[TB96] P. Tanartkit and L.T. Biegler. A nested, simultaneous approach for dynamic optimization problems – I. *Comput. Chem. Engng*, 20(6/7):735–741, 1996.

[TPS00] M. Tadjouddine, J.D. Pryce, and A.F. Shaun. On the implementation of AD using elimination methods via source transformation. *AMOR Report*, 8, November 2000.

[WBPMS03] R. Winkler, U. Brandt-Pollmann, U. Moslener, and J.P. Schlöder. Time-lags in capital accumulation. In D. Ahr, R. Fahrion, M. Oswald, and G. Reinelt, editors, *Operations Research Proceedings*, pages 451–458, Heidelberg, 2003. Springer.

[Wil63] R.B. Wilson. *A simplicial algorithm for concave programming.* PhD thesis, Harvard University, 1963.

[WR03] Inc. Wolfram Research. *Mathematica.* Wolfram Research, Inc., Champaign, Illinois, 5.0 edition, 2003.