Towards Nonlinear Model-Based Online Optimal Control of Chemical Engineering Plants:

Parameterised Controls and Sensitivity Functions for Very Large-Scale Index-2 DAE Systems with State Dependent Discontinuities

Am Fachbereich Informatik der
Technischen Universität Darmstadt
eingereichte

Dissertation

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

von

Dipl.-Math. **Thomas Kronseder**

Referenten der Arbeit:	Prof. Dr. Oskar von Stryk Prof. Dr.-Ing. Wolfgang Marquardt
Tag der Einreichung:	03. Mai 2002
Tag der mündlichen Prüfung:	28. Juni 2002

D17

Preface

This treatise contains the results of my research arising from a cooperation of the Technische Universität München and of the Linde AG, Linde Engineering Division. It has been accepted by the Department of Computer Science of the Technische Universität Darmstadt (D17) as dissertation in partial fulfillment of the requirements for the degree of a Doktor-Ingenieur (Dr.-Ing.).

I would like to thank the referee of this thesis, my tutor and friend Prof. Dr. Oskar von Stryk for his perpetual support of my work and of my academic career as a whole. With his advice and commitment he has been my guide and patron for almost 6 years now.

I would like to express my gratitude to Prof. Dr. Dr.h.c. mult. Roland Bulirsch for his support of my work. My course of studies at the Technische Universität München has been strongly connected with the work done at his chair of numerical mathematics. Due to the commitment of Professor Bulirsch to the application of advanced numerical methods to real-world problems I found at his chair exactly what I had been looking for when starting as a student. Now I have been working at the chair of numerical mathematics for several years by myself.

I thank Prof. Dr.-Ing. Wolfgang Marquardt for revising this thesis as co-referee.

I thank the Linde AG, Linde Engineering Division, for providing me with the possibility to do the interesting research in the simulation and optimal control of processes from chemical engineering application. Especially, I would like to thank Dr. Peter Burr and Dr. Gabriele Engl for their steady support, and my colleague and friend Dipl.-Ing. Andreas Kröner for his ideas, his effort, and his expertise.

This work was supported by the Deutsche Forschungsgemeinschaft within the Schwerpunktprogramm Echtzeit-Optimierung großer Systeme (SPP 468) in the project no. BU 602/9-1. This work was supported by a grant of the Linde AG, Linde Engineering Division.

Greiling, December 2002 Thomas Kronseder

Contents

Introduction

To learn and from time to time
to apply what one has learned
– isn't that a pleasure ?

Confucius: Analects (1,1)

This thesis has been motivated by a concrete industrial control problem. The task is to find a suitable transition for a cryogenic air separation plant from one operational regime to another one. These cryogenic air separation plants are chemical engineering plants which separate ambient air into its main fractions in a distillation-like process.

In the last decades the *model predictive control* (MPC) technique has been developed. This technique promises an elegant solution of advanced control problems especially due to the availability of increasingly powerful numerical tools. However, there are still difficulties both on the theoretical side as soon as the control problem becomes a more general one (in particular if the process model is nonlinear) as well as on the practical side (mainly due to the necessary online capability of the controller), thus keeping model predictive control an area of intense research in both engineering science and numerical mathematics. In this thesis some of the more practical problems of MPC in the context of the industrial application in view are addressed.

Advanced Control Techniques

Due to strongly competitive markets chemical engineering industry is faced with steadily increasing demands on product quality and efficient utilisation of resources (e.g., energy, raw materials, machinery, human labour). In addition also environmental issues have found attention and must be considered in industrial production. As a consequence, the plants have to be run using more intelligent control strategies and in largely varying operational regimes.

On the one hand, these demands are met with the application of increasingly sophisticated numerical methods for simulation and optimisation of large-scale dynamical process models during the engineering phase, cf., e.g.,

[Marq 91], [PaBa 93], [Zapp 94]. Here, the determination of optimal quasi steady-state setpoints by numerical optimisation is state-of-the-art in chemical engineering [Voit 94], [ELBK 97], [Sörg 97] while in chemical engineering the computation of open-loop optimal control strategies for large-scale processes by numerical optimal control is a relatively young technique (cf., e.g., [Bieg 88], [BAFG 98], [EKKS 99]).

On the other hand, model predictive control has been developed since standard control methods, e.g., PID controllers, are in some cases no longer sufficient for some applications. This is especially the case if plants have to be operated under strongly varying operational conditions, or if state constraints are of major importance [BiRa 91]. In MPC the control task is formulated as an optimal control problem on a moving horizon which is solved repeatedly online in parallel to the physical process. Feedback is obtained by setting an estimate of the actual process state as initial conditions for the optimal control problem. A powerful feature of this approach is that the control moves are determined on the basis of a predicted future system behaviour, intrinsically accounting for disturbances by this state feedback. Due to the optimal control formulation and the predictive nature of the technique constraints can be naturally included into the controller design. This has been one of the incentives for the development of MPC [BiRa 91]. The price to be paid for nonlinear MPC (which is the general case of MPC) is a sufficiently detailed model of the process, the online solution of an optimal control problem, and the online estimation of the process state.

In case of very fast or large-scale processes the online requirements are often too high given present standard computational resources. Here, a possible remedy can be found in the theory of neighbouring extremals which provides the basis for a short-cut method. The basic idea is to extend the applicability of a once expensively computed optimal solution for a longer period in time by fast updates accounting for disturbances that are always present in real-world processes, e.g., [Kräm 85], [Pesc 89a], [Pesc 89b], [KuPe 90a], [KuPe 90b], [MaPes 94], [MaPes 95], [BüMa 98], [BüMa 00].

We introduce the optimal load-change of air separation plants as an advanced control task. Difficulties are given by the nonlinearity of the process, by constraints which have to be satisfied all the time, and the size of the models required for an adequate mathematical description of the plants. Based on this data we propose a control concept which is developed in the framework of nonlinear model predictive control. The concept takes into account the different time scales separating the control task into different subtasks. Especially, we discuss an algorithm for the computation of open-loop optimal controls and propose a fast online update technique. In its core the computa-

tion of the control strategy is based on integration and parametric sensitivity analysis of (in our case) very large-scale discontinuous so-called semi-explicit index-2 DAE[(i)] process models.

Discontinuities and Consistent Initial Conditions

This work focuses on nonlinear dynamical process models described by very large-scale DAEs with state and time dependent discontinuities.

On the one hand, these discontinuities arise from the discontinuous nature of the process itself. E.g., in chemical engineering there are processes which consist in a sequence of batches [Allg 97], [Feeh 98], or on a lower level one has to cope with phase changes. In mechanical multibody systems one can observe discontinuous effects caused by the transition between different regimes of friction [Eich 92]. On the other hand, discontinuities can be introduced by simplifications required in the course of modelling. Such simplifications may be mirrored, e.g., by the need for measured data which is often available in the form of tabular data interpolated with a low order of differentiability only.

A numerically satisfactory method for the detection and location of discontinuities is the switching function technique as presented in e.g., [Eich 92]. In this method each discontinuity is indicated by the sign change of a dedicated switching function. The switching function technique has been developed for implementation into integrator algorithms. There it is used to obtain basic information required for efficient integration of discontinuous dynamical systems as for numerical reasons integrators[(ii)] should be restarted after any discontinuity. Some practical details on the design of discontinuous models with an emphasis on switching functions can be found in [ScWi 94].

In the ODE case the initial integration step as well as the restart at discontinuities do not impose major difficulties, cf., e.g., [Watt 83], [GSB 87], [Sham 87]. This situation changes when a DAE has to be integrated numerically. Here, in general the integrator has to be provided with *consistent initial conditions* in order to avoid inefficiency or unnecessary failure [Petz 82]. In addition to the equations of the original DAE initial values have to satisfy constraints that are "hidden" in the DAE. Moreover, without insight into the nature of the problem described by the DAE and without deeper analysis of the DAE itself it is in general neither clear how many dynamic degrees of

[(i)] DAEs are systems of coupled differential and algebraic equations. For a detailed discussion of theoretical aspects of DAEs see Section 1.1.

[(ii)] We restrict to *sequential* integration algorithms such as one-step and multistep methods. For *simultaneous* integration with collocation methods the situation is different.

freedom have to be specified nor which of the variables in the DAE system are suitable candidates for specification.

The development of methods for the (semi-)automatic computation of consistent initial values is still in progress. The techniques available range from purely integrator related methods such as special initial backward Euler-steps [KMG 92], [BCP 96], the solution of a BVP [AmMa 98], and back-tracing [SEYE 81], to more DAE related methods that are based on the compilation and solution of the *consistency equations* [Pant 88a], [LPG 91], [MaSö 93], [CaMo 94], [UKM 95], [CKY 96], [GoBi 99], [CVSB 01], [WuWh 01], or projector based techniques [Hans 92], [Lamo 97], [EsLa 99].

In Chapter 3 we review the different methods for consistent initialisation of higher index DAEs. Keeping these methods in mind we develop two algorithms tailored for the consistent initialisation of large-scale semi-explicit index-2 DAEs arising from real-life chemical engineering applications. These two algorithms are implemented into the simulation environment OPTISIM® of the Linde AG, Linde Engineering Division, [Burr 91], [Burr 93], and are compared in numerical examples.

Sensitivities for Discontinuous Models

In general, models depend on parameters. Thus the question arises on the dependency of the state variable trajectories from the parameters. Mathematically this dependency is expressed by the (parametric) *sensitivity functions*, *sensitivity matrices*, or, shortly, *sensitivities*. If parametric sensitivities of state variable trajectories are to be computed the question is on how discontinuities affect the sensitivities.

One of the first successful attempts to treat the problem of computing sensitivities in discontinuous systems of ODEs has been reported by [Roze 67]. [Feeh 98] derives the corresponding expressions for index-1 DAEs. Furthermore he shows that his work includes the results of [Roze 67] as a special case.

In most contributions on discontinuities only real-valued switching functions have been treated. In the process models of our interest, however, in general several switching functions (or, from a different point of view, one multi-dimensional switching function) are incorporated. Thus, the sequence of discontinuities may change according to the value of the model parameters. The problem is that differentiability with respect to the parameters may be lost at points in parameter space where the sequence of discontinuity events changes [Feeh 98].

In the case of multi-dimensional switching functions an indicator for a po-

tential loss of differentiability is that several entries in the vector of switching functions change their sign at the same point. However, this is no sufficient condition for non-differentiability. It will shown that in a special case differentiability is preserved, and that the test for this special case requires few additional computational effort only.

In Chapter 4 we discuss several approaches to the transfer of parametric sensitivities in discontinuous DAE systems and develop two algorithms for the sensitivity transfer in our special problem setting. These two algorithms are directly based on our algorithms for consistent initialisation developed before.

Notational Conventions

We have chosen to enclose arguments of mappings in general within brackets $(\cdot)$, while square brackets $[\cdot]$ enclose vectors and matrices. Therefore, the construct $\binom{\mu}{\nu}$ denotes the binomial coefficient $\mu!/(\nu!(\mu-\nu)!)$, while in, e.g., $\boldsymbol{f}\left(\left[\begin{smallmatrix}\mu\\ \nu\end{smallmatrix}\right]\right)$ the vector $[\mu^T, \nu^T]^T$ is the argument of the mapping $\boldsymbol{f}$.

Sets are denoted by curly braces $\{\cdot\}$. $\mathbb{N}$ is the set of positive integers, $\mathbb{N}_0$ is defined as $\mathbb{N} \cup \{0\}$. $\mathbb{R}_+$ denotes the set of all non-negative reals, $\mathbb{R}_+ \backslash \{0\}$ is the set of all strictly positive reals. The number of elements in a set is denoted by $\mathrm{card}(\cdot)$. $\uplus$ denotes the union of two disjoint sets.

In general, $\boldsymbol{f}$, $\boldsymbol{g}$, $\boldsymbol{s}$, and $\boldsymbol{F}$ denote real, multi-dimensional functions, while $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, and $\boldsymbol{\xi}$ are reserved for variable vectors. Mostly, parameter vectors are indicated by $\boldsymbol{p}$. Dimensions of multi-dimensional functions are in common of the form m_*, those of variable and parameter vectors are of the form n_*. The real-valued independent variable in the systems of equations is t (time).

$\mathcal{C}^0(\mathbb{R}^\mu, \mathbb{R}^\nu)$ denotes the continuous mappings from $\mathbb{R}^\mu$ into $\mathbb{R}^\nu$, while $\mathcal{C}^1(\mathbb{R}^\mu, \mathbb{R}^\nu)$ represents the continuously differentiable mappings from $\mathbb{R}^\mu$ into $\mathbb{R}^\nu$. $\mathcal{C}^0_{\mathrm{p}}(\mathbb{R}^\mu, \mathbb{R}^\nu)$ denotes the piecewise continuous mappings from $\mathbb{R}^\mu$ into $\mathbb{R}^\nu$.

Let there be a differentiable multi-dimensional function $\boldsymbol{F} \in \mathcal{C}^1(n_{\boldsymbol{\xi}}, m_{\boldsymbol{F}})$: $\boldsymbol{\xi} \mapsto \boldsymbol{F}(\boldsymbol{\xi})$. Then we denote its Jacobian with respect to $\boldsymbol{\xi}$ by

$$D_{\boldsymbol{\xi}}\boldsymbol{F} := \boldsymbol{F}_{\boldsymbol{\xi}} := \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\xi}} := \begin{bmatrix} \partial \boldsymbol{F}_1/\partial \boldsymbol{\xi}_1 & \cdots & \partial \boldsymbol{F}_1/\partial \boldsymbol{\xi}_{n_{\boldsymbol{\xi}}} \\ \vdots & \ddots & \vdots \\ \partial \boldsymbol{F}_{m_{\boldsymbol{F}}}/\partial \boldsymbol{\xi}_1 & \cdots & \partial \boldsymbol{F}_{m_{\boldsymbol{F}}}/\partial \boldsymbol{\xi}_{n_{\boldsymbol{\xi}}} \end{bmatrix} \in \mathbb{R}^{m_{\boldsymbol{F}} \times n_{\boldsymbol{\xi}}}.$$

Similarly, given sufficient differentiability we define the total derivative of the multi-dimensional function $\boldsymbol{F} \in \mathbb{R}^{m_{\boldsymbol{F}}}$ with respect to a vector, say $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$,

as

$$\frac{\mathrm{d}\boldsymbol{F}}{\mathrm{d}\boldsymbol{p}} := \begin{bmatrix} \mathrm{d}\boldsymbol{F}_1/\mathrm{d}\boldsymbol{p}_1 & \cdots & \mathrm{d}\boldsymbol{F}_1/\mathrm{d}\boldsymbol{p}_{n_{\boldsymbol{p}}} \\ \vdots & \ddots & \vdots \\ \mathrm{d}\boldsymbol{F}_{m_{\boldsymbol{F}}}/\mathrm{d}\boldsymbol{p}_1 & \cdots & \mathrm{d}\boldsymbol{F}_{m_{\boldsymbol{F}}}/\mathrm{d}\boldsymbol{p}_{n_{\boldsymbol{p}}} \end{bmatrix} \in \mathbb{R}^{m_{\boldsymbol{F}} \times n_{\boldsymbol{p}}} .$$

For a real-valued function $\boldsymbol{F} \in \mathcal{C}^1(n_{\boldsymbol{\xi}}, 1) : \boldsymbol{\xi} \mapsto \boldsymbol{F}(\boldsymbol{\xi})$ the Nabla operator is defined as

$$\nabla \boldsymbol{F}(\boldsymbol{\xi}) := \begin{bmatrix} \frac{\partial \boldsymbol{F}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}_1} \\ \vdots \\ \frac{\partial \boldsymbol{F}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}_{n_{\boldsymbol{\xi}}}} \end{bmatrix} = \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) \right]^T .$$

More specifically, $\nabla_{\boldsymbol{x}} := [\partial/\partial \boldsymbol{x}]^T$. Please note that in this treatise the symbol Δ is used as a qualifier only (i.e., it is *not* the Laplace operator), e.g., $\Delta \boldsymbol{p}$ means a vector in $\mathbb{R}^{n_{\boldsymbol{p}}}$ of usually small norm in relation to the vector $\boldsymbol{p}$.

Matrices are in general denoted by non-boldface capital letters, most commonly A, B, J, M, P, Q. Exceptions are the identity matrix $\boldsymbol{Id} = \mathrm{diag}(1, \ldots, 1) \in \mathbb{R}^{\nu \times \nu}$, $\nu \in \mathbb{N}$, and the matrix filled with zeros $\mathbf{0} \in \{0\}^{\mu \times \nu}$, $\mu, \nu \in \mathbb{N}$; in most cases their dimensions are not especially noted.

$\boldsymbol{k}$ is reserved for functions describing point constraints, including initial or other boundary conditions. This is already a link to optimal control problems. There, $\boldsymbol{c}$ is used for path inequality constraints, while $\mathcal{J}$ denotes objective functions.

μ and ν are general purpose indices.

Additional symbols will be introduced on demand.

We have added paragraphs which are not strictly necessary for the development of the main points of this work, but which provide additional thoughts and information. These paragraphs are printed in a smaller font and are introduced as

Remark 0.1:

... ◊

The notation in various citations has been adapted, mainly in order to ease keeping the thread between the different sections.

Disclaimer

Chapter 1

Fundamentals

1.1 Some Fundamental Definitions in the Context of DAEs

The Tao is forever undefined.
Small though it is in the unformed state,
it cannot be grasped.
If kings and lords could harness it,
The ten thousand things would come together
And gentle rain fall.
Men would need no more instruction and
all things would take their course.

Lao-Tse: Tao-te-king

Ordinary differential equations (ODEs) are commonly used by scientists and engineers in order to represent dynamical processes in mathematical terms.

In the course of the modelling of physical systems often additional algebraic constraints arise naturally [Matt 89]. Although it may be possible to find an equivalent description by a system consisting of ordinary differential equations only, the description by the original system of differential *and* algebraic equations has been recommended due to advantages in computational modelling and simulation, e.g., [SEYE 81], [GePe 84].

Such combined systems are called *differential-algebraic equations* (DAE).

Remark 1.1:
DAEs are commonly encountered, e.g., in mechanical multibody systems (e.g., [ABES 93], [Führ 88], [FüLe 91]), in the simulation of electric circuits (e.g., [Asch 90], [GüFe 99a], [GüFe 99b]), and in the modelling of chemical engineering processes (e.g., [ByPo 88], [Bart 92], [BSS 95], [ELBK 97]). They also arise if PDEs are solved using the method of lines (e.g., [BCP 96], [PRGJP 97], [Asch 99]). Moreover, optimal control problems with path constraints are to be considered as DAE problems (e.g., [FBB 95], [FeBa 98]). ◊

The generic form of a DAE initial value problem (DAE-IVP) is

$$\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) = 0\,; \quad t \in [t_0, t_\mathrm{f}] \subset \mathbb{R}, \tag{1.1a}$$

$$\boldsymbol{k}^\mathrm{ini}(t_0, \boldsymbol{\xi}(t_0), \dot{\boldsymbol{\xi}}(t_0)) = 0\,, \tag{1.1b}$$

where $t \in [t_0, t_\mathrm{f}]$ is the independent variable (time), $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ is the vector of state variables, and $\boldsymbol{F} : \mathbb{D}_{\boldsymbol{F}} \to \mathbb{R}^{n_\xi}$, $\mathbb{D}_{\boldsymbol{F}} \subseteq [t_0, t_\mathrm{f}] \times \mathbb{R}^{n_\xi + n_\xi}$, is a system of first order DAEs. $\dot{\boldsymbol{\xi}} := \partial\boldsymbol{\xi}/\partial t$ denotes the derivative of $\boldsymbol{\xi}$ with respect to t. The initial conditions are specified via $\boldsymbol{k}^\mathrm{ini} : \mathbb{R}^{1+n_\xi+n_\xi} \to \mathbb{R}^{m_{\boldsymbol{k}^\mathrm{ini}}}$.

In general, the Jacobian[(i)] $[\partial\boldsymbol{F}/\partial\dot{\boldsymbol{\xi}}](t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}})$ of the DAE system $\boldsymbol{F}(t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}})$ with respect to the time differentials of the state variables $\dot{\boldsymbol{\xi}}$ is singular in the domain of $\boldsymbol{F}$; (implicit) ODEs are a special case of DAEs (1.1a) with $\det([\partial\boldsymbol{F}/\partial\dot{\boldsymbol{\xi}}](t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}})) \neq 0$ along a trajectory. However, *ODEs* and *DAEs* are distinguished due to fundamental differences between both types of dynamical systems.

1.1.1 The Solution of a DAE

In slight modification of [KuMe 94] we define the *solution* of the generic DAE(-IVP) as

Definition 1.1 (Solution of a DAE / DAE IVP)
A function $\boldsymbol{\xi} : t \mapsto \boldsymbol{\xi}(t), [t_0, t_f] \to \mathbb{R}^{n_\xi}$, *is called a* solution *of Eq.* (1.1a) *if* $\boldsymbol{\xi}(t)$ *is sufficiently smooth with respect to the free variable* t *in each of its component functions* $\boldsymbol{\xi}_\mu : t \mapsto \boldsymbol{\xi}_\mu(t), [t_0, t_f] \to \mathbb{R}$, $\mu = 1, \ldots, n_{\boldsymbol{\xi}}$, $\boldsymbol{\xi} := [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{n_{\boldsymbol{\xi}}}]^T$, *and if* $\boldsymbol{\xi}(t)$ *satisfies Eq.* (1.1a) *pointwise for all* $t \in [t_0, t_f]$.

It is called a solution of the DAE initial-value problem *Eqs.* (1.1a)–(1.1b) *if* $\boldsymbol{\xi}$ *is a solution of Eq.* (1.1a) *and if* $\boldsymbol{\xi}$ *satisfies the initial condition Eq.* (1.1b).

Definition 1.1 does not explicitly specify the minimum smoothness of a solution[(ii)]. However, obviously at least *some* of the components of a solution vector $\boldsymbol{\xi}(t)$ must be of class $\mathcal{C}^1$ with respect to the free variable t (apart from the degenerate purely algebraic case $[\partial\boldsymbol{F}/\partial\dot{\boldsymbol{\xi}}](t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) \equiv 0$). Moreover, there may be components of a solution of Eq. (1.1a) for which existence of higher order (≥ 2) derivatives with respect to time is required later on. This property is summarised in demanding sufficient smoothness of $\boldsymbol{\xi}(t)$ and will not be noted in the sequel anymore.

In difference to ODE(-IVP)s where existence and uniqueness of the solution can be established under weak conditions on the smoothness of the ODE (cf., e.g., [HNW 87]), existence and uniqueness results in the DAE case are more difficult. [RaRh 94] obtain theoretical results based on the differential geometric approach towards DAEs. There, existence and uniqueness in the DAE case is reduced to

[(i)] Unless explicitly specified we assume throughout this treatise that all functions are sufficiently often differentiable with respect to their arguments.

[(ii)] This is in contrast to [KuMe 94] where a solution has to be of class $\mathcal{C}^1$.

standard ODE theory by construction of a sequence of shrinking manifolds which finally becomes stationary. The intersection of these submanifolds is called the *core*, and is again a submanifold. Additionally, on the core an ODE is constructed which is equivalent to the original DAE.

A characteristic property of DAEs is that – unlike in the case of ODEs – the *structure* of the DAE Eq. (1.1a) determines the admissibility of initial conditions Eq. (1.1b) in a non-trivial way. Here, we use admissibility in the sense that a solution of the DAE-IVP exists, given solvability of the DAE (1.1a) (see Definition 1.2 below). This fundamental issue is subject to closer examination in Chapter 3. Therefore, $\boldsymbol{k}^{\mathrm{ini}}(\cdot)$ will be specified in detail later.

1.1.2 Solvability of DAEs

We assume that a connected open set of initial values is given, e.g., as the manifold defined by a parameterised set of initial conditions, as well as an open interval in time. In common, a DAE (1.1a) is said to be *solvable* if for each of the initial values a locally unique solution exists on the entire interval in time and if the graphs of the solutions form a smooth manifold [BCP 96] (the *solution manifold* [CaGr 95]):

Definition 1.2 (Solvability of a DAE)
Let $I \subset \mathbb{R}$ be an open interval, Ω a connected subset of $\mathbb{R}^{1+n_\xi+n_\xi}$, and let $\boldsymbol{F} : \Omega \to \mathbb{R}^{n_\xi}$ be a differentiable function. Then the DAE (1.1a) *is solvable on I in Ω if there is an r-dimensional family of solutions $\phi(t,c)$ defined on a connected open set $I \times \widetilde{\Omega}$, $\widetilde{\Omega} \subset \mathbb{R}^r$, such that:*

1. *$\phi(t,c)$ is defined on all of I for each $c \in \widetilde{\Omega}$.*
2. *$[t, \phi(t,c), \dot{\phi}(t,c)] \in \Omega$ for $[t,c] \in I \times \widetilde{\Omega}$.*
3. *If $\psi(t)$ is any other solution with $[t, \psi(t), \dot{\psi}(t)] \in \Omega$, then $\psi(t) = \phi(t,c)$ for some $c \in \widetilde{\Omega}$.*
4. *The graph of ϕ as a function of $[t,c]$ is an $r+1$-dimensional manifold* [(iii)].

Only in some special cases solvability of DAEs can be verified a priori, cf., e.g, [UKM 95], [BCP 96]. Otherwise, verification of solvability requires considerable numerical effort. E.g., [CaGr 95] have developed an algorithm to check a slightly modified notion of solvability. However, this method is primarily suitable for DAEs of moderate size given in analytical form. Therefore we *assume* throughout this treatise that the DAEs considered are solvable according to Definition 1.2.

[(iii)] *In [CaGr 95] the term manifold is specified as* differentiable manifold. *We adopt to this definition.*

Remark 1.2:
Other notions of solvability can be found, e.g., in [CaGe 95] (*geometric/uniform solvability*). [RaRh 94a], [RaRh 94b], [Arno 97] treat DAEs which own solutions with *impasse points*, i.e., points that can be reached but not passed by a solution in the classical sense [Arno 97]. ◊

1.1.3 Special forms of DAEs

Several special forms of DAEs are distinguished. As with PDEs a classification according to formal properties is motivated by corresponding analytical and numerical characteristics. Additionally, in certain fields of application DAE models of a typical form are widely spread.

E.g., in chemical engineering the DAEs encountered are in general of the *linear-implicit* type [Marq 91], [EKKS 99]:

Definition 1.3 (Linear-Implicit DAE)
A DAE is called linear-implicit, *if it is of the form*

$$\begin{bmatrix} A_1(t,\boldsymbol{x}(t),\boldsymbol{y}(t)) & A_2(t,\boldsymbol{x}(t),\boldsymbol{y}(t)) \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \dot{\boldsymbol{x}}(t) \\ \dot{\boldsymbol{y}}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(t,\boldsymbol{x}(t),\boldsymbol{y}(t)) \\ \boldsymbol{g}(t,\boldsymbol{x}(t),\boldsymbol{y}(t)) \end{bmatrix},$$

with $\boldsymbol{f} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$, $\boldsymbol{g} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$, $A_1 : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{x}}\times n_{\boldsymbol{x}}}$, $A_2 : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{x}}\times n_{\boldsymbol{y}}}$, *and if* $[A_1]^{-1}$ *exists and is bounded in a neighbourhood of the solution.*

In the sense of the following definition *semi-explicit* DAEs are a special case of linear-implicit DAEs [BCP 96]:

Definition 1.4 (Semi-Explicit DAE)
A DAE is called semi-explicit, *if it is of the form*

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t,\boldsymbol{x}(t),\boldsymbol{y}(t)), \\ 0 &= \boldsymbol{g}(t,\boldsymbol{x}(t),\boldsymbol{y}(t)), \end{aligned} \tag{1.2}$$

with $\boldsymbol{f} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$, *and* $\boldsymbol{g} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$.

In the semi-explicit case there is a natural separation of the variables into *differential variables* $\boldsymbol{x}$ and *algebraic variables* $\boldsymbol{y}$, as well as a separation of the DAE system into a differential part $\boldsymbol{f}$ and algebraic constraints $\boldsymbol{g}$.

Remark 1.3:
In literature, the term *semi-explicit* DAE is also applied to more general systems [BCP 96]

$$\begin{aligned} 0 &= \boldsymbol{f}(t,\boldsymbol{x}(t),\boldsymbol{y}(t),\dot{\boldsymbol{x}}(t)), \\ 0 &= \boldsymbol{g}(t,\boldsymbol{x}(t),\boldsymbol{y}(t)), \end{aligned}$$

with $\boldsymbol{f} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$ and $\boldsymbol{g} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$, given that $[\partial \boldsymbol{f}/\partial \dot{\boldsymbol{x}}]^{-1}$ exists and is bounded in a neighbourhood of the solution. This definition of semi-explicit DAEs is related more closely to general nonlinear DAEs than to linear-implicit DAEs. ◊

Another special form are DAEs of *Hessenberg form* [BCP 96]:

Definition 1.5 (DAE of Hessenberg form)
A DAE is in Hessenberg form *of size r if it is written*

$$\begin{aligned} \dot{\boldsymbol{x}}_1(t) &= \boldsymbol{f}_1(t, \boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \ldots, \boldsymbol{x}_{r-1}(t), \boldsymbol{x}_r(t)) \,, \\ \dot{\boldsymbol{x}}_\mu(t) &= \boldsymbol{f}_\mu(t, \boldsymbol{x}_{\mu-1}(t), \boldsymbol{x}_\mu(t), \ldots, \boldsymbol{x}_{r-1}(t)) \,; \quad \mu = 2, \ldots, r-1 \,, \\ 0 &= \boldsymbol{f}_r(t, \boldsymbol{x}_{r-1}(t)) \,, \end{aligned} \tag{1.3}$$

$\boldsymbol{f}_1 : \mathbb{R}^{1+\sum_{\nu=1}^{r} n_{\boldsymbol{x}\nu}} \to \mathbb{R}^{n_{\boldsymbol{x}1}}$, $\boldsymbol{f}_\mu : \mathbb{R}^{1+\sum_{\nu=\mu-1}^{r-1} n_{\boldsymbol{x}\nu}} \to \mathbb{R}^{n_{\boldsymbol{x}\mu}}$, $\mu = 2, \ldots, r$, *and if*

$$\left[\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{x}_{r-1}}\right] \cdot \left[\frac{\partial \boldsymbol{f}_{r-1}}{\partial \boldsymbol{x}_{r-2}}\right] \cdots \left[\frac{\partial \boldsymbol{f}_2}{\partial \boldsymbol{x}_1}\right] \cdot \left[\frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{x}_r}\right] \text{ is nonsingular.}$$

Remark 1.4:
By definition, $\partial \boldsymbol{f}_{\nu+1}/\partial \boldsymbol{x}_\nu \in \mathbb{R}^{n_{\boldsymbol{x}\nu+1} \times n_{\boldsymbol{x}\nu}}$, $\nu = 1, \ldots, r-1$, and $\partial \boldsymbol{f}_1/\partial \boldsymbol{x}_r \in \mathbb{R}^{n_{\boldsymbol{x}1} \times n_{\boldsymbol{x}r}}$. Thus the matrix product in Definition 1.5 is well defined, and the product is a square matrix in $\mathbb{R}^{n_{\boldsymbol{x}r} \times n_{\boldsymbol{x}r}}$. ◊

1.1.4 The Index of a DAE

An important attribute of a DAE is its so-called *index*[iv] $\iota \in \mathbb{N}_0$. There are several different definitions of an index.

One of the most often encountered indices is the *differential index* ι_{d} [BCP 96]. Its definition is based on structural properties of a DAE (cf. Definition 1.7 below). The *perturbation index* ι_{p} can be regarded as a counterpart to the differential index as it directly refers to numerical properties of a DAE. By construction it provides a measurement for the influence of disturbances on the solution of a DAE [HW 91], [Eich 92]:

Definition 1.6 (Perturbation Index)
The DAE system $\boldsymbol{F}(t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) = 0$, $\boldsymbol{F} : \mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{\xi}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$, has the perturbation index *ι_p along a solution $\boldsymbol{\xi}(t)$ for t in a bounded interval $[0, t_f]$, if ι_p is the smallest integer such that for all functions $\widehat{\boldsymbol{\xi}}(t)$ having a defect*

$$\boldsymbol{F}(t, \widehat{\boldsymbol{\xi}}, \dot{\widehat{\boldsymbol{\xi}}}) = \delta(t)$$

[iv] Often the index is denoted with the symbol ν. We have decided to stick to ι (*iota*) as we use ν for enumerations.

there exists an estimate on $[0, t_f]$

$$\|\widehat{\boldsymbol{\xi}}(t) - \boldsymbol{\xi}(t)\| \leq C \left(\|\widehat{\boldsymbol{\xi}}(0) - \boldsymbol{\xi}(0)\| + \max_{0\leq\tau\leq t} \| \int_0^\tau \delta(\theta)\, d\theta\| \right.$$
$$\left. + \max_{0\leq\tau\leq t} \|\delta(\tau)\| + \ldots + \max_{0\leq\tau\leq t} \|\delta^{(\iota_p - 1)}(\tau)\| \right)$$

whenever the expression on the right-hand side is sufficiently small. $C \in \mathbb{R}_+$ *denotes a constant which only depends on* $\boldsymbol{F}$ *and* t_f.

The differential-geometric approach towards DAEs as vector fields on manifolds motivates the definition of a *geometric index* ι_{g} [Reic 90], [RaRh 94]. The projector based approach of [März 89], [März 92] leads to the introduction of the *tractability index* ι_{t}; we will take a closer look at it in Section 3.1.9. In order to overcome some theoretical shortcomings of the differential index and of the perturbation index [CaGe 95] introduce the *uniform index* ι_{U}, the *maximum differentiation index* ι_{MD}, the *maximum perturbation index* ι_{MP}, and the *uniform differentiation index* ι_{UD}. Relationships between different indices are examined, e.g., in [GHM 92], [CaGe 95], [Hank 95].

Remark 1.5:
[Gear 90] made the widely spread assertion that the differential index of a DAE is equal to its perturbation index, or higher by one at most. By construction of a family of counterexamples [CaGe 95] show that this is *not* always the case. The problem is that continuity with respect to a class of perturbations is not involved in the definitions of both indices. However, [CaGe 95] prove the relations $\iota_{\mathrm{UD}} \leq \iota_{\mathrm{MP}} \leq \iota_{\mathrm{UD}} + 1$ between the *uniform* differentiation index and the *maximum* perturbation index. ◊

We restrict to the *differential index* as discussed in [BCP 96][(v)]:

Definition 1.7 (Differential Index of a DAE)
Let there be a DAE

$$\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) = 0\,, \tag{1.4}$$

$\boldsymbol{F} : \mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{\xi}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$. *Furthermore, with* $\boldsymbol{\xi}^{(i)} := \frac{d^i}{dt^i}\boldsymbol{\xi}(t)|_t \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$ *define* $\vec{\boldsymbol{\xi}}_j := \left[\left[\boldsymbol{\xi}^{(1)}\right]^T, \ldots, \left[\boldsymbol{\xi}^{(j)}\right]^T \right]^T$ *and*

$$0 = \vec{\boldsymbol{F}}_j(t, \boldsymbol{\xi}, \vec{\boldsymbol{\xi}}_j)$$
$$:= \begin{bmatrix} \boldsymbol{F}_{[0]}(t, \boldsymbol{\xi}^{(0)}, \boldsymbol{\xi}^{(1)}) \\ \boldsymbol{F}_{[1]}(t, \boldsymbol{\xi}^{(0)}, \boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(2)}) \\ \vdots \\ \boldsymbol{F}_{[j-1]}(t, \boldsymbol{\xi}^{(0)}, \boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(j)}) \end{bmatrix} := \begin{bmatrix} \boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) \\ \frac{d}{dt}\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) \\ \vdots \\ \frac{d^{j-1}}{dt^{j-1}}\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) \end{bmatrix}. \tag{1.5}$$

[(v)] Unless stated differently, *index* always denotes the *differential index* in the sequel.

The differential index ι_d *of Eq.* (1.4) *is the smallest integer* ι_d *such that* $\vec{\boldsymbol{F}}_{\iota_d+1}$ *uniquely determines the* variable $\dot{\boldsymbol{\xi}}$ *as a continuous function of* $\boldsymbol{\xi}$ *and* t.

Remark 1.6:
$\boldsymbol{\xi}^{(i)}$, $i = 1, \ldots, \iota_\mathrm{d}$, represent the (higher order) time derivatives of the state vector at a fixed, but otherwise arbitrary point of a solution of the DAE. In Eq. (1.5) the variables $\boldsymbol{\xi}^{(i)}$ are interpreted as *free* variables. ◊

Definition 1.7 does not specify the subset in $\mathbb{R}^{1+n_\xi+n_\xi}$ within which the minimising property of ι_d has to hold – the reference work [BCP 96] only remarks that "... As with the definition of solvability in [Definition 1.2], all of these statements, including [Definition 1.7], are taken to hold locally on open subsets of $\mathbb{R}^{1+j\cdot n_\xi}$." ([BCP 96], p.33). This often neglected specification of the set on which ι_d is to hold has already been mentioned in [CaGe 95]. However, the consideration of open subsets in the definition space of the DAEs implicates an interpretation as a global property (e.g., [Unge 90] gives the argument "... [Definition 1.9 (which is a reformulation of Definition 1.7)] does not state anything about *exceptions* at some points along a trajectory of a solution ..." ([Unge 90], p.15)).

Apart from some special forms of DAEs the differential index of a DAE can be considered as a measure of the difficulties that are to be expected for its numerical and analytical treatment (this is also true with other definitions of an index). DAEs with index $\iota_\mathrm{d} \geq 2$ are called *higher index problems* [CaGe 95]. Their treatment is in general substantially more difficult than the index-1 case.

Eq. (1.5) will be of importance later on. Therefore we introduce [LPG 91]

Definition 1.8 (Derivative (Array) Equations)
Equations (1.5) *are called the* derivative equations *or* derivative array equations *of DAE* (1.4).

Definition 1.7 can be stated more intuitively as [BCP 96]

Definition 1.9 (Differential Index of a DAE, Reformulated)
The minimum number of times that all or part of

$$\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) = 0\,, \tag{1.6}$$

$\boldsymbol{F} : \mathbb{R}^{1+n_\xi+n_\xi} \to \mathbb{R}^{n_\xi}$, *must be differentiated with respect to* t *in order to determine* $\dot{\boldsymbol{\xi}}$ *as a continuous function of* $\boldsymbol{\xi}$ *and* t, *is the* differential index ι_d *of the DAE* (1.6).

A more practical, but less general way for the characterisation of the differential index than in Definition 1.7 and Definition 1.9 is the following algorithm discussed in [Gear 88], which essentially reduces a DAE $\boldsymbol{F}(t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) = 0$ of arbitrary index to an index-0 DAE, i.e., an ODE. The number of iterations performed before termination are then equal to the differential index of DAE (1.1a) [BCP 96]:

Algorithm 1 (Index Reduction and Determination)
1. *If* $[\partial \boldsymbol{F}/\partial \dot{\boldsymbol{\xi}}]$ *is nonsingular, then stop.*

2. *Suppose that* $[\partial \boldsymbol{F}/\partial \dot{\boldsymbol{\xi}}]$ *has constant rank and that nonlinear coordinate changes make the DAE semi-explicit (in the sense of Definition 1.4).*

   Differentiate the algebraic equations, let $\boldsymbol{F} = 0$ *denote the new DAE, and return to 1.*

This method does not work in the general case as the conversion to semi-explicit form in step **2** is not always possible.

In [LöPe 86] the number of iterations required by Algorithm 1 is called the *global index* of a (nonlinear) DAE. Therefore, the term "global index" is synonymous to the differential index.

Remark 1.7:
For certain forms of DAEs straightforward index conditions can be formulated, e.g.:

(a) The differential index of the semi-explicit DAE (1.2) is one if and only if $[\partial \boldsymbol{g}/\partial \boldsymbol{y}]$ is non-singular [BCP 96].

(b) The differential index of a DAE (1.3) in Hessenberg form of size r is r [BCP 96].

(c) As a special case of **b**, the differential index of a semi-explicit DAE (1.2) in *triangular form* (i.e., the algebraic equations are of the type $0 = \boldsymbol{g}(t, \boldsymbol{x}(t))$) is two if $[[\partial \boldsymbol{g}/\partial \boldsymbol{x}] \cdot [\partial \boldsymbol{f}/\partial \boldsymbol{y}]]$ is non-singular [LPG 91].

In all three cases the index criterion is expressed as a condition on the rank of a matrix (the rank condition for case **b** is already contained in Definition 1.5). As the numerical determination of the rank of a matrix is expensive for larger systems and tends to be badly conditioned [UKM 95], it is advantageous to rely on the structural properties of the matrices and employ an algorithm as described in [Duff 81]. An additional problem of conditions **b** and **c** in connection with large (and sparse) DAEs is that the product of the sparse Jacobians involved in general need no longer be sparse.

For semi-explicit DAEs of the type

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t, \boldsymbol{x}(t)) + G\boldsymbol{y}(t)\,, \\ 0 &= H\boldsymbol{x}(t) - A\boldsymbol{y}(t)\,, \end{aligned}$$

with $\boldsymbol{f} : \mathbb{R}^{1+n_{\boldsymbol{x}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$ and constant matrices $A \in \mathbb{R}^{n_{\boldsymbol{y}} \times n_{\boldsymbol{y}}}$, $G \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{y}}}$, $H \in \mathbb{R}^{n_{\boldsymbol{y}} \times n_{\boldsymbol{x}}}$ [DuGe 86] have proposed an algorithm to determine whether the index exceeds two. The algorithm is based on structural properties of the Jacobian of the DAE. A drawback is that it can exhibit non-polynomial complexity for certain problems.

In Section 3.1.5 we discuss the *Algorithm of Pantelides* which (with some restrictions) can be utilised to evaluate the index of a general DAE (1.1a). It is also based on structural properties of the DAE but has a computational complexity of $\mathcal{O}(n_{\boldsymbol{\xi}}^3)$ [UKM 95]. ◊

1.1.5 Consistent Initial Conditions

As will be discussed in detail within Section 3.1.8, Algorithm 1 detects information besides the differential index:

It generates a system of explicit ODEs which is called the *underlying ODE* (UODE) of the DAE. Additionally, during the course of its run *new* algebraic equations appear which are not obvious in the original DAE. These algebraic equations

are called *hidden constraints* or *hidden equations.* Taken together the UODE and all algebraic equations give the corresponding extended system [UKM 95]:

Definition 1.10 (Corresponding Extended System of a DAE)
The corresponding extended system

$$\begin{aligned} \dot{\boldsymbol{\xi}}(t) &= \boldsymbol{\mathcal{F}}(t, \boldsymbol{\xi}(t)) \,, \\ 0 &= \boldsymbol{\mathcal{G}}(t, \boldsymbol{\xi}(t)) \,, \end{aligned} \tag{1.7}$$

of a DAE (1.1a) *is built up by the underlying ODE* $\boldsymbol{\mathcal{F}} : \mathbb{R}^{1+n_\xi} \to \mathbb{R}^{n_\xi}$ *together with the hidden constraints and the algebraic equations of the original DAE collected in* $\boldsymbol{\mathcal{G}} : \mathbb{R}^{1+n_\xi} \to \mathbb{R}^{n_\xi - n_{ddf}}$, *where the fixed number* $n_{ddf} \in \{0, \ldots, n_\xi\}$ *is the number of* dynamic degrees of freedom.

The underlying ODE of a DAE is not uniquely determined. However, different UODEs have the same solution for identical initial values [Eich 92].

Remark 1.8:
The underlying ODE as well as the hidden equations are equally contained in the derivative array equations; Algorithm 1 (or Algorithm 6 in Section 3.1.8) provides them in a more concise form. ◊

In the case of ODEs $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \dot{\boldsymbol{x}}(t))$, $\boldsymbol{f} : \mathbb{R}^{1+n_x} \to \mathbb{R}^{n_x}$, it is often a natural choice to specify initial conditions $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ with an arbitrary vector $\boldsymbol{x}_0$ in the domain of $\boldsymbol{f}$. More generally, depending on the Jacobian $\left[\partial \boldsymbol{f} / \partial \boldsymbol{x}\right]$ also initial conditions of the form $\dot{\boldsymbol{x}}(t_0) = \dot{\boldsymbol{x}}_0$ may be imposed with an arbitrary value of $\dot{\boldsymbol{x}}_0$ within the range of values of $\boldsymbol{f}$ (e.g., if admissible $\dot{\boldsymbol{x}}_0 = 0$ defines a stationary point as initial point), or an appropriate mixture of both types of initial conditions.

On the other hand, in the general DAE case not all vectors (or exactly, subsets of the components of the vectors) $\{\boldsymbol{\xi}_0, \dot{\boldsymbol{\xi}}_0\}$ in the domain of the mapping $\boldsymbol{F}(t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}})|_{t=t_0}$ are admissible values for specification as initial conditions as they have to satisfy the corresponding extended system Eq. (1.7). I.e., these vectors are constrained to the manifold defined by the original DAE *and* the hidden equations. Due to their special role the admissible vectors are called *consistent initial conditions.* More precisely, consistent initial conditions can be defined as [UKM 95]

Definition 1.11 (Consistent Initial Conditions)
The vectors $\boldsymbol{\xi}_0$ and $\dot{\boldsymbol{\xi}}_0$ *are called* consistent initial conditions *of the DAE* (1.1a) *at* t_0, *if they satisfy the corresponding extended system Eq.* (1.7) *at* $t = t_0$.

Remark 1.9:
Definition 1.11 avoids the distinction between the subset of the components of the vectors $\boldsymbol{\xi}_0$ and $\dot{\boldsymbol{\xi}}_0$ which can be specified externally and the complementary set of components which is fixed by the DAE. ◊

A definition of consistent initial conditions independent from the corresponding extended system and in accordance with Definition 1.7 (differential index of a DAE) following [Eich 92] is:

Definition 1.12 ((Strictly) Consistent Initial Conditions)
Let ι_d be the index of the DAE $\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) = 0$, $\boldsymbol{F} : \mathbb{R}^{1+n_\xi+n_\xi} \to \mathbb{R}^{n_\xi}$. Initial conditions $\boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0 \in \mathbb{R}^{n_\xi}$ *are called* (strictly) consistent, *if the system of equations*

$$\begin{aligned}
\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t))\Big|_{\substack{t=t_0 \\ \boldsymbol{\xi}=\boldsymbol{\xi}_0}} &= 0\,, \\
\frac{d}{dt}\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t))\Big|_{\substack{t=t_0 \\ \boldsymbol{\xi}=\boldsymbol{\xi}_0}} &= 0\,, \\
&\vdots \\
\frac{d^{\iota_d}}{dt^{\iota_d}}\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t))\Big|_{\substack{t=t_0 \\ \boldsymbol{\xi}=\boldsymbol{\xi}_0}} &= 0\,,
\end{aligned}$$

interpreting $\dot{\boldsymbol{\xi}}(t)|_{t=t_0}, \ldots, \boldsymbol{\xi}^{(\iota_d+1)}(t)|_{t=t_0}$ as variables $\dot{\boldsymbol{\xi}}, \ldots, \boldsymbol{\xi}^{(\iota_d+1)}$, *has a solution* $\dot{\boldsymbol{\xi}}(t_0, \boldsymbol{\xi}_0)$.

By construction, the corresponding extended system is equivalent to the derivative array equations in the sense that the former can be generated from the latter by repeated substitution and elimination. Therefore, a set of consistent initial conditions $\{\boldsymbol{\xi}_0, \dot{\boldsymbol{\xi}}(t_0, \boldsymbol{\xi}_0)\}$ according to Definition 1.12 is consistent according to Definition 1.11 with $\boldsymbol{\xi}_0 := \dot{\boldsymbol{\xi}}(t_0, \boldsymbol{\xi}_0)$, and vice versa.

Remark 1.10:
The conceptual difference between the two definitions of consistent initial conditions leads to two different methods for their computation: *Pantelides' Algorithm* – which is related to Definition 1.12 – is discussed in Section 3.1.5; in Section 3.1.8 we consider *Gear's Approach* which is based on Algorithm 1 and Definition 1.11. ◊

Definition 1.12 is based on the derivative array equations in connection with a set of initial conditions of the special form

$$\boldsymbol{k}^{\mathrm{ini}}(t_0, \boldsymbol{\xi}(t_0), \dot{\boldsymbol{\xi}}(t_0)) := \boldsymbol{\xi}(t_0) - \boldsymbol{\xi}_0 = 0\,.$$

The step to a more general specification is straightforward and gives rise to the definition of *consistency equations* in the sense of [LPG 91]

Definition 1.13 (Consistency Equations)
Let there be a DAE (1.1a) *and initial conditions* (1.1b). *Then the nonlinear system of equations composed of the derivative equations Eq.* (1.5) *of Eq.* (1.1a) *together with Eq.* (1.1b) *is called the* consistency equations *of the problem Eqs.* (1.1a)–(1.1b).

The corresponding extended system Eq. (1.7) of a DAE (1.1a) is a system of $n_{\boldsymbol{\xi}}$ explicit differential equations $\boldsymbol{\mathcal{F}} : \mathbb{R}^{1+n_\xi} \to \mathbb{R}^{n_\xi}$ and $n_{\boldsymbol{\xi}} - n_{\mathrm{ddf}}$ algebraic equations $\boldsymbol{\mathcal{G}} : \mathbb{R}^{1+n_\xi} \to \mathbb{R}^{n_\xi - n_{\mathrm{ddf}}}$ (the number of dynamic degrees of freedom n_{ddf} has been introduced in Definition 1.10). Therefore, n_{ddf} "appropriate" initial conditions

$\boldsymbol{k}^{\mathrm{ini}}(\cdot)$ specifying the initial state have to be added in order to obtain a square system, fixing n_{ddf} components of $\{\boldsymbol{\xi}(t_0), \dot{\boldsymbol{\xi}}(t_0)\}$. The choice of these equations – or of the components of the vectors $\boldsymbol{\xi}$, $\dot{\boldsymbol{\xi}}$ that can be assigned arbitrary initial values, respectively – is *not* free as we require a full rank system

$$\begin{aligned} \dot{\boldsymbol{\xi}}(t_0) &= \boldsymbol{\mathcal{F}}(t_0, \boldsymbol{\xi}(t_0)) \,, \\ 0 &= \boldsymbol{\mathcal{G}}(t_0, \boldsymbol{\xi}(t_0)) \,, \\ 0 &= \boldsymbol{k}^{\mathrm{ini}}(t_0, \boldsymbol{\xi}(t_0), \dot{\boldsymbol{\xi}}(t_0)) \,, \end{aligned}$$

which can be solved for the consistent initial conditions. This motivates [UKM 95]:

Definition 1.14 (Dynamic Degrees of Freedom)
Variables $\boldsymbol{\xi}$ or time derivatives $\dot{\boldsymbol{\xi}}$ which can be assigned arbitrary initial conditions and still allow consistent initialisation are called dynamic degrees of freedom.

For a DAE Eq. (1.1a) there may (and in general will) exist different subsets of the variables $\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}$ of cardinality n_{ddf} that are suitable for specification as the set of dynamic degrees of freedom. The non-uniqueness of this set of degrees of freedom in connection with the inherently combinatorial nature of the problem of specifying the dynamic degrees of freedom is one of the mayor difficulties in the automatic numerical computation of consistent initial conditions for general large-scale DAEs (see Section 3).

In [Camp 83], [BCP 96] a different concept of consistent initial conditions is introduced. There, initial conditions are said to be consistent if they admit a smooth solution of the DAE through the initial point:

Definition 1.15 ((Smoothly) Consistent Initial Conditions)
A vector $\boldsymbol{\xi}_0 \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$ is called a (smoothly) consistent initial condition *for the DAE Eq.* (1.1a) *at $t = t_0$ if there is a* differentiable *solution $\boldsymbol{\xi}(t)$ to Eq.* (1.1a)*, defined on an interval $[t_0, t_0 + \epsilon] \subset [t_0, t_f]$, $\epsilon > 0$, such that $\boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0$.*

Remark 1.11:
[Cobb 83] shows that even solvability in a distributive sense can be adequate in case of systems that suffer a change at $t = t_0$, e.g., by the opening or closing of switches in electric circuits. According to this definition, *arbitrary* initial conditions can be consistent. ◊

1.2 DAEs and Structural Calculus

Vestigia terrent.
The footprints frighten me.
(From a story about a fox who saw footprints lead into, but not out of a lion's den.)

Horace, Epistulae

As we will see later on, structural calculations are of special interest in the context of DAE analysis. In this section we give a summary of the structural calculus discussed in [UKM 95].

Structural calculus distinguishes *hard zeros* and *nonzeros* only [UKM 95]:

Definition 1.16 (Fundamental Assignment of Structural Values)
Whenever there is a numerical (or symbolical) representation of operands in $\mathbb{R}$ *such that a certain operation in* $\mathbb{R}$ *yields a nonzero result, then the result of the corresponding structural representation of this operation is "*$*$*". Conversely, if there is no such numerical (or symbolical) representation then the structural result is "0".*

Thus structural calculus can be regarded as the "crudest way of computation" [UKM 95]. Besides hard zeros and nonzeros in practice *soft zeros* have to be considered. Soft zeros originate from numerical operations which *appear* to deliver constantly zero as a result but which *may* as well generate a nonzero entry. These soft zeros give raise to problems at the interface from real-valued arithmetics to structural calculus as in such cases it is difficult to decide whether the numerical result of floating point operation is a blurred hard zero (with the nonzero value caused by round-off error) or whether it is actually a structural nonzero.

The notion of structural properties of operands applied to matrices leads to a characterisation of matrices by their (structural) *pattern* [UKM 95]:

Definition 1.17 (Pattern of a Matrix)
The representation of a matrix $M \in \mathbb{R}^{m \times n}$ *where nonzero elements are represented by "*$*$*" and zero elements are represented by "0" is called the* pattern of M *and is abbreviated by* $\mathrm{pat}([M])$.

Based on the pattern of a matrix the *structural rank* of a matrix is then defined by [UKM 95]

Definition 1.18 (Structural Rank of a Matrix)
Let $N_M \subset \mathbb{R}^{m \times n}$ *be the set of all possible numerical representations* M_μ *of* $\mathrm{pat}([M])$. *Let* $\mathrm{rank}\,([M_\mu])$ *be the numerical rank of* $M_\mu \in N_M$. *Then the maximum of all values* $\mathrm{rank}\,([M_\mu])$ *is called the* structural rank s_r *of* $\mathrm{pat}([M])$, *i.e.,* $s_r(\mathrm{pat}([M])) = \max_{M_\mu \in N_M} (\mathrm{rank}\,([M_\mu]))$.

By definition the structural rank of a matrix is an *upper bound* for the numerical rank of a matrix.

In practice the structural rank for a matrix-valued function given in dense representation has to be generated based on a small number of numerical evaluations and thus sample patterns of this matrix (consider, e.g., the determination of the structural rank of a Jacobian which is expensive to evaluate numerically). The problem is then the distinction between hard zeros, i.e., zero entries in the numerical representation of the matrix that are structurally zero, and soft zeros, i.e., zero

entries in the numerical representation of the matrix that are structurally nonzero. In case of matrices given in a sparse representation the (fixed) sparsity pattern of the matrix may be – and practically is – used as *the* structural pattern of the matrix, independent from the numerical values of the matrix entries. This elegant method allows a fast and reliable determination of the structural rank of a matrix, given that the sparse representation of the matrix is *carefully* implemented.

Example 1:
Consider the mapping

$$(\boldsymbol{x}_1, \boldsymbol{x}_2) \mapsto \begin{bmatrix} \boldsymbol{x}_1 + \boldsymbol{x}_1\boldsymbol{x}_2 \\ 0\boldsymbol{x}_2 \end{bmatrix},$$

and a sparse numerical representation of its Jacobian $\left[\begin{smallmatrix} 1 & \boldsymbol{x}_1 \\ \cdot & 0 \end{smallmatrix}\right]$ (the element "$\cdot$" is not contained in the sparse matrix), as well as the corresponding sparsity pattern $\left[\begin{smallmatrix} * & * \\ 0 & * \end{smallmatrix}\right]$. Depending on the value of $\boldsymbol{x}_1$ the upper right entry in the numerical Jacobian varies; in case of $\boldsymbol{x}_1 = 0$ this entry is a soft zero. Due to the entry in the lower right corner the structural rank of the Jacobian based on the sparsity pattern is 2, while the numerical rank of the Jacobian is 1. This bad result can be avoided by recognising the lower right entry as a hard zero and removing it from the sparse Jacobian. However, such unnecessary soft zeros are hard to detect numerically as apart from extreme cases (such as in this example) they do not affect sparse matrix solvers which are the most prominent numerical algorithms employing structural information. ◊

From Section 1.1 it is clear that we need a structural equivalent to differentiation if we intend to apply structural calculus in the context of DAEs. Unfortunately, there is no definition of a structural differentiation that provides the correct pattern of the Jacobian of a function in the general case unless higher order derivative information is used. The two extreme possibilities are *structurally nonlinear* and *structurally linear* differentiation which provide the *maximum* and the *minimum* number of nonzeros in the Jacobian of the differentiated equations, respectively [UKM 95]:

Definition 1.19 (Structurally Nonlinear/Linear Differentiation)
The structural differentiation of a function $\widetilde{G} : t \mapsto \widetilde{G}(t) := G(\boldsymbol{\xi}(t))$, $\widetilde{G} \in \mathcal{C}^1(\mathbb{R}, \mathbb{R}^{m_G})$ *with respect to time* t, *where* $G : \boldsymbol{\xi} \mapsto G(\boldsymbol{\xi})$, $G \in \mathcal{C}^1(\mathbb{R}^{n_{\boldsymbol{\xi}}}, \mathbb{R}^{m_G})$, *and* $\boldsymbol{\xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{n_{\boldsymbol{\xi}}}]^T \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$, $\boldsymbol{\xi}_\nu : t \mapsto \boldsymbol{\xi}_\nu(t)$, $\boldsymbol{\xi}_\nu \in \mathcal{C}^1(\mathbb{R}, \mathbb{R})$, $\nu = 1, \ldots, n_{\boldsymbol{\xi}}$, *is called a*

1. structurally nonlinear *differentiation if* $[d\widetilde{G}/dt] = [\partial G/\partial \boldsymbol{\xi}]\dot{\boldsymbol{\xi}}$ *is considered to depend on* all *variables* $\boldsymbol{\xi}_\nu$, $\nu = 1, \ldots, n_{\boldsymbol{\xi}}$, *appearing in* $G(\boldsymbol{\xi})$.

2. structurally linear *differentiation if* $[d\widetilde{G}/dt] = [\partial G/\partial \boldsymbol{\xi}]\dot{\boldsymbol{\xi}}$ *is considered to depend on* no $\boldsymbol{\xi}_\nu$, $\nu = 1, \ldots, n_{\boldsymbol{\xi}}$.

After these preparations central concepts of the analysis of DAEs can be restated in the structural sense [UKM 95]:

Definition 1.20 (Structural Index/Number of Degrees of Freedom)
The structural index ι_s and the structural number of degrees of freedom r_s of a DAE system $\boldsymbol{F}(\boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) = 0$, $\boldsymbol{F} : \mathbb{R}^{2n_\xi} \to \mathbb{R}^{n_\xi}$, are the index and the number of degrees of freedom obtained if all computations [connected with Definition 1.7 and Definition 1.10] are carried out according to the structural algebra defined above.

Structural index and structural degrees of freedom of a DAE can be shown to be independent of the kind of structural differentiation used [UKM 95]:

Proposition 1.1 (Structural Index and Jacobian Patterns)
The structural index ι_s and the structural number of degrees of freedom r_s of a DAE system $\boldsymbol{F}(\boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) = 0$, $\boldsymbol{F} : \mathbb{R}^{2n_\xi} \to \mathbb{R}^{n_\xi}$, is solely determined by $\mathrm{pat}([\partial \boldsymbol{F}/\partial \dot{\boldsymbol{\xi}}])$ *and* $\mathrm{pat}([\partial \boldsymbol{F}/\partial \boldsymbol{\xi}])$.

Consequences of Proposition 1.1 are:

1. For the computation of the structural properties ι_s and r_s it is sufficient to know $\mathrm{pat}([\partial \boldsymbol{F}/\partial \dot{\boldsymbol{\xi}}])$ and $\mathrm{pat}([\partial \boldsymbol{F}/\partial \boldsymbol{\xi}])$, i.e., the only information required is for each equation which of the variables and which of the time derivatives, respectively, *occur* in it.

2. ι_s and r_s of a nonlinear DAE are identical to those of the linearisation of the DAE around a solution trajectory.

Until recently it was commonly asserted that the structural index of a DAE is equal to or smaller than its differential index [Unge 90], [UKM 95], [BCP 96]. [RMB 00] have found examples originating from the simulation of DAE models of electric circuits with differential index 1 and arbitrarily high structural index.

However, structural algebra implemented on computers is *per se* free of round-off errors and can be used as a basis for very efficient algorithms. Moreover, currently structural calculus provides the only numerical technique applicable for the analysis of large-scale DAEs.

1.3 Discontinuities

Natura non facit saltus.
(Nature doesn't make any leaps.)

Carl von Linné: Philosophia botanica

According to [Cell 79], [Bart 92], [BaPa 94], processing systems in general own a combined discrete/continuous structure. More precisely, nearly all processes of our interest are principally of continuous nature, more or less frequently subject to discontinuous changes. Therefore, discontinuities have to be considered in modelling, simulation, and optimal control. *Combined discrete/continuous systems* are also called *hybrid dynamical systems.*

[ScSc 99] give a general introduction into hybrid dynamical systems. A review of the developments in the area of hybrid dynamical systems with emphasis on chemical process control can be found in [EKK 97].

As a first preparation we define discontinuities in dynamical systems, extending [HaGr 80]:

Definition 1.21 (Discontinuity)
A discontinuity *in a dynamical system is an instantaneous transition from one state of the system to another. As the discontinuity occurs the set of model equations (and/or external controls – if present –) describing the system may change so as to represent the new system state rather than the old state.*

From the modelling point of view, hybrid dynamical systems can be described, e.g., by the following formalism summarised in Definition 1.22 (see [GaBa 98], [GFB 99]). However, other formalisms are also considered, cf., e.g., [BSBS 00].

Definition 1.22 (Description of Hybrid Dynamical Systems)
A hybrid dynamical system *is described by a* state space $S = \bigcup_{\mu=1}^{m_S} S_\mu$ *where each* mode S_μ, $\mu = 1, \ldots, m_S$, *is characterised by*

1. *the independent variable t (time), a set of state variables $\boldsymbol{\xi}^{(\mu)} \in \mathbb{R}^{n_{\boldsymbol{\xi}}^{(\mu)}}$ (and their derivatives with respect to time), controls $\boldsymbol{u}^{(\mu)} \in \mathbb{R}^{n_{\boldsymbol{u}}^{(\mu)}}$, and time independent parameters $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$,*

2. *a set of equations*

$$0 = \boldsymbol{F}^{(\mu)}(t, \boldsymbol{\xi}^{(\mu)}(t), \dot{\boldsymbol{\xi}}^{(\mu)}(t), \boldsymbol{u}^{(\mu)}(t), \boldsymbol{p}) \,,$$

   $\boldsymbol{F}^{(\mu)} : \mathbb{R}^{1+2n_{\boldsymbol{\xi}}^{(\mu)}+n_{\boldsymbol{u}}^{(\mu)}+n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}^{(\mu)}}$, which in connection with a consistent initial condition

$$0 = \boldsymbol{k}^{ini(\mu)}(t, \boldsymbol{\xi}^{(\mu)}(t), \dot{\boldsymbol{\xi}}^{(\mu)}(t), \boldsymbol{u}^{(\mu)}(t), \boldsymbol{p}) \,,$$

   $\boldsymbol{k}^{ini(\mu)} : \mathbb{R}^{1+2n_{\boldsymbol{\xi}}^{(\mu)}+n_{\boldsymbol{u}}^{(\mu)}+n_{\boldsymbol{p}}} \to \mathbb{R}^{m_{\boldsymbol{k}^{ini}}^{(\mu)}}$, at $t = t_0^{(\mu)}$ describes the system dynamics within an interval $[t_0^{(\mu)}, t_f^{(\mu)}[$,

3. *and a possibly empty set of transitions $J^{(\mu)} \subset \{1, \ldots, m_S\}$ to other modes S_ν. A transition from mode S_μ to mode*[vi] *S_ν is possible if $\nu \in J^{(\mu)}$.*

[vi] *The case that a mode S_ν may be reached from mode S_μ in different ways can be included by extending the notation to incorporate subindices.*

Each potential transition (from mode S_μ to mode S_ν, $\nu \in J^{(\mu)}$) is characterised by a switching condition [(vii)]

$$\boldsymbol{Q}_\nu^{(\mu)}(t, \boldsymbol{\xi}^{(\mu)}(t), \dot{\boldsymbol{\xi}}^{(\mu)}(t), \boldsymbol{u}^{(\mu)}(t), \boldsymbol{p}) \,,$$

$\boldsymbol{Q}_\nu^{(\mu)} : \mathbb{R}^{1+2n_{\boldsymbol{\xi}}^{(\mu)}+n_{\boldsymbol{u}}^{(\mu)}+n_{\boldsymbol{p}}} \to \{true, false\}$, *determining the transition time* $t = t_{f,\nu}^{(\mu)}$ *(the transition is triggered when* $\boldsymbol{Q}_\nu^{(\mu)}$ *becomes true), and a* transition function *(also called* transition condition, *or* jump function*)*

$$0 = \boldsymbol{K}_\nu^{(\mu)}(t, \boldsymbol{\xi}^{(\nu)}(t), \dot{\boldsymbol{\xi}}^{(\nu)}(t), \boldsymbol{u}^{(\nu)}(t), \boldsymbol{\xi}^{(\mu)}(t), \dot{\boldsymbol{\xi}}^{(\mu)}(t), \boldsymbol{u}^{(\mu)}(t), \boldsymbol{p}) \,,$$

$\boldsymbol{K}_\nu^{(\mu)} : \mathbb{R}^{1+2n_{\boldsymbol{\xi}}^{(\nu)}+n_{\boldsymbol{u}}^{(\nu)}+2n_{\boldsymbol{\xi}}^{(\mu)}+n_{\boldsymbol{u}}^{(\mu)}+n_{\boldsymbol{p}}} \to \mathbb{R}^{m_{\boldsymbol{K}_\nu^{(\mu)}}}$, *connecting the variables at the end of mode* S_μ *with the variables of* S_ν *at its start. As a special case, the transition function*

$$0 = \boldsymbol{K}_1^{(0)}(t_0, \boldsymbol{\xi}^{(1)}(t_0), \dot{\boldsymbol{\xi}}^{(1)}(t_0), \boldsymbol{u}^{(1)}(t_0), \boldsymbol{p}) \,,$$

$\boldsymbol{K}_1^{(0)} : \mathbb{R}^{1+2n_{\boldsymbol{\xi}}^{(1)}+n_{\boldsymbol{u}}^{(1)}+n_{\boldsymbol{p}}} \to \mathbb{R}^{m_{\boldsymbol{K}_1^{(0)}}}$, *determines the initial conditions of the initial mode* S_1 *at* $t_0 = t_0^{(1)}$.

The switching condition $\boldsymbol{Q}_\nu^{(\mu)}$ *is composed by logical expressions. In practice,* $\boldsymbol{Q}_\nu^{(\mu)}$ *will be replaced by an equivalent scalar switching function* $\boldsymbol{q}_\nu^{(\mu)}$ *indicating the transition by a sign change.*

If there are different possible transitions starting from mode S_μ the active transition is indicated by the earliest root in any of the scalar switching functions associated to mode S_μ. The special case of different scalar switching functions $\boldsymbol{q}_\nu^{(\mu)}$ having an identical root is pathological in the deterministic framework given by Definition 1.22.

Remark 1.12:
Note that Definition 1.22 is an abstract description for hybrid systems and especially the switching functions $\boldsymbol{q}_\nu^{(\mu)}$ have to be distinguished from practically implemented switching functions. In practice (cf. the discussion on the location of discontinuities below) it may be the case that two or more switching functions vanish at the same time. But as in application different *parts* of the model are affected coinciding roots do not cause a contradiction. The point is that is such a case $\mu \in \mathbb{N}$ "practical" switching functions correspond to 2^μ modes with accordingly defined "theoretical" switching functions. ◊

As solvability in the sense of Definition 1.2 does not directly apply to hybrid DAE systems, we extend it for such systems in an intuitive way:

[(vii)] [GaBa 98] and [GFB 99] use the term *transition condition* instead of the term *switching condition*. We applied this modification in order to avoid confusion with the *transition function* introduced below. Note that we will use the terms *transition function* and *transition condition* synonymously.

Definition 1.23 (Solvability / Solution of Hybrid DAE Systems)
A hybrid dynamical system $S = \bigcup_{\mu=1}^{m_S} S_\mu$ *as given in Definition 1.22 is* solvable *if each dynamical system* $0 = \boldsymbol{F}^{(\mu)}$ *is solvable in the sense of Definition 1.2 on the time interval* $I_\mu = [t_0^{(\mu)}, t_f^{(\mu)}[$ *in a suitable domain* Ω_μ*, and if the transition functions* $\boldsymbol{K}_\nu^{(\mu)}$ *(including* $\boldsymbol{K}_1^{(0)}$*) together with the initial conditions* $\boldsymbol{k}^{ini(\mu)}$ *allow the determination of consistent initial conditions at the transition times and at the initial time, respectively.*

A function $\boldsymbol{\xi} : t \mapsto \boldsymbol{\xi}(t), [t_0, t_f] \to \mathbb{R}^{n_\xi}$ *is called a* solution *of the hybrid system if there is a decomposition of the interval* $[t_0, t_f]$ *such that for each partial interval there exists a dynamical system* $0 = \boldsymbol{F}^{(\mu)}$ *with the property that* $\boldsymbol{\xi}$ *restricted to this partial interval is a solution of* $0 = \boldsymbol{F}^{(\mu)}$ *in the sense of Definition 1.1, and if all active switching, transition, and initial conditions are satisfied in the entire interval* $[t_0, t_f]$.

The numerical problems arising in the simulation of dynamical systems with discontinuities are well known, see, e.g., [Cell 79], [GeOs 84], [Eich 92], [ToBa 02]. Plain integration ignoring a discontinuity can lead to gross inefficiency, unreliable results, or failure of advanced (variable-order, variable step-size) integrator algorithms. Thus, fully satisfactory and reliable results can only be achieved if each discontinuity is located and integration is restarted in consequence.

Discontinuities which only depend on time (*time events*) can be handled easier than state-dependent discontinuities (*state events*) as time events are essentially known a priori, rendering costly location unnecessary. Therefore, the discussion below primarily addresses state events (although time events can be handled in the same way as state events – if efficiency is not of interest).

1.3.1 Hidden Discontinuities

In the worst case explicit information about a discontinuity is missing. I.e., in the context of Definition 1.22 the discontinuity is actually *executed*, e.g., within a lower level model routine, but this event is *not signalled* by the means of a public switching function. As such discontinuities are hidden within the simulation code they are termed *hidden discontinuities* [ToBa 02].

There are primarily two reasons for the existence of hidden discontinuities [ToBa 02]:

- On the one hand, modelling with switching functions requires considerable effort during the implementation of a model functionality (see the next section on switching functions).

- On the other hand, the discontinuous part of the model behaviour may not be of major interest. In this case the integrator is expected to cope with the discontinuities internally, accepting the possible drawbacks of such a method as mentioned above.

For ODE models given as black-box routines without switching function information [GeOs 84] developed a method for the location of discontinuities based on the local error estimation formulas of linear multistep methods. However, due to its inherent limitations [GeOs 84] regard their method as a remedy only and recommend the application of switching functions whenever possible.

Hidden discontinuities can also significantly deteriorate numerically computed parametric sensitivity information. As the transition times cannot be located exactly, in a consequence the correct transition of the sensitivities across these discontinuities cannot be performed (methods for the transition of sensitivities across discontinuities are discussed in Chapter 4). Therefore, [ToBa 02] propose to generate the switching function code automatically in order to tackle with the problem of hidden discontinuities in its origin. They have modified an automatic differentiation tool in order to analyse the model code for potential discontinuities and then add code for switching function handling.

1.3.2 Location of Discontinuities Using Switching Functions

Switching function based methods as considered in [HaGr 80] are

1. to assume that the discontinuity has taken place at the *end* of the integration step in which a sign change in the switching functions has been detected.

   This is a very simple but also dangerous method, as it produces unreliable results especially for larger step-sizes. Due to the demand for highly precise, reliable, and efficient simulation this method is in general inappropriate for use in state-of-the-art integrators.

   Remark 1.13:
   [Otte 95] explicitly uses *step events* which are *defined* as discontinuities that can become active at the end of a successful executed integration step only. ◊

2. to locate the discontinuity more precisely within the integration interval where a change of sign of a switching function has been detected.

   This can be achieved by, e.g., inverse interpolation. In this method the trajectory of the active switching function is interpolated on the basis of a few evaluations within the interval of interest. Assuming that the interpolation is sufficiently exact the roots of the interpolating polynomial are then taken as the roots of the switching function, cf., e.g., [Cell 79], [Eich 92]. Other methods for the localisation of discontinuities in ODE and DAE systems are discussed in, e.g., [Krön 02].

A widely used technique for the exact and efficient location of discontinuities is *discontinuity locking* [PBa 96] (cf. Figure 1.1). In this technique a discontinuity is monitored by the sign change in at least one of the switching functions *after* a successfully executed integration step, i.e., the signs of the switching functions

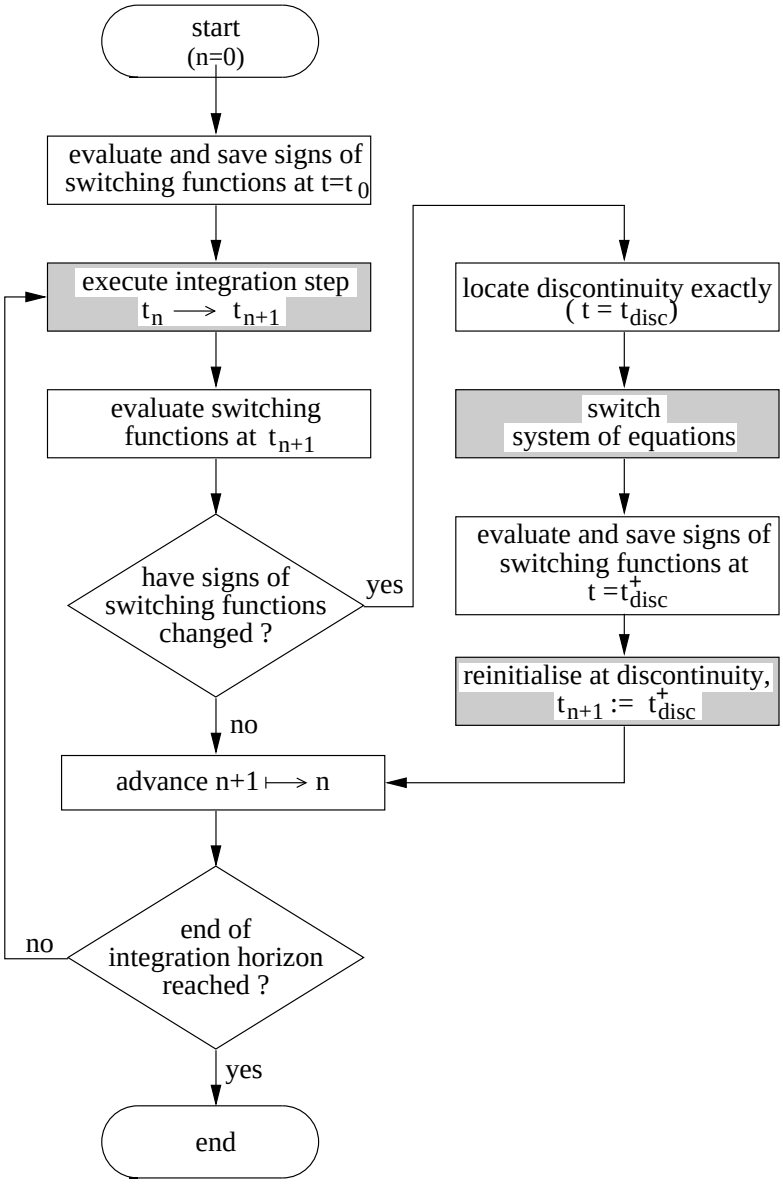

Figure 1.1: Integration of discontinuous systems using the discontinuity locking technique (sketch). The main steps are highlighted.

at time t_n and t_{n+1} are compared. The fundamental assumption justifying this comparison is that the value of the state vector calculated for time t_{n+1} in the trial integration step is at least numerically "reasonable" *although* the model equations have not been switched (thus the name *discontinuity locking*). The exact point of discontinuity can then be located by a root-finding algorithm employing, e.g., inverse interpolation in order to increase the efficiency of the method. In case

of more than one switching function exhibiting a root within the integration interval the earliest root is used. After this the model equations are switched and integration is restarted at the discontinuity.

For the developer of a model unit the discontinuity locking technique results in several restrictions and in additional effort (unless the automated switching function generation method of [ToBa 02] is employed) [Eich 92], [ScWi 94]:

- Discontinuous effects have to be expressed in terms of switching functions (cf. Definition 1.22).

- Decisions about which of the model equations are to be used are allowed on the basis of an additional discrete state vector only. I.e., the discrete state vector contains the authoritative information about the active model variant.

- This discrete state vector must not be modified by any routines apart from the switching function driver.

  In an implementation the switching of the model equations (see right branch of Figure 1.1) is realised by a modification of the discrete state vector. The new value of the discrete state vector is determined by the switching function driver based on the actual state variable vector. Other changes in the discrete state vector are not permitted.

  Remark 1.14:
  In a consequence the switching function driver has to be provided with the required modelling level knowledge. ◊

[Eich 92] presents a refined inverse interpolation method which is especially designed for application within linear multistep integrators (e.g., BDF algorithms). There, a considerable amount of computational time is preserved by employing the *natural interpolation polynomial* of the discretised state variable vector in order to compute the values of the switching functions required for inverse interpolation.

Additionally, [Eich 92] treats the problem of *inconsistent switching*[(viii)] by introducing a three-valued switching logic under application of the notion of generalised solutions of differential equations according to Filipov. Inconsistent switching is given if the state trajectory does not cross the *switching manifold* in state space (given by the switching functions) at the point of the discontinuity, staying on this manifold for a non-trivial interval in time.

As examined in [PBa 96] the treatment of discontinuities in DAE models is aggravated by potential *discontinuity sticking*. Discontinuity sticking is caused by the interpolation of the state variables applied in the discontinuity location phase. Due to the interpolation error (which is reflected in the missing consistency in

[(viii)] The term *inconsistent switching* is not related to the notion of consistency in the context of DAEs.

the interpolated state variable values) the discontinuity may not be located precisely enough. Therefore, in the next integration step the same discontinuity may be triggered again which necessitates the repeated treatment of the discontinuity. In order to overcome this problem [PBa 96] extend the discontinuity detection and location algorithm. Core part is the *consistent event location*, i.e., the point of the discontinuity is determined simultaneously with the corresponding consistent states at the discontinuity before the discontinuity is executed. According to [Krön 02], however, discontinuity sticking is rarely observed in non-academic examples.

In practice, additional issues have to be considered:

The discontinuity location procedures in general assume that the discontinuities are indicated either by a sign change or by a switching function becoming exactly zero, starting from a non-zero value. In order to avoid the problem when the switching function is exactly zero at the start of the integration step, [Otte 95] introduces a small ϵ-band around the switching functions.

If a switching function owns an even number of roots within an integration step the corresponding discontinuities cannot be detected as the sign of the switching function is the same at both the start and the end of the step. Therefore, [Cell 79] proposes to consider (higher order) time derivatives of the switching function as well. Then Rolle's Theorem provides a means to locate the first change of sign of the original switching function by iteratively excluding a of a part of the integration time step, starting from exactly one root of a higher order derivative of the switching function and going down to the undifferentiated switching function. However, the maximum number of oscillations within an integration interval has to be known in advance. Additionally, higher order time derivatives of the switching functions have to be provided.

Finally, simultaneous switching events have to be considered. Especially, the execution of an event can immediately trigger another event, e.g., in *boolean arrays* (or *deterministic finite automata*). In such a case switching has to be repeated until no further change in a switching function occurs [Otte 95]. Special measurements to prevent dead-locks have to be taken.

1.4 Sensitivity Analysis for DAEs

Dicit ei Simon Petrus "Domine, quo vadis?" [...]
(Simon Peter asked him, "Lord, where are you going?"[...])

The Bible, John 13:36

Mathematical models of real-world problems in most cases contain parameters. Examples are the mass of a pendulum (see Section 6.2.1), heat transfer parameters in a heat exchanger (see Example 6 on page 131), or parameters introduced by application of the direct single shooting method for the numerical solution of

optimal control problems where control variables are replaced by parameterised functions (see Section 2.4.2).

Now let there be a parameter dependent DAE-IVP

$$\boldsymbol{F}(t, \boldsymbol{\xi}(t;\boldsymbol{p}), \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}), \boldsymbol{p}) = 0\,; \quad t \in [t_0, t_\mathrm{f}] \subset \mathbb{R}, \tag{1.8a}$$

$$\boldsymbol{k}^\mathrm{ini}(t_0, \boldsymbol{\xi}(t_0;\boldsymbol{p}), \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}), \boldsymbol{p}) = 0\,, \tag{1.8b}$$

with the model $\boldsymbol{F} : \mathbb{R}^{1+2n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$, initial conditions $\boldsymbol{k}^\mathrm{ini} : \mathbb{R}^{1+2n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{m_{\boldsymbol{k}^\mathrm{ini}}}$, and the constant vector of parameters $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$, $n_{\boldsymbol{p}} \in \mathbb{N}$. The notation $\cdot(t;\boldsymbol{p})$ is used to indicate the dependence of both the state variable trajectories $\boldsymbol{\xi}(\cdot)$ and of their first order derivatives with respect to time $\dot{\boldsymbol{\xi}}(\cdot)$ from the vector of parameters $\boldsymbol{p}$ (this notation may also be applied to other quantities than $\boldsymbol{\xi}(\cdot)$ and $\dot{\boldsymbol{\xi}}(\cdot)$ in the sequel).

In the context of parameter estimation, design optimisation, optimal control, model reduction, and experimental design knowledge on the dependency of the solution trajectories of the parameter dependent DAE-IVP Eqs. (1.8a)–(1.8b) from the values of the parameters is required, cf., e.g., [PSLRC 01]. This investigation is called the *sensitivity analysis* of the DAE-IVP Eqs. (1.8a)–(1.8b), e.g., in slight modification of [Gerd 01]:

Definition 1.24 (Sensitivity Analysis for DAEs)
The investigation of the dependency of the solution of a parameter dependent DAE-IVP Eqs. (1.8a)–(1.8b) *with consistent initial conditions* $\boldsymbol{\xi}(t_0;\boldsymbol{p})$, $\dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p})$ *from the parameters* $\boldsymbol{p}$ *is called the* sensitivity analysis *for the DAE-IVP Eqs.* (1.8a)–(1.8b)

Given differentiability of the solution trajectory $\boldsymbol{\xi}(t;\boldsymbol{p})$, $t \in [t_0, t_f]$, *with respect to* $\boldsymbol{p}$, *sensitivity analysis for the DAE-IVP Eqs.* (1.8a)–(1.8b) *means to calculate the (parametric)* sensitivity functions, sensitivity matrices, *or shortly, the (parametric)* sensitivities

$$\boldsymbol{\omega}(t;\boldsymbol{p}) := \left[\frac{\partial \boldsymbol{\xi}(t;\boldsymbol{p})}{\partial \boldsymbol{p}}\right] \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{p}}}\,.$$

Thus the sensitivities represent the linearisation of the trajectories around a nominal parameter value. E.g., consider a trajectory $\boldsymbol{\xi}(t;\boldsymbol{p}_0)$ and the corresponding sensitivity functions $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t;\boldsymbol{p}_0)$ evaluated at the nominal value of the parameter $\boldsymbol{p}_0$. Then for a small disturbance $\Delta\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$ in the parameter a first order approximation of the trajectory $\boldsymbol{\xi}(t;\boldsymbol{p}_0 + \Delta\boldsymbol{p})$ is given by

$$\boldsymbol{\xi}(t;\boldsymbol{p}_0 + \Delta\boldsymbol{p}) \doteq \boldsymbol{\xi}(t;\boldsymbol{p}_0) + \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t;\boldsymbol{p}_0)\Delta\boldsymbol{p}\,.$$

In the case of a parameter dependent ODE-IVP without discontinuities a detailed discussion of sensitivity analysis including proofs for existence and uniqueness of the sensitivity matrices can be found in [HNW 87]. Existence of the sensitivities in the case of semi-explicit index-1 DAEs without discontinuities has been

shown, e.g., in [Heim 92]. [Gerd 01] proves existence for general nonlinear DAEs of finite, but otherwise arbitrary index. Sensitivities for discontinuous ODEs are treated in, e.g., [ScSc 99]. [Feeh 98] addresses existence of sensitivities for ODEs and index-1 systems with discontinuities. A detailed discussion of the problem of computing sensitivity functions for discontinuous nonlinear ODEs, nonlinear index-1 DAEs, and semi-explicit index-2 DAEs is the subject of Chapter 4.

Three different approaches have been investigated for the numerical approximation of parametric sensitivity functions for DAE models: *Numerical differentiation (finite difference approximations)* (e.g., [BHK 94], [Grup 96], [SWS 95]), backward *integration of the adjoint equations* (e.g., [MoSa 86], [BeSe 92], [GPS 95], [BBiS 99], [CLPS 00], [CLP 00]), and *integration of the (associated) sensitivity equations* (e.g., [CaSt 85], [LeKr 85], [MaPe 96], [HeSt 96], [FTB 97]). For a comparison of these methods see, e.g., [RoLu 91], [PSLRC 01]. According to [BSS 95] numerical algorithms for the approximation of sensitivity matrices can be divided into two classes:

1. External numerical differentiation methods (*END*):

   END obtains the sensitivity functions basically independent from the numerical integration of the original state variable trajectories. E.g., the sensitivity equations may be integrated independently from the model equations. This approach has the advantage that the integrator may be used as a black box.

2. Internal numerical differentiation methods (*IND*):

   In the IND approach the integrator scheme employed for the numerical integration of the model equations is differentiated, and a modified integration scheme for the combined calculation of both state variable trajectories and sensitivities is constructed. In this way suitable approximations to the sensitivities of the *numerical* solution may be obtained even for loose integration precisions. In contrast to END, IND is stable in the sense of backward error analysis [BSS 95]. However, deeper analysis of the integrator algorithm and extentions of its program code are necessary. E.g., in Section 2.4.3.b and Section 5.2 we discuss the implementation of an IND algorithm for the integration of the sensitivity equations.

In general, *internal numerical differentiation* is superior to *external numerical differentiation* in terms of reliability and efficiency, cf., e.g., [BSS 95], [Kieh 99]. However, IND is not applicable in every case, especially if the *code* of the algorithm employed for the integration of the DAE-IVP Eqs. (1.8a)–(1.8b) cannot be modified, while END can be performed given access to an integrator as a method only.

Due to practical reasons we employ integration of the sensitivity equations, cf. Section 2.4.3. Formally, the sensitivity equations can be obtained by total differentiation of the DAE-IVP Eqs. (1.8a)–(1.8b) with respect to the parameters

$\boldsymbol{p}$ along a solution trajectory $\boldsymbol{\xi}(t;\boldsymbol{p})$, $t \in [t_0, t_\mathrm{f}]$:

$$
\begin{aligned}
0 &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{p}} \boldsymbol{F}(t, \boldsymbol{\xi}(t;\boldsymbol{p}), \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}), \boldsymbol{p}) \\
&= \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\xi}}\right]_{\substack{t,\boldsymbol{\xi}(t;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}),\boldsymbol{p}}} \cdot \left[\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t;\boldsymbol{p})\right] + \left[\frac{\partial \boldsymbol{F}}{\partial \dot{\boldsymbol{\xi}}}\right]_{\substack{t,\boldsymbol{\xi}(t;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}),\boldsymbol{p}}} \cdot \left[\frac{\partial \dot{\boldsymbol{\xi}}}{\partial \boldsymbol{p}}(t;\boldsymbol{p})\right] + \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{p}}\right]_{\substack{t,\boldsymbol{\xi}(t;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}),\boldsymbol{p}}} \\
&= \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\xi}}\right]_{\substack{t,\boldsymbol{\xi}(t;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}),\boldsymbol{p}}} \cdot \boldsymbol{\omega}(t;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{F}}{\partial \dot{\boldsymbol{\xi}}}\right]_{\substack{t,\boldsymbol{\xi}(t;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}),\boldsymbol{p}}} \cdot \dot{\boldsymbol{\omega}}(t;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{p}}\right]_{\substack{t,\boldsymbol{\xi}(t;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}),\boldsymbol{p}}}, \qquad (1.9a)
\end{aligned}
$$

$$
\begin{aligned}
0 &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{p}} \boldsymbol{k}^{\mathrm{ini}}(t_0, \boldsymbol{\xi}(t_0;\boldsymbol{p}), \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}), \boldsymbol{p}) \\
&= \left[\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{\xi}}\right]_{\substack{t_0,\boldsymbol{\xi}(t_0;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}),\boldsymbol{p}}} \cdot \left[\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t_0;\boldsymbol{p})\right] + \left[\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \dot{\boldsymbol{\xi}}}\right]_{\substack{t_0,\boldsymbol{\xi}(t_0;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}),\boldsymbol{p}}} \cdot \left[\frac{\partial \dot{\boldsymbol{\xi}}}{\partial \boldsymbol{p}}(t_0;\boldsymbol{p})\right] + \left[\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{p}}\right]_{\substack{t_0,\boldsymbol{\xi}(t_0;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}),\boldsymbol{p}}} \\
&= \left[\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{\xi}}\right]_{\substack{t_0,\boldsymbol{\xi}(t_0;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}),\boldsymbol{p}}} \cdot \boldsymbol{\omega}(t_0;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \dot{\boldsymbol{\xi}}}\right]_{\substack{t_0,\boldsymbol{\xi}(t_0;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}),\boldsymbol{p}}} \cdot \dot{\boldsymbol{\omega}}(t_0;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{p}}\right]_{\substack{t_0,\boldsymbol{\xi}(t_0;\boldsymbol{p}) \\ \dot{\boldsymbol{\xi}}(t_0;\boldsymbol{p}),\boldsymbol{p}}}. \qquad (1.9b)
\end{aligned}
$$

An important property of the sensitivity equations Eqs. (1.9a)–(1.9b) is that they always form a linear DAE-IVP in the sensitivities independent from the nonlinearity of the model DAE [CaSt 85]. Due to Proposition 1.1 on page 14 the sensitivity DAE owns the same structural index as the model DAE.

As pointed out by [CLP 00] there are pathological cases in which the sensitivity DAE-IVP Eqs. (1.9a)–(1.9b) is not solvable although the original DAE-IVP Eqs. (1.8a)–(1.8b) is well defined. This can be the case if the original DAE-IVP is solvable for a *special* choice of the parameters only, rendering parametric sensitivity information meaningless. Therefore we assume throughout this treatise that the DAE-IVPs considered are well-posed in the sense of Definition 1.25, following [CLP 00]:

Definition 1.25 (Well-Posed Parameter Dependent DAE-IVP)
The parameter dependent problem Eqs. (1.8a)–(1.8b) *is called* well-posed *if for the given vector of parameters* $\boldsymbol{p}$ *there exists a value* $\epsilon \in \mathbb{R}_+ \setminus \{0\}$ *and a neighbourhood* $\mathcal{N}_\epsilon(\boldsymbol{p}) := \{\bar{\boldsymbol{p}} \in \mathbb{R}^{n_p} \,|\, \|\bar{\boldsymbol{p}} - \boldsymbol{p}\| < \epsilon\}$ *around* $\boldsymbol{p}$ *so that for any vector of parameters* $\bar{\boldsymbol{p}} \in \mathcal{N}_\epsilon(\boldsymbol{p})$ *the DAE-IVP Eqs.* (1.8a)–(1.8b) *has a unique solution* $\boldsymbol{\xi}(t;\boldsymbol{p})$, $t \in [t_0, t_\mathrm{f}]$.

Additionally, in order to perform parametric sensitivity analysis we assume that for each fixed parameter $\boldsymbol{p}$ of interest the stronger condition of differentiable parametric dependency of the trajectory $\boldsymbol{\xi}(t;\boldsymbol{p})$ holds on a neighbourhood $\mathcal{N}_\epsilon(\boldsymbol{p})$, $\epsilon > 0$, around $\boldsymbol{p}$.

Chapter 2

Aspects of Nonlinear Model-Based Predictive Optimal Control with Large-Scale Process Models

2.1 Application and Background

2.1.1 An Industrial Optimal Control Challenge

Gases like nitrogen (N_2), oxygen (O_2), and argon (Ar) are used in industry as raw materials or as auxiliary substances. E.g., nitrogen is the basis of fertilisers, oxygen is required for the refinement of steel, and argon is needed for welding. At the same time, these three gases are the main components of air (see Table 2.1). Thus a natural idea is to obtain such industrial gases by the separation of *air* which is available at any place on earth in sufficient amounts.

nitrogen	N_2	78.084	[vol%]
oxygen	O_2	20.946	[vol%]
argon	Ar	0.932	[vol%]
carbon dioxide	CO_2	≈ 335	[vppm]
neon	Ne	18.18	[vppm]
helium	He	5.239	[vppm]
krypton	Kr	1.14	[vppm]
xenon	Xe	0.086	[vppm]

Table 2.1: Average composition of air (without variable constituents [Rohd 94]).

In 1902 Dr. Carl von Linde, the founder of the Linde AG, built the world's first commercially viable cryogenic air separation plant. This type of air separation plants is based on liquefaction of ambient air and rectification (which can be regarded as a highly efficient form of distillation) in order to separate the various fractions contained in the feed. In industrial application the energy consumption required for cooling to the prescribed deep temperatures governs the operational costs of the process. Therefore, the different processing parts in such air separation

plants are strongly interconnected in order to keep as much of the energy within the process as possible.

Nowadays, air separation plants are often integral parts of other industrial production facilities. Their capacities range from 25 [t/d] (tons per day) to 15000 [t/d] of processed air. E.g., the plant depicted in Figure 2.1 consists in two identical cryogenic air separation plants and has a capacity of $2 \times 2150\,[\mathrm{t/d}]$ gaseous O_2 in 99.5 [%] purity and $2 \times 600\,[\mathrm{t/d}]$ gaseous N_2 with a maximum impurity of 10 [ppm] O_2.

Figure 2.1: An air separation plant (picture by Linde AG).

Currently, in Mexico the world's largest nitrogen generation facilities provide $4 \times 335000\,[\mathrm{Nm^3/h}]$ ($\approx 4 \times 10000\,[\mathrm{t/d}]$) pure nitrogen [Lind 98]. The nitrogen is injected into an oil well in order to increase crude oil production. The facilities consist of four identical cryogenic air separation plants built by the Linde AG, each of which is larger than any other currently existing air separation plant. The energy required is generated by an auxiliary 550 [MW] power plant.

By the arguments given above, modern air separation plants are very complex industrial facilities with increasingly growing demands on productivity, safety, and efficiency. In consequence, construction and operation must rely more and more on numerical simulation.

A critical phase in the control of these plants occurs when a load-change, i.e., a transition from one operational point to another one (e.g., from 100% to 60% air throughput) must be performed. It is of utmost importance that various constraints on gas concentrations at certain points in the plant are not violated in order to guarantee safe operation of the plants and purity of the products also during the load-change (as an example the data for an existing air separation plant are given in Table 2.2; see also, e.g., Section 6.3.5, Section 6.3.6). Other objectives such as maximisation of product gain or minimisation of energy consumption are of minor importance. During a load-change the controls of the process essentially consist of continuously working valves (see Figure 2.2).

The numerical treatment of this difficult problem using the powerful tools of

Table 2.2: Example 2: Data on constraints to be satisfied during load-changes of an existing air separation plant of the type shown in Figure 2.2 (cf., e.g., [EKKS 99]).

name	min [mol/mol]	max [mol/mol]	description
O_2 LOX	0.997	1.0	O_2 fraction in liquid O_2 product
O_2 GOX	0.997	1.0	O_2 fraction in gaseous O_2 product
O_2 DLIN	0.0	$5.0 \cdot 10^{-6}$	O_2 fraction in liquid N_2 product
O_2 GAN	0.0	$5.0 \cdot 10^{-6}$	O_2 fraction in gaseous N_2 product
Ar Prod	0.965	1.0	Ar fraction in Ar product
O_2 feed ArC	0.90	1.0	O_2 fraction in feed of argon column

numerical optimal control (for an overview see, e.g., [BBBB 01]) provides the basic motivation for this thesis; though, the techniques presented here may also apply to other optimal control problems.

2.1.2 The Basics of Cryogenic Air Separation Plants

The development of efficient solution procedures for difficult optimal control problems originating from industrial real-life applications requires sound knowledge of the engineering background as well as of the physics (or chemistry, biology, ...) of the underlying application. Therefore we discuss in this section the basics of *cryogenic low pressure air separation plants*, cf., e.g., [BBKS 83], [Rohd 94], [Voit 94]. The different operational groups introduced below can be found in Figure 2.2.

Cryogenic air separation basically works in the following way:

1. In the feed preparation section the feed air is compressed, pre-cooled, and washed. Water and carbon dioxide are removed in molecular sieves.

2. The dry air is cooled near to the dew point ($\approx -170°$C). Heat is transferred in the main heat exchanger. The cooling duty originates from the cold products leaving the process and from compression/expansion processes. (For clarity, details of the cooling facilities such as the *nitrogen cycle* are omitted in Figure 2.2.)

3. The separation of the various components of the feed air takes place in the rectification section:

   (a) In the *high pressure column* (*HPC*, pressure $\approx$ 6 [bar]), the feed is separated into pure liquid *high pressure nitrogen* at the top of the column and impure liquid *high pressure oxygen* at the sump of the column.

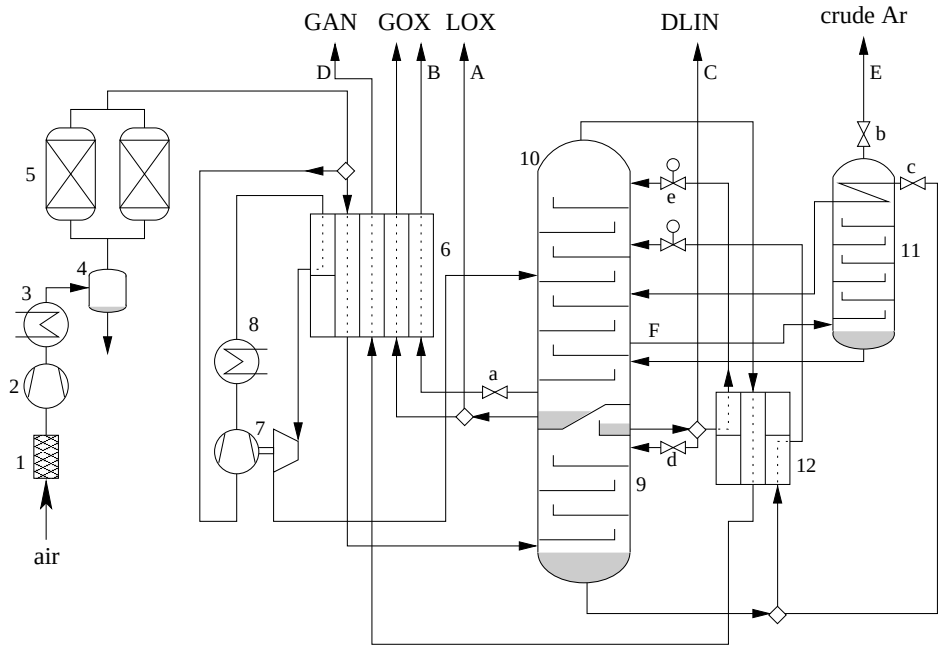

Figure 2.2: A simplified flowsheet of a low pressure cryogenic air separation plant without refrigeration cycle according to [BBKS 83], [ELBK 97], [Zapp 94].

main units: 1: filter, 2: compressor, 3: cooling, 4: flash, 5: molecular sieves, 6: main heat exchanger, 7: compressor/expander, 8: cooling, 9: high pressure column, 10: low pressure column, 11: argon column, 12: heat exchanger

products: GAN: gaseous N_2, DLIN: high pressure liquid N_2, GOX: gaseous O_2, LOX: liquid O_2, crude Ar

controls: a: GOX drain, b: crude Ar drain, c: Ar condenser turnover, d: reflux HPC, e: reflux LPC

constraints (an example set of constraints is specified in Table 2.2): A: O_2 LOX, B: O_2 GOX, C: O_2 DLIN, E: Ar Prod, F: O_2 feed ArC

(b) The high pressure oxygen is fed into the *low pressure column* (*LPC*, pressure ≈ 1.5 [bar]) at the middle section; high pressure nitrogen is fed into the low pressure column at the top section. At the top of the low pressure column highly pure nitrogen is obtained. At its bottom oxygen with about 0.3 [mol%] impurities is removed in liquid and gaseous fractions.

(c) At a certain position in the middle section of the low pressure column (the *argon side-stream level*) a gaseous fraction is drained off which contains argon in a higher concentration. This fraction is fed into the

argon column. At the top of the argon column, *crude argon* is removed. The sump product of the argon column is fed back into the low pressure column.

It is of *crucial* importance that the feed of the argon column contains nitrogen[(i)] only in a very low concentration as otherwise the rectification process in this column can break down. Due to the strong interaction in the plant this would cause harm to the entire air separation process. Once nitrogen has entered the argon column exceeding a critical amount there is no quick countermeasure due to large time constants present in the air separation plant. E.g., according to [MoLe 91] a typical high purity distillation column has an open-loop time constant of approximately 200 [min], and a generally achievable closed-loop time constant of approximately 20 [min]. This means that when it becomes obvious that the process dynamics tend to evolve into an undesired state it is already too late for counteractions. Even worse, a load-change strategy that fulfils all constraints within, say, the initial two hours, can lead to a break down one hour later.

2.2 Modelling and Simulation of Chemical Engineering Processes

We accept as an axiom
that a meaningful system
produces meaningful mathematics.

Bellman [Bellm 71], Sect. 14.1

Numerical simulation and optimisation tools have become indispensable in the development and design of chemical engineering processes [Marq 91], [Sawy 92], [Voit 94], [BAFG 98], [Marq 99a]:

On the one hand, production has to be steadily improved, i.e., feedstock and energy consumption has to be as efficient as possible – which also includes the trend to more flexible plant operation – while product quality still has to be guaranteed. On the other hand, growing restrictions on pollution have to be satisfied. Additionally, safety and risk analysis is necessary in order to allow safe operation of the highly optimised processes. Finally, the development of the processes itself has to be fast and efficient while meeting all of the demands above.

[(i)] In normal operation the measurement of the very small nitrogen fraction at the argon transition is not feasible. As a remedy, the oxygen fraction is measured. This indirect technique is justified by physical reasons as an appropriate lower bound on the oxygen fraction excludes a *breakthrough* of nitrogen at the argon transition (cf. Table 2.2: O_2 *fraction in feed of argon column*).

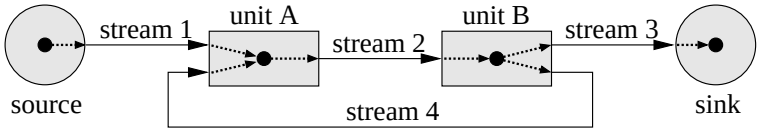

Figure 2.3: The flowsheet, a graph oriented description of a process.

2.2.1 Flowsheeting

Due to the the complexity of the plants and due to the high accuracy requirements in industrial application (cf., e.g., Table 2.2) the process models are typically of large scale (consider, e.g., the example given in Section 6.2.3). Therefore, such models have to be generated by means of computer-aided modelling tools.

In general, the mathematical model of a chemical engineering plant is based on the *flowsheet* of the process. A flowsheet is a graph-oriented description which consists in the parts of a plant (the *units*) in its nodes while the edges between the nodes represent physical flows or flows of information (the *streams*), see Figure 2.3. This *flowsheeting technique* is a common tool in chemical engineering [ELBK 97]. A simple flowsheet has already been shown in Figure 2.2.

The graph-oriented description allows automated compilation of the entire mathematical model of the plant using standard or legacy models of the single units, cf., e.g., [HRW 99]. In turn, the standard models are provided in unit libraries.

Example 3:
As an example for a standard unit consider the single stage thermal separator (*flash*), see Figure 2.4. In the flash drum the feed is separated into a vapour fraction which is richer in the lower boiling components, and into a liquid fraction which is richer in the higher boiling components. The vapour ascends and is removed from the top of the drum while the liquid is withdrawn from the bottom. An index-2 DAE model of the single-stage equilibrium separation unit with neglectable holdup in the vapour phase is given by [ELBK 97] [(ii)]

$$\frac{\mathrm{d}}{\mathrm{d}t}M = F - V - L\,, \tag{2.1a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(Mx_\mu) = Fz_\mu - Vy_\mu - Lx_\mu\,; \quad \mu = 1, \ldots, n_{\mathrm{comp}} - 1\,, \tag{2.1b}$$

$$\sum_{\mu=1}^{n_{\mathrm{comp}}} x_\mu = 1\,, \tag{2.1c}$$

[(ii)] In order to keep consistency with the chemical engineering literature we *temporarily* overload symbols which have already been used in a different context.

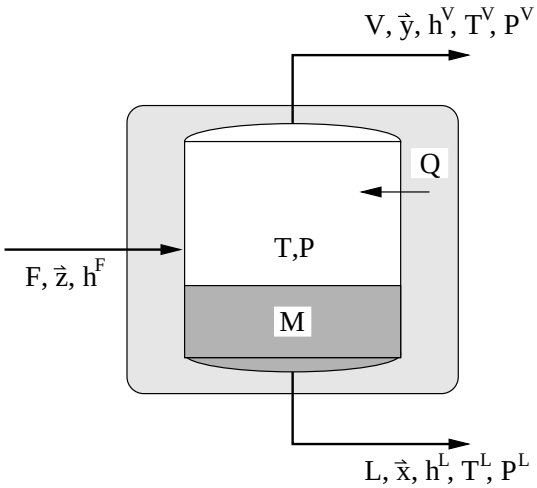

Figure 2.4: Flash drum.

$$\sum_{\mu=1}^{n_{\text{comp}}} y_\mu = 1\,, \tag{2.1d}$$

$$y_\mu = K_\mu(T,P,x,y)x_\mu\,; \quad \mu = 1,\ldots,n_{\text{comp}}\,, \tag{2.1e}$$

$$L = \Phi(M)\,, \tag{2.1f}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(Mh^L) = Fh^F - Vh^V - Lh^L + Q\,, \tag{2.1g}$$

$$h^L = h^L(T^L,P^L,x)\,, \quad h^V = h^V(T^V,P^V,y)\,, \tag{2.1h}$$

$$P^V = P^L = P\,, \quad T^V = T^L = T\,, \tag{2.1i}$$

where M is the molar liquid holdup in drum, V and L are the molar vapour and liquid flowrates, x_μ and y_μ are the mole fractions of component μ, $\mu = 1,\ldots,n_{\text{comp}}$, in liquid and vapour, h^L and h^V are the molar enthalpies of liquid and vapour, T, T^L, and T^V are the temperatures in drum, liquid, and vapour, and P^L and P^V are the pressures in liquid and vapour. The pressure P in the drum, the heat source Q, the flowrate F of feed, the mole fraction z_μ of component μ, $\mu = 1,\ldots,n_{\text{comp}}$, in the feed, and the molar enthalpy of the feed h^F are chosen as control variables. For further details on the model see, e.g., [ELBK 97].

Once Eqs. (2.1a)–(2.1i) have been inserted into a library this model can (and as one of the fundamental units in chemical engineering will) be used several times within a flowsheet without the need to care for the underlying equations. ◊

Example 4:
Another example is the model of a simple one dimensional heat exchanger unit Eqs. (3.58)–(3.63) discussed in Example 6 on page 131). ◊

2.2.2 The Numerical Tools

As reviewed in [Marq 99a] the beginnings of numerical process simulation date back to the early 1950's. Since then numerical process simulation has experienced a rapid progress. Nowaday several sophisticated software packages for the numerical treatment of chemical engineering processes are available, e.g., ABACUSS ([FeBa 96] [PBa 96], [Allg 97], [Feeh 98]), ABACUSS II ([Bart 00], [TCB 01]), ASPEN CUSTOM MODELLER™ (which is based on SPEEDUP® [Bart 00]), DIVA ([HMG 88], [KHMG 90], [Krön 02]), gPROMS® ([Bart 92], [PaBa 93], [OhPa 96]), HYSYS™, OPTISIM® ([Burr 91], [Burr 93], [BMS 94], [Voit 94], [Zapp 94], [ELBK 97], [EnSc 97], [Sörg 97]), and SPEEDUP® ([Pant 88b]).

[Marq 91] gives a detailed overview on the issues connected to dynamic process simulation and a summary of the simulation tools available in the early 1990's. Some of the tools are compared in [KFGE 97] (ABACUSS, DIVA, and gPROMS®) and in [Luyb 01] (ASPEN DYNAMICS™, HYSYS™).

The simulation tools can be classified according to the methodology they employ for solving the model. [Bart 00] discusses three basic methods:

- In the *simultaneous equation-oriented* method [PaBa 93] the entire flowsheet is translated into a single system of equations which is then solved simultaneously. In application these systems are typically very large.
- In contrast, the *sequential-modular* method [HiHe 86] proceeds step-wise. Each unit is simulated on its own, using the input generated from the simulation of the upstream units. After convergence the result is passed on to the downstream units.
- The *simultaneous-modular* [Bart 00] method blends the two methods above. Here the overall system of equations is partitioned into blocks of equations which are solved in sequence. The partitioning is done independently from the division of the process into units.

A remarkable result of [Bart 00] is that in application the sequential option is still indispensable although the simultaneous method is theoretically and numerically more appealing. On the one hand, the simultaneous equation-oriented approach is better suited for more sophisticated tasks such as e.g., dynamic simulation, dynamic sensitivity analysis, and optimal control than the sequential-modular method. On the other hand, in practice the sequential-modular technique is the more robust and reliable tool for generating an initial steady-state solution. The problem of simultaneous equation-oriented approach is that multi-dimensional Newton-type root-finding methods (which in general are at the core of these simulation algorithms) require an initial guess sufficiently close to the desired solution. In contrast, the sequential-modular method can use unit specific knowledge in order to find a solution.

Basis of our practical work is the software tool OPTISIM® [Burr 93], [ELBK 97]. OPTISIM® is an in-house modelling, simulation, and optimisation tool of the

Linde AG. Its steady-state calculation capabilities include design optimisation, data reconciliation and parameter fitting, and optimisation of plant operation (on-line and off-line). Its dynamic calculation features cover the development of control strategies, the tuning of control parameters (trial and error), and operator training. Recently, OPTISIM® has been extended with an optimal control algorithm [EKKS 99]. This optimal control algorithm serves as our starting point.

Within OPTISIM® a unit model in general consists of a set of differential and/or algebraic equations together with their first order partial derivatives coded in executable form. Additionally, switching functions are provided if necessary.

In standard operation OPTISIM® employs the simultaneous equation-oriented simulation technique; the computation of a steady-state solution can also be performed in the sequential-modular mode. In order to compile the model of an entire process in the simultaneous equation-oriented mode OPTISIM® analyses the flowsheet of the plant, assigns the vector of state variables, and generates a calling sequence for the unit models. The evaluation of the numerical model of the flowsheet or of its partial derivatives consists then in the combination of the evaluations of the unit model equations or of the partial derivatives connected to each unit in the flowsheet, respectively.

2.2.3 Special Properties of the Models Considered

In general, dynamical process models of chemical engineering plants are described by DAEs in linear implicit form (cf. Definition 1.3). OPTISIM® uses semi-explicit DAE models[(iii)]

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t), \operatorname{sign} \boldsymbol{q}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t)) , \tag{2.2a}$$

$$0 = \boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t), \operatorname{sign} \boldsymbol{q}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t)) , \tag{2.2b}$$

$\boldsymbol{f} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}+m_{\boldsymbol{q}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$, $\boldsymbol{g} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}+m_{\boldsymbol{q}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$, where discontinuities are handled by switching functions $\boldsymbol{q} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}} \to \mathbb{R}^{m_{\boldsymbol{q}}}$. For ease of notation we include model parameters in the vector of external controls $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^{n_{\boldsymbol{u}}}$ as a special case.

In the applications of interest the models show several properties that become important factors in the design and practical applicability of both off-line and on-line optimal control algorithms.

2.2.3.a Size of the Models Considered

As already noted, dynamic models of the processes of interest are of very large scale (in our case, $n_{\boldsymbol{x}} + n_{\boldsymbol{y}} \approx \mathcal{O}(1000) \ldots \mathcal{O}(10000)$). Currently the numerical simulation of such models is still only possible by exploiting the sparsity of the

[(iii)] $\operatorname{sign}([\boldsymbol{q}]) := [\operatorname{sign}(\boldsymbol{q}_1), \ldots, \operatorname{sign}(\boldsymbol{q}_{m_{\boldsymbol{q}}})]^T \in \{-1, 0, 1\}^{m_{\boldsymbol{q}}}$

Jacobian matrices (about 0.01 to 0.1 percent nonzero entries), e.g., [ELBK 97]. Although the Jacobians neither possess any special structure nor in general exhibit definiteness, their sparsity allows the application of special solvers designed for very large and sparse linear systems of equations ([Duff 99] discusses various methods).

2.2.3.b Differential Index 2

The large-scale models of real-world air separation plants treated by the simulation tool OPTISIM® exhibit an index of two, cf., e.g., Section 6.3.5 and Section 6.3.6. The index of the DAEs has been verified in a structural sense by Pantelides' Algorithm (see Section 3.1.5). This result has been obtained by the implementation SPALG [UKM 95] of Pantelides' Algorithm which was added to OPTISIM® in the course of our research. Causes for the higher index of these DAEs can be found at both, unit modelling level and flowsheet level.

As investigated by [GaCa 92] in some cases the higher index of single unit models can be directly attributed to modelling issues. However, they also note that under certain modelling conditions higher index problems cannot always be avoided. A closer inspection of OPTISIM® shows that in part the higher index of some of its process models is due to the modelling paradigms applied. Thus this higher index has to be classified as a problem which is not easily avoided. In the context of chemical engineering, the modelling of dynamic equilibrium processes is a typical situation where higher index DAEs arise as there differential variables are related by the phase equilibrium conditions [PGMS 88]. As an example for the consequences of the *specification* of the simulation problem itself [PGMS 88] examine the rigorous model of a rectification column. In the example problem either the column top pressure or the column bottom pressure has to be specified externally. If the column top pressure is fixed then the model is shown to have a differential index of two. If the bottom pressure is fixed then the index column equals the number of column trays plus one (note that high purity rectification columns typically own $50 - 150$ (theoretical) trays). If, however, the perfect control law for the column top pressure imposed by the fixed pressure condition is relaxed by a suitable real control law, or if it is replaced by a pressure drop relation then the index of the model is one.

At flowsheet level the *connection* of several index-1 unit models can produce an index of the overall system exceeding one [LeSr 93a], [LeSr 93b]. For the special case of DAEs arising from the modelling of chemical engineering processes [Moe 95] and [Krön 02] observe that these DAEs are in general of index 1 if all unit models are of index 1 and if the units are coupled by algebraic states only.

[Matt 89] and [FeBa 96] state that problems with higher index arise naturally. Therefore, higher index problems are an active field of research in chemical engineering, e.g., [GPS 95], [Feeh 98], [EKKS 99]. Higher index problems are also present in other fields of application, especially in models of electrical circuits (e.g., [GüFe 99a]) or in mechanical multibody systems (e.g., [ABES 93], [FüLe 91]).

Finally, there is a controversy whether the treatment of higher index problems is reasonable at all [Marq 91]. On the one hand, higher index problems are termed as ill-conditioned, leading to the recommendation that they should be avoided. On the other hand, they are considered as reasonable problems which can be handled by tailored numerical methods.

Given a higher index DAE model index reduction may be performed in order to obtain an index-1 model. In this context the *dummy derivative method* (cf. [MaSö 92], [MaSö 93], [CeEl 93], and Section 3.1.6) has already been successfully applied, e.g., in ABACUSS [Feeh 98], and DYMOLA™ [CeEl 93], [ECO 93]. In both algorithms automated formula manipulation is employed. In the case of OPTISIM® automated formula manipulation is ruled out as the unit models are available as executable code only. Thus the advantages of the index reduced model have to be seen in relation to the numerical costs imposed by the necessarily numerical differentiation of parts of the Jacobian of the DAE (cf. Section 3.2.3.b). Here automatic differentiation (e.g., ADIFOR [BCCGH 92]) may be an interesting approach to be examined in future investigations. In the context of the presented work automatic differentiation has been out of scope due to lack of resources for implementation. Moreover, during the course of an integration the initially generated index-1 model can become inappropriate, and a new index-reduced model has to be determined (*dummy derivative pivoting*, [MaSö 93]). The monitoring of the index-reduced system requires numerical rank estimation [MaSö 93] which is costly especially for large-scale DAE systems as in our setting. In order to address this problem less expensive monitoring strategies have been investigated, cf., e.g, [Feeh 98].

Remark 2.1:
The largest numerical examples considered by [Feeh 98] are the index-2 model of a distillation column with 195 equations, and the startup of a coupled CSTR (*continuously stirred tank reactor*) / distillation column section modelled by an index-2 DAE with 680 equations. ◊

In summary, reformulation of the unit models and index reduction by the dummy derivative method may be considered as possible approaches in order to obtain index-1 models instead of index-2 models in the industrial simulation environment OPTISIM®. However, on the one hand the realisation of each of these approaches requires considerable effort and resources. On the other hand, algorithms suitable for the numerical integration and parametric sensitivity analysis of large-scale index-2 DAEs are already employed in OPTISIM® [ELBK 97], [EKKS 99]. Therefore the decision has been made to stick to the DAE models, and to extend the numerical algorithms where necessary (cf. Chapter 4).

2.2.3.c Discontinuities

Typically, the process models considered exhibit discontinuities. As already discussed in Section 1.3 the discontinuities are commonly classified into purely *time dependent* discontinuities (where the time of the discontinuity is fixed, e.g., in

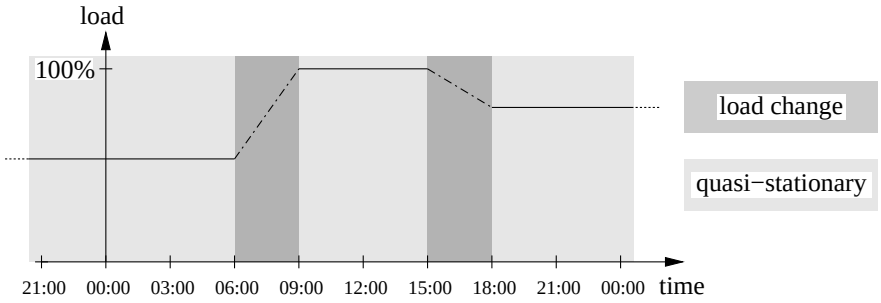

Figure 2.5: A possible sequence of load-changes and phases of quasi-stationary operation at constant load (process air input).

ramp forcing functions) and *state dependent* discontinuities which in general occur at a priori unknown times. State dependent discontinuities arise, e.g., due to the physical nature of the process (such as in batch processes), by the necessity for different models in different domains in state space (e.g., the model of an open valve differs from that of a closed valve), or by necessary model simplifications (say, continuously interpolated physical property measurements). State dependent discontinuities are generally more difficult to handle than purely time dependent discontinuities as their location requires considerable effort (cf. Section 1.3.2).

2.3 A Concept for Model Predictive Optimal Control

Omnia tempus habent
et suis spatiis transeunt universa sub caelo.
(For everything there is a season,
and a time for every matter under heaven.)

The Bible, Qoh 3:1

2.3.1 Problem Setting

In common, cryogenic air separation plants are operated quasi-stationarily at a constant load over many periods of time, i.e., the amount of processed air is kept constant. From time to time a transition of the process to a different load has to be performed (see Figure 2.5). Such a transition is termed a *load-change*. More flexible plant operation is required. Thus load-changes will occur more often than usual in the past. For the problems treated a load-change takes about $1-3$ hours. In our framework (see Figure 2.8 on page 49) the operator has to specify the type of the load-change, and the optimality criterion to be minimised as well as the

constraints to be satisfied during the load-change. As discussed in Section 2.1.1 and Section 2.1.2 the strict fulfilment of the constraints is of highest priority during the load-change.

We now consider the optimal control of an air separation plant in permanent operation, e.g., during a year ($[t_0, t_f] = [0, 365]$ days). The load history (see Figure 2.5) admits application of the Bernoulli-Bellman Principle of Optimality so that we may split the long term optimal control problem into optimal control problems on smaller horizons without loss of optimality. Each of these horizons covers *quasi-stationary* operation at load A — *instationary* load-change — and *quasi-stationary* operation at load B.

Given the specifications of the customer the quasi-stationary points of operation can be computed efficiently, cf., e.g., [ELBK 97], [Sörg 97]. Therefore we assume that the values of the state variables $\boldsymbol{x}^*_{\text{load}A/B}$ and $\boldsymbol{y}^*_{\text{load}A/B}$, and of the control variables $\boldsymbol{u}^*_{\text{load}A/B}$ characterising the different loads at the beginning and at the end of the load-change are known.

In contrast to the quasi-stationary case the determination of load-change strategies was mainly based on human expertise and experimental data in the past. This data needs to be obtained from time and cost expensive tests conducted on the ready-to-use air separation plant. As a consequence it has been proposed to employ advanced numerical methods in order to facilitate the determination of the load-change strategies. [Nijs 96] proposed a direct transcription method suitable for index-1 DAE models in the context of air separation plants. A problem, however, was the approximation of the gradients required for the sequential quadratic programming (SQP) solver by finite differences of perturbed trajectories via external numerical differentiation. Such an approximation is numerically costly, of limited accuracy, and to be handled with great care [Kieh 99]. In [EKKS 99] a method for the numerical off-line determination of load-change strategies for air separation plants modelled by large-scale index-2 DAEs has been developed. This method is based on direct transcription of the optimal control problem into an optimisation problem employing the direct single shooting approach. Reliable gradient information is obtained via computation of the sensitivities by efficient integration of the sensitivity DAE associated to the model DAE (see Section 2.4 and [BBBB 01] for details). However, only time-dependent discontinuities in the models have been considered there. The extension of the algorithm for the numerical approximation of sensitivity functions to DAE models with state dependent discontinuities will be discussed in Section 4.3.

Applying the Bernoulli-Bellman Principle of Optimality we now split the long term optimal control problem into a sequence of load change problems, each of which can be solved separately. If we consider, e.g, the j^{th} load-change on a prediction horizon $P_j=[t_{0,j}, t_{f,j}]$, $j = 1, 2, 3, \ldots$, covering the entire load-change the corresponding optimal control problem incorporating the DAE model Eqs. (2.2a)–(2.2b) can be formulated as:

Definition 2.1 (Optimal Control Problem)
Let there be a dynamical process described by a model Eqs. (2.2a)–(2.2b). *Given an interval in time* $[t_{0,j}, t_{f,j}] \subset \mathbb{R}$, *an objective function* $\mathcal{J} : \mathcal{C}_p^0 \to \mathbb{R}$ *in Mayer form (represented by* $E : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}$*), path inequality constraints* $\boldsymbol{c} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}} \to \mathbb{R}^{m_{\boldsymbol{c}}}$, *point inequality constraints* $\boldsymbol{k}_\mu^{ic} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}} \to \mathbb{R}^{m_{\boldsymbol{k}_\mu^{ic}}}$, $\mu = 1, \ldots, m_{\boldsymbol{k}^{ic}}$, *and initial conditions* $\boldsymbol{k}^{ini} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}+n_{\boldsymbol{u}}} \to \mathbb{R}^{m_{\boldsymbol{k}^{ini}}}$ *the* optimal control problem *is to find optimal controls* $\boldsymbol{u} \in \mathcal{C}_p^0(\mathbb{R}, \mathbb{R}^{n_{\boldsymbol{u}}})$ *such that* [(iv)]

$$\mathcal{J}[\boldsymbol{u}] = E(t_{f,j}, \boldsymbol{x}(t_{f,j}), \boldsymbol{y}(t_{f,j})) \quad \longrightarrow \quad \min_{\boldsymbol{u}}! \tag{2.3}$$

subject to [(v)]

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t), \operatorname{sign} \boldsymbol{q}) \,;\; t \in [t_{0,j}, t_{f,j}] \,, \\
0 &= \boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t), \operatorname{sign} \boldsymbol{q}) \,, \\
0 &\geq \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t)) \,, \\
0 &\geq \boldsymbol{k}_\mu^{ic}(t_\mu, \boldsymbol{x}(t_\mu), \boldsymbol{y}(t_\mu), \boldsymbol{u}(t_\mu)) \,;\; \mu = 1, \ldots, m_{\boldsymbol{k}^{ic}} \,, \\
0 &= \boldsymbol{k}^{ini}(t_{0,j}, \boldsymbol{x}(t_{0,j}), \boldsymbol{y}(t_{0,j}), \dot{\boldsymbol{x}}(t_{0,j}), \boldsymbol{u}(t_{0,j})) \,.
\end{aligned}$$

In our applications each horizon P_j typically covers approximately one up to several hours of real-time. In general, the objective consists in a weighted sum of partial objectives

$$\mathcal{J}[\boldsymbol{u}] = \omega_1 \mathcal{J}_1[\boldsymbol{u}] + \cdots + \omega_{n_\mathcal{J}} \mathcal{J}_{n_\mathcal{J}}[\boldsymbol{u}], \quad \omega_\nu \geq 0, \nu = 1, \ldots, n_\mathcal{J} \,,$$

corresponding to, e.g., product gain and energy consumption.

Remark 2.2:
$\mathcal{J}[\boldsymbol{u}]$ may also include penalty terms in order to account for constraints that are hard to fulfil, e.g., [(vi)]

$$\mathcal{J}_{\boldsymbol{c}_\mu}[\boldsymbol{u}] = E_{\boldsymbol{c}_\mu}(t_{f,j}) = \int_{t_{0,j}}^{t_{f,j}} (\boldsymbol{c}_{\mu,+}(t, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{q}))^2 \, \mathrm{d}t \,,$$

cf., e.g., [VSP 94b]. Here, $\boldsymbol{c}_\mu : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}} \to \mathbb{R}$ represents a real-valued component of the multi-dimensional function $\boldsymbol{c}$, i.e., $\mu \in \{1, \ldots, m_{\boldsymbol{c}}\}$ fixed but otherwise arbitrary.

The penalty term $\mathcal{J}_{\boldsymbol{c}_\mu}[\boldsymbol{u}]$ vanishes at the *exact* solution of the infinite dimensional optimal control problem Eq. (2.3). It becomes significant after discretisation of the constraints $\boldsymbol{c}$ on a grid on the horizon $[t_{0,j}, t_{f,j}]$ which is one method to treat path (inequality)

[(iv)] Without restriction of generality we consider optimal control problems with the objective functional in Mayer form. Objective functionals in Lagrange form or Bolza form are easily transformed into Mayer form. Additionally, the weighted least squares objective can be found in parameter identification problems, cf. Remark 2.3 below.

[(v)] For ease of notation we drop the arguments of the switching function, i.e.,

$$\boldsymbol{q} := \boldsymbol{q}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t)) \,.$$

[(vi)] The positive part of a real-valued function $\eta(\cdot)$ is defined as

$$\eta(\cdot)_+ := \max(\eta(\cdot), 0) \,.$$

constraints numerically in the context of direct optimal control algorithms (cf. Section 2.4.2). ◊

Remark 2.3:
The optimal control problem specified in Definition 2.1 is not restricted to the load-change problem. In case of a parameter estimation problem $\mathcal{J}$ describes a nonlinear least squares objective consisting of a sum of squares of differences between values obtained from measurement and simulation. E.g., if at the sample times t_ν^{mmt}, $\nu = 1, \ldots, n_{\mathrm{mmt}}$, measurements $\boldsymbol{z}_{(\nu)}^{\mathrm{mmt}}$ of (typically) a subset of the state variables $\boldsymbol{z} \subset \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_{\boldsymbol{x}}}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_{\boldsymbol{y}}}\}$, $\boldsymbol{z} \in \mathbb{R}^{n_{\boldsymbol{z}}}$, are available, then the weighted least squares objective with weights $\boldsymbol{w}_\nu \in \mathbb{R}_+ \backslash \{0\}$, $\nu = 1, \ldots, n_{\boldsymbol{z}}$, can be denoted as

$$\mathcal{J}[\boldsymbol{u}] = \sum_{\substack{\nu=1,\ldots,n_{\mathrm{mmt}},\\ t_\nu^{\mathrm{mmt}} \in [t_{0,j}, t_{\mathrm{f},j}]}} \sum_{\mu=1}^{n_{\boldsymbol{z}}} \frac{1}{\boldsymbol{w}_\mu} \left(\boldsymbol{z}_\mu(t_\nu^{\mathrm{mmt}}) - \boldsymbol{z}_{\mu,(\nu)}^{\mathrm{mmt}} \right)^2 .$$

This type of objective applies to item **(c)** in Figure 2.8 on page 49 and is used in some of the examples given in Section 6.2.3. ◊

2.3.2 Model Predictive Control

Prediction is very difficult, especially of the future.

Niels Bohr

In the last decades the *(nonlinear) model predictive control* ((N)MPC) strategy has been developed which can serve as a basic framework for an online solution of the optimal control problem Eq. (2.3).

Originally, MPC algorithms have been investigated since standard controllers – typically, PID controllers (*proportional plus integral and derivative* controllers, cf., e.g., [Dubb 81], [Fish 91]) – are no longer able to fulfil increasing requirements on controller performance [BiRa 91]. Especially, tightly coupled processes, plants operated in largely varying regions of operation, and process constraints are difficult to handle with standard PID controllers. Here, the prediction of the process behaviour makes MPC suitable for complex processes and changing operational regimes – as far as covered by the process model. At the same time MPC provides a means for explicit incorporation of both hard and soft constraints as a part of the controller formulation.

Still most process control problems can be successfully addressed by conventional control schemes, cf. [Sebo 99]. Therefore [Sebo 99] recommends to apply advanced control schemes such as (N)MPC only if they offer significant advantages over conventional control techniques. In this context [LoPe 99a], [LoPe 99b] analyse the dependency of the economic benefit of an MPC controller from the structural design of the control loop.

2.3.2.a General Work

[EaRa 92] define model predictive control as

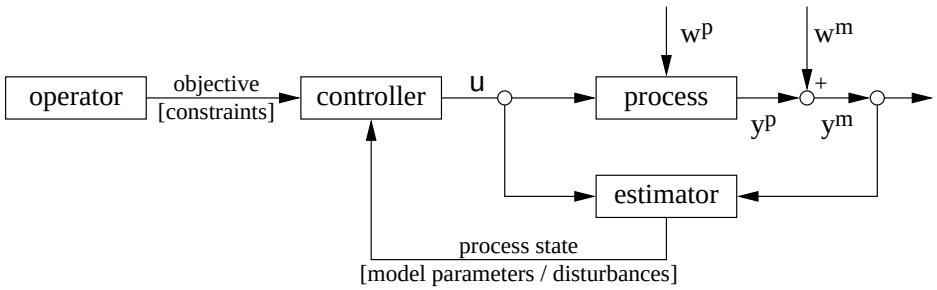

Figure 2.6: The basic structure of an MPC control loop, adapted from [PRE 90], [Rawl 99]. Items in square brackets [...] are optional or advanced features.
$\boldsymbol{u}$: controls, y^{p}: plant outputs, y^{m}: measured outputs,
w^{p}: external process disturbances, w^{m}: measurement errors

Definition 2.2 (Model Predictive Control)
Model predictive control (MPC) *is a control scheme in which the controller determines a manipulated variable profile that optimises some open-loop performance objective on a time interval extending from the current time to the current time plus a prediction horizon. This manipulated variable profile is implemented until a plant measurement becomes available. Feedback is incorporated by using the measurement to update the optimisation problem for the next step.*

[EaRa 92] emphasise that the characteristic feature of model predictive control is the repeated solution of an open-loop optimal control problem over a time horizon starting at the present and extending into the future. The incorporation of process knowledge in form of a model is not a proprietary feature of MPC since classic feedback controllers may also include process models. The general control structure is sketched in Figure 2.6.

Definition 2.2 already contains the basic strategy for an MPC algorithm. The corresponding algorithm consists in the repeated execution of the following steps during the progress of the process:

Algorithm 2 (Basic MPC Algorithm)

1. *Initialise system. Fix an initial control strategy.*
2. *Implement the open loop control on the actual control horizon. In parallel, execute the subsequent calculations.*
3. *Estimate the state of the process.*
4. *Compute an open loop optimal control on the actual optimisation horizon using the estimated state as initial conditions.*
5. *Advance optimisation and control horizon in time.*

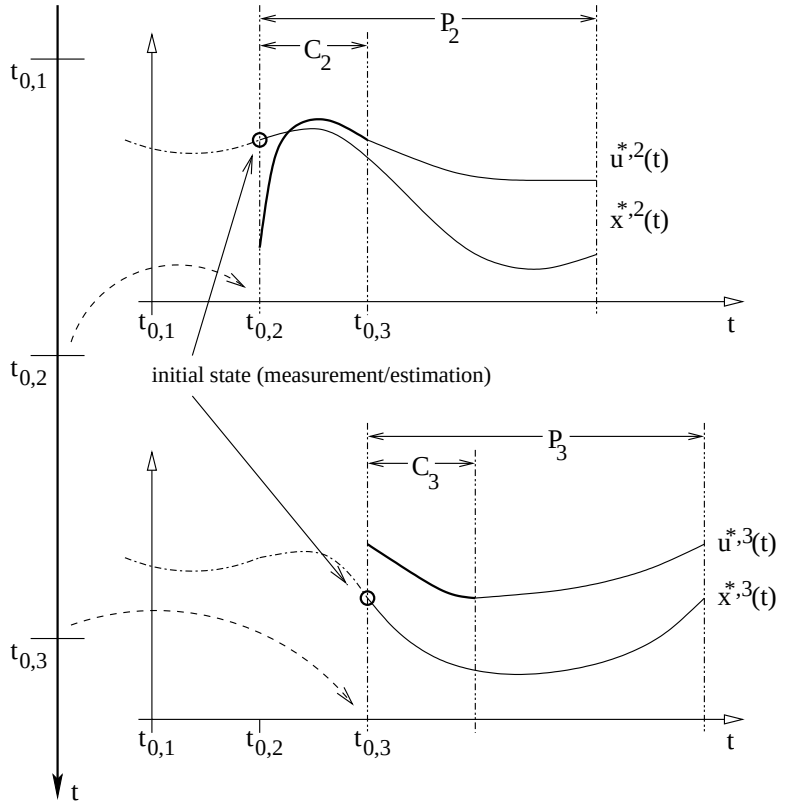

Figure 2.7: The progress of an ideal moving horizon model predictive controller in time (without delay due to measurement, data transfer, estimation, and feedback computation). A sketch considering these dead-times can be found in [BBBB 01].
$P_{2;3}$: prediction horizon, $C_{2;3}$: control horizon, $\boldsymbol{x}^{*,2;3}$: predicted optimal internal process states on $P_{2;3}$, $\boldsymbol{u}^{*,2;3}$: open-loop optimal control on the respective prediction horizon.

6. *Continue with step* ***2****.*

The *optimisation horizon* is the interval in time starting with the present on which the open-loop optimal control problem is solved, i.e., the portion of the future for which a prediction of the process behaviour is made and then taken into account for control design. The *control horizon* is a subinterval of the prediction horizon (in general, it is a proper subinterval of the prediction horizon). Only the restriction of the open-loop optimal control to the control horizon is implemented. A schematic overview of the progress of the algorithm in time is given in Figure 2.7.

The length of both the prediction horizon and the control horizon are two tuning parameters in the *moving horizon technique.* An enlargement of these

horizons increases the robustness[vii] against modelling errors and disturbances. However, also the reaction time deteriorates and the computational time required for the larger optimal control problems grows [PRE 90].

Model predictive control methods are classified into LMPC and NMPC methods. In case of a linear process model, a quadratic objective function, and constraints that are linear in the controls the respective algorithm is termed a *linear model predictive control* method (LMPC). Otherwise (especially if the model is nonlinear) the control algorithm is called a *nonlinear model predictive control* method (NMPC) [BiRa 91].

Although constraints are easily incorporated into the MPC formulation they can lead to problems regarding stability already for linear moving horizon controllers with finite prediction horizon. If an infinite prediction horizon is applicable it removes instability, however at the cost of infeasibility[viii] [RaMu 93]. [AlBo 95] have shown the existence of a finite prediction horizon for NMPC with a discrete time model and explicit state and control constraints such that the controller is asymptotically stable, given that some technical assumptions hold. Yet the computation of the required length of the horizon is impractical. According to [ScRa 97] a final solution for the stability problem for constrained MPC in the general case has not yet been found.

Remark 2.4:
In the context of nonlinear control systems already the definition of stability is nontrivial. Especially, there is no single definition of stability [AlDo 97]. For further information we refer to the discussion of stability in [AlDo 97]. ◊

[HAM 00] consider the decomposition of the control problem according to different time scales inherent in the process. If the disturbances can be split up into a constant or slowly varying part and into high frequency noise also the control algorithm naturally decomposes into tasks that have to be performed infrequently, such as the estimation of slowly varying parameters, and into tasks that have to be solved in higher frequency, such as the computation of new optimal controls. E.g., [LiBi 90] employ this decomposition in their Newton-type control method.

Until now, the ongoing research has produced a variety of predictive control approaches. The interest in this field is mirrored in a huge number of publications so that the following discussion can only give some aspects of the work that has been done (however, [Mayn 00] asserts convergence of the NMPC techniques proposed). For further information and in some cases extensive bibliographic reference

[vii] "A property is *robust* if, should the property hold at one point, it holds in an open neighbourhood of that point. [...] Robustness [in the context of automatic control] refers to the notion that small changes in initial conditions and/or structural properties of the system elements preserve solution properties [...] and yield only small variation in closed loop performance [...]." ([BGW 90], p. 5).

[viii] According to [ScRa 97] *infeasibility* means that no sequence of control inputs within the input limits exists which satisfies the state constraints (or, in our more general setting, the path constraints).

we refer to general reviews, e.g., [GPM 89], [Bequ 91a], [Fish 91] (who also provides reference to standard control techniques), [Mayn 97], [MeRa 97], [Hens 98], [ABQRW 99], [MoLe 99], [Rawl 99], [Sebo 99], [MRRS 00], [BBBB 01].

Furthermore, [Benn 96] gives an interesting overview on the historical development of (conventional) automatic control, and [AlZh 00] survey many recent results.

2.3.2.b Linear Model Predictive Control

[MuRa 93] discuss LMPC in a theoretical framework given by a stabilising infinite horizon linear quadratic regulator. Other overviews on LMPC can be found, e.g., in [GPM 89], [LeCo 97]. E.g., *DMC* (dynamic matrix control) and *QDMC* (quadratic dynamic matrix control), [EaRa 92]), *MAC* (model algorithmic control), and *IMC* (internal model control, [GaMo 82]) are LMPC methods.

The basic MPC algorithm does not prescribe a certain kind of representation for the process, although the particular choice of the model affects the final algorithm, e.g., in the range of processes that can be treated (such as integrating systems), in the incorporation of disturbances, and in the applicability to more complex systems. Model types traditionally used for LMPC are step response models, finite impulse response models, transfer function models, and models in state space form [MoLe 91], [Rick 91]. Other models are considered, e.g., in the review of [BBBB 01]. An issue besides the modelling of the known process is the modelling of *a priori* unknown disturbances. In the context of LMPC [MoLe 91] show that inappropriately modelled disturbances can lead to poor closed-loop performance independent from the tuning of the controller.

2.3.2.c Nonlinear Model Predictive Control

Chemical engineering processes are known to be inherently nonlinear. Thus the quality of the prediction based on a linear model (as in LMPC) is limited in case of processes operated in larger operational regions or processes subject to significant disturbances. In order to obtain a better prediction of the system behaviour a straightforward step is to employ a nonlinear process model [ABQRW 99], aiming at an improvement of the control performance.

In this sense NMPC can be regarded as an evident extension of LMPC. However, the nonlinearity makes its theoretical and numerical investigation cumbersome and renders its application far from trivial. Questions arise for, e.g., stability and robustness of the method, efficiency of the calculations, suitable feedback design incorporating disturbances, and proper handling of constraints [BiRa 91].

The solution of the nonlinear programming problems (NLPs) resulting from the MPC formulation after discretisation of the optimal control problems defined on the moving prediction horizon poses two major problems. Either the convergence of the NLP solver may be too slow, or the solver may not converge to a suitable solution at all. In order to address the first problem [BiRa 91] recommend

to utilise the warm start capabilites of modern NLP solvers, to abort the NLP solution process prematurely and continue with this result, or to solve a relaxed control problem (the latter will be discussed in Section 2.6.3). The second problem can be alleviated by cascading the optimisation-based controller with lower level controllers, e.g., with PID controllers. If cascaded, the optimisation-based controller can be taken out of the loop without causing a shut-down of the process. Additionally, the resulting two-stage controller offers improved robustness properties [BiRa 91]. In this context [Mayn 00] emphasises that the basic problem is in the nonconvexity of the NLP which is in general observed for NMPC.

[Rick 91], [Lee 00] emphasise that the difficulties connected with the generation of a suitably accurate nonlinear model pose one of the main obstacles in NMPC besides the timing constraints in the online framework noted above. [Rick 91] considers the combination of nonlinear steady-state models with low order linear time-invariant dynamic models, models in linear time-invariant dynamic form but with time-varying parameters, and nonlinear ODE (state-space) models. [Lee 00] also discusses nonlinear extensions of modelling techniques known from linear system identification such as *NARMAX*. In this context *neural networks* are of special interest as an alternative to the complexity in the development of models based on physical knowledge (*fundamental models*) and to the difficult model identification procedure required [Rick 91]. Practical issues of neural networks in process modelling are discussed in, e.g., [Qin 97]. In the long term it is to be expected that both fundamental models and empirical models will be tied together in order to complement each other [Miha 99], [Lee 00].

Stability and robustness for NMPC are issues under active investigation. Several methods have been proposed in order to arrive at NMPC schemes with guaranteed stability. Addition of a *terminal equality state constraint* to the online optimal control problems has been used, e.g., in [MaMi 90]. However, due to numerical and theoretical restrictions this approach is not recommended for practical application [ChAl 98]. For a class of finite horizon NMPC applications with nonlinear time invariant ODE model and control constraints [YaPo 93], [Oliv 96], [OlMo 96a], [OlMo 96b] obtain stability by addition of a *contractive constraint* to the repeatedly solved optimisation problems. Again, this technique is not suitable for practical applications [ChAl 98]. [MiMa 93] have proposed to employ a receding horizon controller in order to reach a *terminal region*, and then switch to a stable local linear controller within this region. The resulting *dual-mode controller* is then stable. The main problem of the dual-mode controller is in the computation of the terminal region and of the stabilising linear controller. Starting from the idea of the dual-mode controller *quasi-infinite horizon NMPC* has been developed, cf., e.g., [ChAl 96], [ChAl 98]. Similar to the dual-mode controller the infinite time horizon is split into a finite interval and an infinite remainder. In quasi-infinite horizon NMPC then the contribution of the infinite part to the objective is taken into account by a *terminal penalty* term. As the terminal penalty can be determined offline only a finite horizon optimal control problem needs to be solved

online. Further information on stability and robustness in the context of NMPC can be found in [ChAl 98]. [ChAl 98] also state that a general robustness theory for NMPC is still missing. [NMS 00] give a concise and practically oriented comparison of the various approaches employed in order to obtain NMPC algorithms with guaranteed stability properties. In the context of robustness they especially mention $\mathcal{H}_\infty$ control as a promising technique. [AlDo 97] address stability and robustness analysis for NMPC, controllability and observability of nonlinear control systems, and nonlinearity measures in a differential geometric control framework.

Remark 2.5:
An early treatise on controllability and observability of linear time invariant DAEs has been given in [YiSi 81]. ◊

[FiAl 99] examine NMPC with index-1 DAE models. They conclude that most of the MPC schemes developed for ODE models can be applied, eventually after some minor modifications. As an example they design a quasi-infinite horizon NMPC algorithm which uses *control parameterisation* (see Section 2.4) in order to solve the open-loop optimal control problem and apply it to the control of a high-purity binary rectification column. [DUFS 01], [Dieh 01] use *direct multiple shooting* (again, we refer to Section 2.4) in connection with NMPC for index-1 DAE models. In order to improve the real-time capabilities of their approach they employ a special warm start technique for the underlying SQP solver extending, e.g., [LBS 97]. [Bell 97] develop an approach for robust constrained NMPC which handles index-1 DAE as well as – at least theoretically – DAEs of any finite constant index by a derivative-free partitioning technique. The constraints are treated by introduction of slack variables in connection with an interior point method. In order to obtain robustness the (inequality) constraint bounds are modified using linearisation and an estimation based on Gronwall's Lemma such that their feasibility can be guaranteed in a neighbourhood of the nominal solution at least during the control horizon. The optimal control problem is solved numerically by an indirect approach; in order to do so necessary conditions for constrained optimal control problems with an index-1 DAE model are derived. [KuDa 98], [KuDa 99] emphasise that the formulation of a process model as a higher index DAE may be advantageous for controller design if this higher index is caused by the approximation of fast reactions with quasi steady-state conditions. In such a case a model with exact rate conditions is typically stiff due to largely different time scales. This, in turn, can lead to controller ill-conditioning and instability. Therefore they propose an algorithm for the design of controllers based on higher index DAEs which relies on a special index reduction into ODE form.

In the end, however, according to [MoLe 99] fully nonlinear MPC is not widely spread in industry as there has not been a clear justification for the online solution of NLPs together with the underlying nonlinear dynamics. Instead, linearisation approaches are employed, cf., e.g., [Bequ 91a], [Hens 98]. The various methods and their extensions attempt to apply linearisation as far as required in order to arrive at a suitably simple control problem while at the same accounting for the

process nonlinearities in an adequate manner.

2.3.2.d State Estimation, Parameter Estimation, and Data Reconciliation

The *state estimator* is one of the central parts of the MPC algorithm (see Figure 2.6 on page 40) as feedback in MPC controllers is achieved by the update of the current process state used as initial conditions for the solution of the moving optimal control problems [MuEd 97]. *Parameter estimation* is required in order to provide a suitably tuned process model, which is essential for a successful MPC implementation (see above). *Dynamic data reconciliation* primarily is a step in which the imperfect transient measurement data is preprocessed for state and parameter estimation in such a way that it is consistent with the predictions from the model. In general estimates of the process state, of unknown process parameters, and information on the model uncertainty are obtained simultaneously with the adjusted measurements [BBDM 98].

Especially in the nonlinear case these tasks are complex, difficult, and still subject to investigation. Yet we will only pay attention to some aspects in this area as the details are beyond the scope of this treatise. For a more general discussion of state estimation and data reconciliation we refer to, e.g., [LEL 92], [MuEd 97], [BBDM 98], [Soro 98], [ABQRW 99].

[BGW 90] emphasise the close interplay between state estimation and control algorithm. This has also been confirmed in the context of modern control and estimation methods [AHM 98]. In the work of [AHM 98] an optimisation based estimation technique (*receding horizon estimation*, RHE) has shown superior to the classical *extended Kalman filter* (EKF). Similar earlier results have also been summarised in [LEL 92]. [MiMa 95] construct an MPC controller explicitly using a *moving horizon observer*.

The motivation for the time window approach employed in RHE, or synonymously, in *moving horizon estimation* (MHE) is to limit the amount of data used for state estimation due to real-time requirements. Without this restricted point of view the measurement data to be used as the basis for state estimation steadily increases with time, finally rendering the estimation problem intractable. However, as pointed out in [RaRa 00] a plain MHE approach can lead to instability. Therefore, the concept of *arrival cost* has been introduced which reintroduces past data that is not part of the estimation window. In oder to guarantee stability of the estimator it is sufficient to penalise a suitable global lower bound on the arrival cost. Still the problem is to obtain this global bound. Therefore, [RaRa 00] develop an alternative stable MHE estimator which does not require the global lower bound on the arrival cost.

A practical aspect in application is that typically not all measurement data is available at the same time, i.e., some measurements such as temperatures can be obtained frequently, while other measurements such as concentrations are taken

infrequently. Therefore, [Bequ 91b] propose a *multirate approach* for state estimation and data reconciliation, which in the case of RHE amounts to a slightly generalised objective function.

[BBDM 98], [BBDM 01] develop an RHE algorithm for dynamic data reconciliation of DAE models based on a simultaneous optimal control strategy using wavelets. This wavelet-based strategy is especially suitable in a real-time environment as from iteration to iteration the solution is steadily refined. Thus a coarse approximation may be used in case of a lack of time, or an improved approximation may be used if more computational time is available. [SBR 00] consider state and parameter estimation for a class of DAE models. They propose to apply standard LQ filters or EKF to the index reduced system. Nonlinear DAEs are treated by linearisation of the system around a reference trajectory. The problem of consistent initial conditions is not dwelt into deeply. [Grup 96], [HeSt 96], [SWS 95] use optimisation based approaches for parameter identification in DAEs.

2.3.2.e Industrial Application of MPC

[QiBa 97], [QiBa 00] provide overviews of commercially available linear model predictive control algorithms. Additionally, the historical development of the MPC technique is reviewed.

According to [Hens 98] on the one hand currently all industrially available MPC algorithms are based on linear approaches. On the other hand, there are strongly nonlinear processes or processes with largely varying operational conditions which necessitate the application of nonlinear control methods. However, [Hens 98] see a number of open questions in the context of NMPC, e.g., the adequate modelling of nonlinear processes (as already mentioned above), robustness and stability analysis for nonlinear control schemes, a measure for nonlinearity in order to facilitate the choice between linear and nonlinear methods, and finally the investigation of NMPC algorithms that are suitable for application to large-scale problems (such as real-time optimal control of entire plants). As noted above, some of these questions (stability and robustness analysis, nonlinearity measures) have already been addressed, cf., e.g., [AlDo 97].

[OgWr 97] review the state of nonlinear process control in the chemical engineering industry arriving at a similar result as [Hens 98]. They state that due to economical reasons plants have to be operated such that nonlinearities inherent in the processes become more and more obvious. Thus nonlinear control will become increasingly important although presently linear control is the predominant technique.

Still MPC is not the final solution to all control issues. E.g., [GRP 91] emphasise that limitations in process understanding and thus shortcomings in the modelling set the limits in the implementation of MPC control loops. Besides a discussion of model identification and validation issues they therefore introduce the *total control concept*, which contains an explicit *exception handling layer* above

the (MPC) control layer. Within the exception handling layer the process is monitored, abnormal operation is detected, and corrective action is implemented (in practice this layer always exists; it may only not be recognised as such). According to [GRP 91] full automation of the exception layer – and thus of overall plant control – in general is not applicable, as the human process understanding required for the appropriate counteractions is often difficult to fit into an algorithm.

2.3.3 The Overall Optimal Control Concept

Inspired by the idea of model predictive control as discussed in Section 2.3.2 we have developed a concept for an online optimal control method. Our concept is summarised in Figure 2.8. In its design we have taken into account several important characteristics present in our special problem setting:

- The processes to be controlled are of very large scale as we consider highly integrated industrial plants (especially cryogenic air separation plants).
- The processes show a highly nonlinear dynamic behaviour.
- The fulfilment of (nonlinear) path constraints is essential.
- Although the governing time constants of the process are relatively large (cf. Section 2.1.2), also disturbances on smaller time scales have to be considered in order to avoid intolerable drift away from a previously determined optimal operation strategy. Otherwise loss of optimality – or more severe – violation of the constraints may be the consequence.
- Detailed rigorous nonlinear dynamic process models are available.

The concept consists in several interconnected components (bold letters refer to Figure 2.8):

- The *operator* **(a)** specifies the optimal control task. He (or she) defines the final process state to achieve (e.g., $\boldsymbol{x}(t_\mathrm{f})$, respectively $\boldsymbol{x}(t_{\mathrm{f},j})$), the objective to be minimised by optimal control, and the constraints.

  As an example the optimal control task can consist in a load-change of an air separation plant from state A (100% load) to state B (60% load) within 1-2 hours. The product gain is to be maximised without violating safety constraints and product specifications.

  In general, the operator is a human who also supervises the process, e.g, the operating personnel of an air separation plant. The operator may (relatively) infrequently reformulate the optimal control problem, i.e., he (or she) may choose a different objective, may select alternative constraints, or may request a new final state. In case of an air separation plant additionally the downstream customer (such as a steelworks) may request varying product specifications (especially product amounts and purities). These product

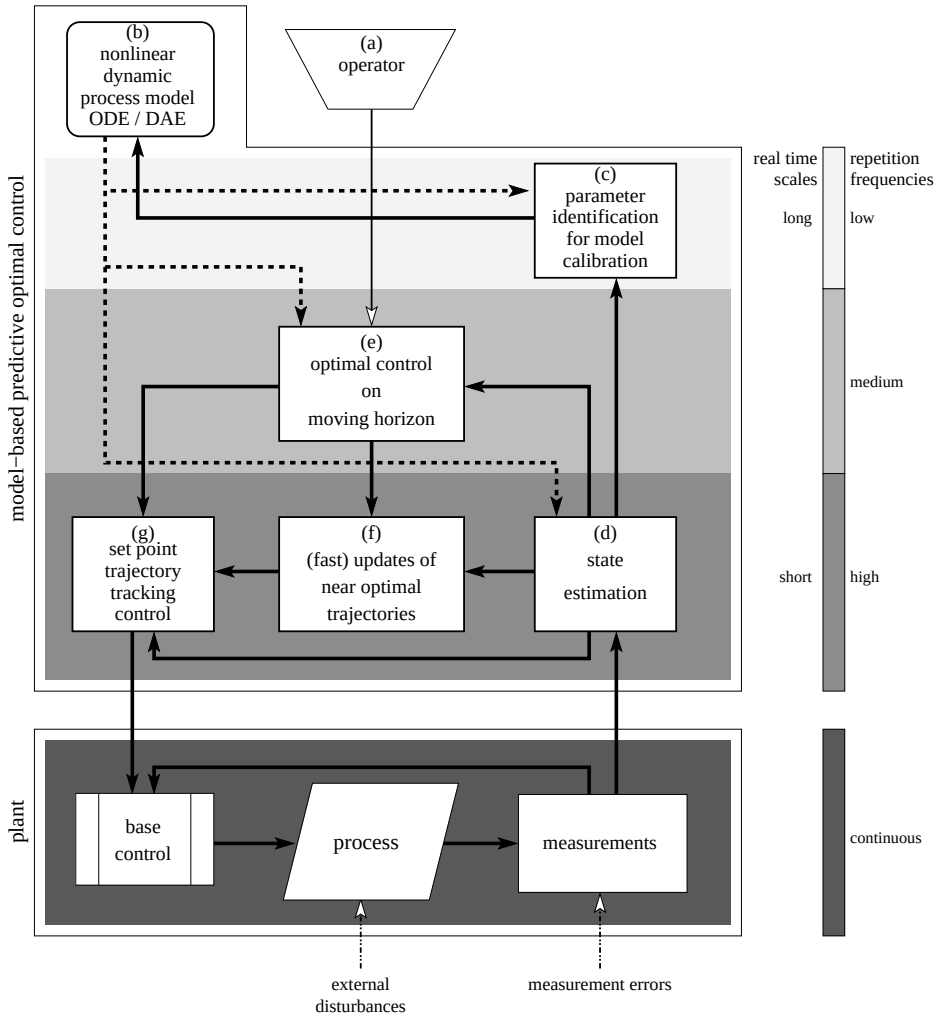

Figure 2.8: Overall concept of a model predictive control algorithm for real-time optimal control of large-scale dynamical systems.
Solid lines indicate the flow of information from one item to another item in the (direction defined by the arrows), the dotted lines indicate in which parts of the control concept the process model enters enters as a basic ingredient of the computations.

specifications are then transformed into corresponding (optimal) points of operation for the air separation plant.

- A very detailed (and thus large-scale) *process model* **(b)** provides the basis of the real-time optimal control concept. It consists in a large system of ODEs or of DAEs (the latter being our model type of interest, see Section

2.2.3).

- In order to preserve the reliability of the model on the long term the model is recalibrated from time to time. To this end the model parameters are estimated by an optimisation based *parameter estimation algorithm* **(c)** using measurement data collected over a longer period of time (as an example consider, e.g., the parameter estimation problem discussed in Section 6.3.2). By this model adaptation disturbances with low frequency or gradual changes in the process can be considered. In the context of chemical engineering systems typical disturbances of this class are, e.g., fouling of heat exchanger areas and decalibrated or malfunctioning sensors.
- The *state estimator* **(d)** determines the process state for all state variables of the process model at the current time. According to the respective application different real-time requirements have to be fulfilled. These requirements increase from parameter identification **(c)** (long term), over optimisation on moving horizon **(e)** (mid term), to fast update of near optimal trajectories **(f)** and setpoint trajectory tracking control **(g)** (short term).

  For state estimation see the discussion and references in Section 2.3.2.d.
- Repeated computation of *optimal controls on a moving horizon*(**e**) determines new optimal setpoint trajectories within the current prediction horizon. The process behaviour is predicted on the basis of the estimation of the process state at the beginning of the prediction horizon and on the nonlinear process model. As optimal control algorithm the open-loop method described in Section 2.4 is employed.

  The optimal controls and the respective state trajectories have to satisfy the specifications (such as objective and constraints) imposed by the operator.
- The *fast update of near optimal trajectories* **(f)** is based on the linearisation of neighbouring parameterised extremals of the solution of the optimal control problem on the current prediction horizon, see Section 2.5 and Section 2.6. It is executed within the current control horizon if the deviation between predicted state and actual state becomes too large.
- The *setpoint trajectory tracking control* **(g)** of the optimal state and control trajectories is primarily required in case of large deviations between process and dynamical model. Such deviations can arise under two conditions:

  1. The numerical computation of the optimal setpoint trajectories on the current prediction horizon (or their near optimal updates, respectively) is slower than required by the prescribed real-time repetition rate. This depends on the size of the dynamical process model, on the computational resources available, and on the optimal control method and its implementation.
  2. Within the prediction horizon the deviation between the progress of the actual process and the model based prediction of the state variable trajectories are too large.

If the open-loop optimal control is used as a pre-control the drag error can be reduced, cf., e.g., [Stry 94] for an application in optimal control of an industrial robot. Also [HAM 00] consider tracking control in the context of MPC.

Altogether, the setpoint trajectory tracking control offers an additional control layer which increases robustness of the overall controller. Furthermore, it can help to avoid shut-down of the process in cases when both the master control problem **(e)** cannot be solved within the required time range and when the fast update of near optimal trajectories **(f)** fails (cf. [BiRa 91], as discussed in Section 2.3.2).

- The *plant* is split up into *base control*, *process*, and *measurements*.

  The base control consists in the automated control system of the plant. Although in this concept the MPC layer **(b)**–**(g)** theoretically should account for most of its original functionality it has to be kept for safety and reliability reasons.

  We consider the lowest level control layer – which in chemical engineering plants is given by, e.g., temperature, flow, pressure, or quality controls – as an integral part of the process itself. These lowest level controllers directly influence the process behaviour and thus have to be accounted for in the process model **(b)** in order to provide a sufficiently precise description of the process for the MPC computations.

2.4 Open-Loop Optimal Control Algorithm

2.4.1 Applied Optimal Control

One can affirm that the charm, exerted all along by the calculus of variations on so many first rate greatnesses of mind, is chiefly traceable to the role which particular problems have played and are playing even today in the development of this theory.

C. Carathéodory, cf. [PeBu 94]

Consider the optimal control problem Eq. (2.3) describing, e.g., the optimal load-change of an air separation plant. Then the principle approach of moving and receding horizon MPC is to achieve an optimal[(ix)] dynamical path during the course of the process in the presence of unmodelled or unpredictable disturbances by solution of a sequence of substitute optimal control problems. Here, substitute optimal control problem means that the time interval may be (and typically is) smaller than the overall optimal control horizon. This has two effects:

[(ix)] Although often applied in an intuitive way, the notion of optimality in a real-time environment is not finally clarified. For a discussion of this topic we refer to [ESBZ 97] and [KSBK 01].

- A closed control loop is obtained by the state feedback induced from employing the actual process state as initial conditions for solving the sequence of optimal control problems (closed-loop characteristics of MPC).

- The optimal control problems on finite horizons (open-loop characteristics of MPC) can be handled with established optimal control techniques even for large-scale processes.

Based on the requirements imposed by our applications of interest we have to select a method for the (open-loop) solution of the optimal control problem(s) on finite horizons. This method will then be discussed in the remaining sections of this chapter.

As reviewed in [BBBB 01] there are three basic approaches to solve optimal control problems with Bolza, Lagrange, or Mayer type objective functionals:

1. *Hamilton-Jacobi-Carathéodory-Bellman* partial differential equations (HJCB) [PeBu 94] and *Dynamic Programming* [Bellm 65],

2. *Calculus of Variations*, *Euler-Lagrange differential equations* (EL-DEQ), and the *Maximum Principle* (*indirect methods*) [PBGM 65], [BrHo 75], and

3. *direct methods* based on a finite dimensional parameterisation of the controls [BoPl 84], [Kraf 85], [Bett 98], [Stry 00].

In order to utilise the advantages of different methods also the combination of indirect methods with direct methods to *hybrid methods* has been proposed in, e.g., [BoPl 84] (direct and indirect multiple shooting), and elaborated for the combination of direct collocation with indirect multiple shooting [StrBu 92], [Stry 94] (for a discussion of the indirect multiple shooting approach we refer to, e.g., [Buli 71], [StBu 90], and the bibliography in [BBBB 01]).

We restrict our choice to direct methods **3** as appropriate extensions of the methods **1** and **2** (and especially the algorithms derived of these methods) to optimal control problems with large-scale higher index DAE models are not known. The difficulties are both of theoretical and practical nature, cf., e.g., [BBBB 01]. Though, some results regarding the Calculus of Variations [Jonc 88] and the Maximum Principle [LiYa 88] regarding optimal control problems with DAE state equations are available. More recent investigations concerning necessary conditions for optimal control problems with index-1 DAE models have been made by [Bell 97] (there mixed state and control path constraints are considered) and [PiVi 97] (their results are restricted to semi-explicit index-1 DAEs and simple control and state constraints).

A common feature of all direct methods is the parameterisation of the controls. This parameterisation restricts the optimal control problems where the candidates for the controls are members of an infinite dimensional class of functions, e.g., $\boldsymbol{u}(\cdot) \in \mathcal{C}^1_{\mathrm{p}}(\cdot, \mathbb{R}^{n_{\boldsymbol{u}}})$, to finite dimensional problems, where the possible controls are

the members of a finite dimensional family of functions $\widehat{\boldsymbol{u}}(\cdot, \boldsymbol{p})$ parameterised by a vector of shape parameters $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$, e.g., piecewise polynomial functions (see Section 2.4.2 below). In other words, *control parameterisation* converts the infinite dimensional optimal control problem, i.e., the determination of a vector of optimal controls $\boldsymbol{u}$, into a finite dimensional *nonlinear programming problem* (NLP), i.e., the determination of an optimal vector of shape parameters $\boldsymbol{p}$. The computationally attractive feature of direct methods is that these NLPs can be solved by *sequential quadratic programming* (SQP) which will be discussed in Section 2.4.2.b. The relatively recent development of these efficient optimisation algorithms provides a key prerequisite for the successful application of direct methods to real-world problems.

The difference between the various direct methods is found in the extent to which also the dynamical part of the optimal control problem, i.e., the model equations, is subject to parameterisation.

On the one hand, there is the *direct single shooting* method. Here, the model IVP is repeatedly integrated for fixed sets of optimisation parameters as requested by the optimisation method. On the other hand, there is the *direct collocation* method where both control and state variables are fully parameterised. Thus, optimisation and integration are treated simultaneously as one problem. This means that state variable trajectories satisfying the model equations in general are available only after convergence of the optimisation method at the optimal point. The third popular method, *direct multiple shooting*, takes a position in between direct single shooting and direct collocation. Similar to direct single shooting the controls are parameterised. Additionally, a grid on the integration interval including start and final time is selected (the *multiple shooting nodes*), splitting the integration interval into *multiple shooting intervals*. On each multiple shooting interval a new IVP is defined. As in the direct single shooting approach the IVPs are integrated separately. However, the initial conditions at the multiple shooting nodes are added to the optimisation problem as new optimisation parameters. Continuity of the states is enforced by augmenting the parameterised optimisation problem with continuity conditions. Thus the full solution of the dynamic model over the entire optimisation interval is a direct part of the final optimisation problem as in direct collocation.

A detailed discussion and comparison of these three direct methods can be found in [BBBB 01]. Based on this up-to-date review, currently direct single shooting emerges as the preferable approach in the context of our application.

2.4.2 A Direct Single Shooting Algorithm

2.4.2.a Control Parameterisation

Parameterisation of the control variables provides the central step for the transformation of the optimal control problem into an optimisation problem in the single

shooting technique.

Often a general full (open loop) parameterisation of the controls by piecewise polynomial shape functions $\widehat{\boldsymbol{u}}(t,\boldsymbol{p}) = \sum_{\mu=1}^{m_\mathrm{I}} \widehat{\boldsymbol{u}}_\mu(t,\boldsymbol{p}_\mu)$, $\boldsymbol{p} = (\boldsymbol{p}_1^T,\ldots,\boldsymbol{p}_{m_\mathrm{I}}^T)^T \in \mathbb{R}^{n_{\boldsymbol{p}_\boldsymbol{u}}\cdot m_\mathrm{I}}$, $\boldsymbol{p}_\mu \in \mathbb{R}^{n_{\boldsymbol{p}_\boldsymbol{u}}}$, $\mu = 1,\ldots,m_\mathrm{I}$, $n_{\boldsymbol{p}_\boldsymbol{u}} \in \mathbb{N}$ is applied, where m_I is the number of control intervals and $\widehat{\boldsymbol{u}}_\mu(\cdot) \equiv 0$ outside control interval μ, $\mu = 1,\ldots,m_\mathrm{I}$. If consistent initialisation at every point of the control mesh is to be avoided this parameterisation has to satisfy additional differentiability conditions. As shown in [Gerd 01] these differentiability conditions are related to the highest order time derivative of the controls present in the UODE.

In the large-scale optimal control problems regarding the computation of load-change strategies for cryogenic air separation plants (cf. Section 6.3.5, Section 6.3.6) we employ a tailored low-dimensional global parameterisation with a single control interval ($m_\mathrm{I} = 1$) based on control functions that are used in the conventional (closed loop) automated control of these plants. In this way on the one hand application level knowledge is directly incorporated into the numerical solution of the optimal control problem, and on the other hand the results of the optimal control algorithm are in a form which is immediately accessible to the user. These control functions are of the form $\boldsymbol{u}_\mathrm{ac}(t,\boldsymbol{x})$, $\boldsymbol{u}_\mathrm{ac} : \mathbb{R}^{1+n_{\boldsymbol{x}}} \to \mathbb{R}^{n_{\boldsymbol{u}}}$; we denote their parameterised counterparts as $\widehat{\boldsymbol{u}}_\mathrm{ac}(t,\boldsymbol{x},\boldsymbol{p})$, $\widehat{\boldsymbol{u}}_\mathrm{ac} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{u}}}$. This approach is also known as *specific optimal control* [HiRa 71].

By substitution of the controls $\boldsymbol{u}(\cdot)$ with their parameterised counterparts $\widehat{\boldsymbol{u}}(t,\boldsymbol{p})$ or $\widehat{\boldsymbol{u}}_\mathrm{ac}(t,\boldsymbol{x},\boldsymbol{p})$ and enforcement of the path inequality constraints $\boldsymbol{c}(t,\boldsymbol{x},\boldsymbol{y},\boldsymbol{u})$ on a mesh $\tau_j^{0,\boldsymbol{c}} < \cdots < \tau_j^{M_j,\boldsymbol{c}} \in [t_{0,j};t_{\mathrm{f},j}]$, with $M_j+1 \in \mathbb{N}$ mesh points, ([VSP 94a], [VSP 94b]) the infinite dimensional optimal control problem Eq. (2.3) is converted into the finite dimensional optimisation problem (NLP)

$$\widehat{\mathcal{J}}[\boldsymbol{p}] = \widehat{E}(t_{\mathrm{f},j},\widehat{\boldsymbol{x}}(t_{\mathrm{f},j};\boldsymbol{p}),\widehat{\boldsymbol{y}}(t_{\mathrm{f},j};\boldsymbol{p})) \longrightarrow \min_{\boldsymbol{p}}! \tag{2.4a}$$

subject to

$$\begin{aligned} 0 &\geq \widehat{\boldsymbol{c}}(\tau_j^{\mu,\boldsymbol{c}},\widehat{\boldsymbol{x}}(\tau_j^{\mu,\boldsymbol{c}};\boldsymbol{p}),\widehat{\boldsymbol{y}}(\tau_j^{\mu,\boldsymbol{c}};\boldsymbol{p}),\boldsymbol{p})\,;\ \mu = 1,\ldots,M_j\,,\\ 0 &\geq \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(t_\mu,\widehat{\boldsymbol{x}}(t_\mu;\boldsymbol{p}),\widehat{\boldsymbol{y}}(t_\mu;\boldsymbol{p}),\boldsymbol{p})\,;\ \mu = 1,\ldots,m_{\boldsymbol{k}^{\mathrm{ic}}}\,, \end{aligned} \tag{2.4b}$$

and

$$\begin{aligned} \dot{\widehat{\boldsymbol{x}}}(t;\boldsymbol{p}) &= \widehat{\boldsymbol{f}}(t,\widehat{\boldsymbol{x}}(t;\boldsymbol{p}),\widehat{\boldsymbol{y}}(t;\boldsymbol{p}),\boldsymbol{p},\operatorname{sign}\widehat{\boldsymbol{q}})\,;\ t \in [t_{0,j},t_{\mathrm{f},j}]\,,\\ 0 &= \widehat{\boldsymbol{g}}(t,\widehat{\boldsymbol{x}}(t;\boldsymbol{p}),\widehat{\boldsymbol{y}}(t;\boldsymbol{p}),\boldsymbol{p},\operatorname{sign}\widehat{\boldsymbol{q}})\,,\\ 0 &= \widehat{\boldsymbol{k}}^{\mathrm{ini}}(t_{0,j},\widehat{\boldsymbol{x}}(t_{0,j};\boldsymbol{p}),\widehat{\boldsymbol{y}}(t_{0,j};\boldsymbol{p}),\dot{\widehat{\boldsymbol{x}}}(t_{0,j};\boldsymbol{p}),\boldsymbol{p})\,. \end{aligned} \tag{2.4c}$$

Here, we use $\widehat{\cdot}$ as a qualifier in order to denote the functions and variables which are derived by the parameterisation of the respective terms in Eq. (2.3).

Remark 2.6:

After discretisation the path inequality constraints are enforced at the mesh points only. This means that reasonable results (with respect to the original optimal control problem) can only be expected if the frequency of the constraints is small in comparison to the density of the mesh points and the parameterisation of the control. In order to improve the confidence in the solution obtained from Eqs. (2.4a)–(2.4c), e.g., penalty terms on the integral error as discussed in Section 2.3.1, Remark 2.2, can be introduced [VSP 94b]. ◊

2.4.2.b Sequential Quadratic Programming

By parameterisation the optimal control problem is *transformed* into a finite dimensional optimisation problem. The transformed problem is suitable for *solution* by an optimisation algorithm. Due to their efficiency, robustness, and broad range of application (as already mentioned in Section 2.4.1) we employ SQP methods for the solution of the NLP Eqs. (2.4a)–(2.4c). Several sophisticated SQP algorithms are available, e.g., NPSOL [GMSW 98], SNOPT [GMS 97b], or SOCS© [BeHu 97] – only to enumerate some of them. [Wrig 02] briefly discusses some available SQP algorithms, as well as implementations of other optimisation methods.

In the sequel we discuss some of the basics of a certain type of SQP methods which we employ (*infeasible path line-search based* SQP methods with *differentiable augmented Lagrangian merit function*). Detailed information is provided in, e.g., [Flet 87], [GMW 95]. [GoTo 00] discuss various approaches of state-of-the-art SQP methods for the solution of large-scale NLPs covering also *trust region* methods which make up the second large class of SQP methods.

For ease of notation we consider the general class of NLPs

$$\begin{aligned} \overline{\Phi}(\overline{\boldsymbol{p}}) &\longrightarrow \min_{\overline{\boldsymbol{p}}}! \\ \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}) = 0\,; 1 \le \mu \le m_{\mathrm{ec}}\,, &\quad \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}) \ge 0\,; 1 \le \nu \le m_{\mathrm{ic}}\,, \end{aligned} \tag{2.5}$$

where $\overline{\boldsymbol{p}} \in \mathbb{R}^{\overline{p}}$ are the optimisation variables, $\overline{\Phi} : \mathbb{R}^{\overline{p}} \to \mathbb{R}$ is the objective, $\boldsymbol{a}_\mu : \mathbb{R}^{\overline{p}} \to \mathbb{R}$, $\mu = 1, \ldots, m_{\mathrm{ec}}$, are equality constraints, and $\boldsymbol{b}_\nu : \mathbb{R}^{\overline{p}} \to \mathbb{R}$, $\nu = 1, \ldots, m_{\mathrm{ic}}$, denote inequality constraints The qualifier $\bar{\cdot}$ refers to variables and functions which are used in the context of the general NLP Eq. (2.5). As SQP is based on derivative information objective and constraints are assumed to be twice continuously differentiable. Closely connected to the NLP Eq. (2.5) is its *Lagrangian function*

$$L(\overline{\boldsymbol{p}}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \overline{\Phi}(\overline{\boldsymbol{p}}) - \sum_{\mu=1}^{m_{\mathrm{ec}}} \boldsymbol{\alpha}_\mu \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}) - \sum_{\nu=1}^{m_{\mathrm{ic}}} \boldsymbol{\beta}_\nu \boldsymbol{b}_\nu(\overline{\boldsymbol{p}})\,. \tag{2.6}$$

The additional variables $\boldsymbol{\alpha} \in \mathbb{R}^{m_{\mathrm{ec}}}$ and $\boldsymbol{\beta} \in \mathbb{R}^{m_{\mathrm{ic}}}$ in Eq. (2.6) are called the *Lagrangian multipliers.*

Remark 2.7:
In non-academic problems global $\mathcal{C}^2$ differentiability in the objective function and in

the constraints often cannot be guaranteed. However, modern SQP algorithms such as NPSOL [GMSW 98] or SNOPT [GMS 97b] can cope with local, isolated discontinuities if they do not appear near the optimum. ◊

In preparation of the following discussion we introduce the notion of an *active (inequality) constraint*, cf., e.g., [GMW 95]:

Definition 2.3 (Active Inequality Constraint of an NLP)
An inequality constraint $\boldsymbol{b}_\nu$, $\nu \in \{1, \ldots, m_{ic}\}$, *of the NLP of Eq.* (2.5) *is said to be* active *at* $\overline{\boldsymbol{p}}$ *if* $\boldsymbol{b}_\nu(\overline{\boldsymbol{p}}) = 0$.

Assume that $\overline{\boldsymbol{p}}^*$ is a local minimiser of Eq. (2.5). Further, let

$$\mathcal{B}(\overline{\boldsymbol{p}}^*) := \{\nu \mid 1 \leq \nu \leq m_{\mathrm{ic}};\ \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}^*) = 0\}$$

be the index set of all active inequality constraints at $\overline{\boldsymbol{p}}^*$. If at least one of the constraints $\boldsymbol{a}_\mu$, $\mu = 1, \ldots, m_{\mathrm{ec}}$, $\boldsymbol{b}_\nu$, $\nu \in \mathcal{B}$, is nonlinear we say that *constraint qualifications* hold at $\overline{\boldsymbol{p}}^*$ if the gradients of the active inequality constraints and the gradients of all equality constraints are linearly independent. If all of the active inequality constraints and all equality constraints are linear we also say that constraint qualifications hold.

Remark 2.8:
Other *constraint qualifications* than the assumption of linearly independent gradients of the active constraints (in the sense of Definition 2.3 all equality constraints are active at a local minimiser) can be employed, cf., e.g., [GMW 95]. ◊

After these preparations the first order necessary conditions of optimality for the NLP of Eq. (2.5) can be formulated as in the following Theorem 2.1, cf., e.g., [Flet 87], [GMW 95], [Stry 00]:

Theorem 2.1 (1^{st} Order NLP Optimality Conditions (KKT-Conditions))
Let $\overline{\Phi}$, $\boldsymbol{a}_\mu$, $\mu = 1, \ldots, m_{ec}$, *and* $\boldsymbol{b}_\nu$, $\nu \in m_{ic}$, *be continuously differentiable functions. If* $\overline{\boldsymbol{p}}^*$ *is a local minimum of the NLP Eq.* (2.5) *such that constraint constraint qualifications hold and that the conditions*

$$\begin{aligned} \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}^*) &= 0\,,\ \mu = 1, \ldots, m_{ec}\,, \\ \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}^*) &= 0\,,\ \nu \in \mathcal{B}(\overline{\boldsymbol{p}}^*)\,, \end{aligned}$$

are satisfied then there exist multipliers $\boldsymbol{\alpha} \in \mathbb{R}^{m_{ec}}$ *and* $\boldsymbol{\beta} \in \mathbb{R}^{m_{ic}}$, *and the following relations hold*

$$\nabla_{\overline{\boldsymbol{p}}} L(\overline{\boldsymbol{p}}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \nabla_{\overline{\boldsymbol{p}}} \overline{\Phi}(\overline{\boldsymbol{p}}^*) - \sum_{\mu=1}^{m_{ec}} \boldsymbol{\alpha}_\mu \nabla_{\overline{\boldsymbol{p}}} \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}^*) - \sum_{\nu=1}^{m_{ic}} \boldsymbol{\beta}_\nu \nabla_{\overline{\boldsymbol{p}}} \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}^*) = 0\,, \tag{2.7a}$$

$$\begin{aligned} \boldsymbol{\alpha}_\mu \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}^*) &= 0\,,\ \mu = 1, \ldots, m_{ec}\,, \\ \boldsymbol{\beta}_\nu \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}^*) &= 0\,,\ \nu = 1, \ldots, m_{ic}\,, \end{aligned} \tag{2.7b}$$

$$\boldsymbol{\beta}_\nu \geq 0\,,\ \nu = 1, \ldots, m_{ic}\,. \tag{2.7c}$$

The idea of SQP is to compute a sequence of vectors of optimisation variables $\{\overline{\boldsymbol{p}}^i\}_i$ which converges to the optimal solution of the NLP. Basically, $\{\overline{\boldsymbol{p}}^i\}_i$ is generated by solving a sequence of quadratic subproblems such that $\{\overline{\boldsymbol{p}}^i\}_i$ converges to a solution of the first order necessary optimality conditions Eqs. (2.7a)–(2.7c) of the NLP (therefore the term *sequential quadratic programming*), cf., e.g., [Han 77], [Powe 78], [Schi 81a], [Schi 81b]. These subproblems are set up by linearisation of the constraints and by a quadratic approximation of the *Lagrangian function* of the NLP Eq. (2.5). In each iteration of a line-search SQP method the new optimisation variables and the corresponding multipliers are generated according to

$$\begin{bmatrix}\overline{\boldsymbol{p}}^{i+1}\\ \boldsymbol{\mu}^{i+1}\end{bmatrix} = \begin{bmatrix}\overline{\boldsymbol{p}}^{i}\\ \boldsymbol{\mu}^{i}\end{bmatrix} + \sigma^i \begin{bmatrix}\boldsymbol{d}^i\\ \boldsymbol{\lambda}^i - \boldsymbol{\mu}^i\end{bmatrix} .$$

The *direction of search* $\boldsymbol{d} \in \mathbb{R}^{n_{\overline{\boldsymbol{p}}}}$, the updates $\boldsymbol{\lambda} \in \mathbb{R}^{m_{\mathrm{ec}}+m_{\mathrm{ic}}}$ for the Lagrangian multipliers $\boldsymbol{\mu} := [\boldsymbol{\alpha}^T, \boldsymbol{\beta}^T]^T$, and the *step length* $\sigma \in \mathbb{R}$ are calculated in two consecutive steps:

At first, the direction of search $\boldsymbol{d}^i$ is determined from the *quadratic programming problem* (QP)

$$\begin{aligned} &\frac{1}{2}(\boldsymbol{d}^i)^T H^i \boldsymbol{d}^i + \nabla\overline{\Phi}(\overline{\boldsymbol{p}}^i)^T \boldsymbol{d}^i \longrightarrow \min_{\boldsymbol{d}^i}! \\ &\nabla \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}^i)^T \boldsymbol{d}^i + \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}^i) = 0\,;\ 1 \le \mu \le m_{\mathrm{ec}}\,, \\ &\nabla \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}^i)^T \boldsymbol{d}^i + \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}^i) \ge 0\,;1 \le \nu \le m_{\mathrm{ic}}\,, \end{aligned} \tag{2.8}$$

which arises from a quadratic approximation of the Lagrangian (2.6) and linearisation of the constraints. $H^i \in \mathbb{R}^{n_{\overline{\boldsymbol{p}}} \times n_{\overline{\boldsymbol{p}}}}$ is the Hessian of the Lagrangian L at $\overline{\boldsymbol{p}}^i$. The QP Eq. (2.8) itself is solved by an iterative quadratic programming method which usually employs either an *active set* or an *interior point* strategy.

Remark 2.9:
In order to avoid failure of the whole algorithm due to an unsolvable quadratic subproblem actual implementations use modified versions of Eq. (2.8), cf., e.g., [GMW 95]. ◊

The updates $\boldsymbol{\lambda}^i$ for the Lagrangian multipliers of the NLP are taken as the multipliers connected to the solution of Eq. (2.8). Then the step length σ^i is chosen so as to achieve a (sufficient) decrease of a *test* or *merit function*

$$\theta_{\boldsymbol{\rho}}(\sigma) = \psi_{\boldsymbol{\rho}}\left(\begin{bmatrix}\overline{\boldsymbol{p}}^{i}\\ \boldsymbol{\mu}^{i}\end{bmatrix} + \sigma^i \begin{bmatrix}\boldsymbol{d}^i\\ \boldsymbol{\lambda}^i - \boldsymbol{\mu}^i\end{bmatrix}\right) ,$$

where $\boldsymbol{\rho} \in \{\mathbb{R}_+ \backslash \{0\}\}^{m_{\mathrm{ec}}+m_{\mathrm{ic}}}$ is the vector of positive *penalty parameters*. Different penalty functions $\psi_{\boldsymbol{\rho}}(\cdot)$ have been proposed, e.g., the *differentiable augmented*

Lagrangian function

$$\psi_{\boldsymbol{\rho}}\left(\begin{bmatrix}\overline{\boldsymbol{p}}\\ \boldsymbol{\mu}\end{bmatrix}\right) := \overline{\Phi}(\overline{\boldsymbol{p}}) - \sum_{\mu=1}^{m_{\mathrm{ec}}}\left(\boldsymbol{\alpha}_\mu \boldsymbol{a}_\mu(\overline{\boldsymbol{p}}) - \frac{1}{2}\boldsymbol{\rho}_\mu \boldsymbol{a}_\mu(\overline{\boldsymbol{p}})^2\right) - \sum_{\nu\in\mathcal{B}_{\boldsymbol{\rho}}(\overline{\boldsymbol{p}})}\left(\boldsymbol{\beta}_\nu \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}) - \frac{1}{2}\boldsymbol{\rho}_{\nu+m_{\mathrm{ec}}} \boldsymbol{b}_\nu(\overline{\boldsymbol{p}})^2\right) - \frac{1}{2}\sum_{\nu\in\widetilde{B}_{\boldsymbol{\rho}}(\overline{\boldsymbol{p}})}\frac{\boldsymbol{\beta}_\nu^2}{\boldsymbol{\rho}_{\nu+m_{\mathrm{ec}}}},$$

with the set $\mathcal{B}_{\boldsymbol{\rho}}(\overline{\boldsymbol{p}}) := \left\{\nu \mid 1 \le \nu \le m_{\mathrm{ic}};\ \boldsymbol{b}_\nu(\overline{\boldsymbol{p}}) \le \boldsymbol{\beta}_\nu/\boldsymbol{\rho}_{\nu+m_{\mathrm{ec}}}\right\}$ and its complement $\widetilde{B}_{\boldsymbol{\rho}}(\overline{\boldsymbol{p}}) := \{1,\ldots,m_{\mathrm{ic}}\} \setminus \mathcal{B}(\overline{\boldsymbol{p}})$, cf., e.g., [GMW 95].

An obvious problem connected to the QP Eq. (2.8) is that 2nd order derivative information of the Lagrangian (2.6) is needed which in practice is rarely available (see, however, the exceptions enumerated in Remark 2.10 on page 60). Therefore SQP algorithms mostly use an approximated Hessian $\overline{H}$ instead of the exact Hessian H. A popular technique is the BFGS scheme, where, e.g., starting from the identity matrix, a positive definite approximation to the Hessian is constructed from updates after each SQP iteration according to

$$\begin{aligned}\boldsymbol{y}^i &:= \nabla_{\overline{\boldsymbol{p}}} L(\overline{\boldsymbol{p}}^{i+1}, \boldsymbol{\alpha}^i, \boldsymbol{\beta}^i) - \nabla_{\overline{\boldsymbol{p}}} L(\overline{\boldsymbol{p}}^i, \boldsymbol{\alpha}^i, \boldsymbol{\beta}^i)\,,\\ \boldsymbol{s}^{i+1} &:= \overline{\boldsymbol{p}}^{i+1} - \overline{\boldsymbol{p}}^i\,,\\ \overline{H}^{i+1} &:= \overline{H}^i - \frac{1}{(\boldsymbol{s}^i)^T\overline{H}^i\boldsymbol{s}^i}\overline{H}^i(\boldsymbol{s}^i)^T\boldsymbol{s}^i\overline{H}^i + \frac{1}{(\boldsymbol{y}^i)^T\boldsymbol{s}^i}\boldsymbol{y}^i(\boldsymbol{y}^i)^T\,.\end{aligned}$$

In order to preserve a positive definite approximate Hessian the update has to be modified in case of $(\boldsymbol{y}^i)^T\boldsymbol{s}^i \le 0$. Other methods for the approximation of the Hessian are discussed, e.g., in [BCH 97].

In application frequently special NLPs with a quadratic objective

$$\overline{\Phi}(\overline{\boldsymbol{p}}) = \sum_{\mu=1}^{m_\psi}\psi_\mu(\overline{\boldsymbol{p}})^2 \longrightarrow \min_{\overline{\boldsymbol{p}}}!$$

have to be considered. In the context of direct shooting methods such NLPs arise if the objective of the underlying optimal control problem is a sum of quadratic terms e.g., if the model parameters of the state equations are to be identified by a weighted l_2-criterion, or if the deviation of the trajectory from a prescribed path has to be minimised (with the deviation measured in the L_2-norm), see also Remark 2.3 on page 39. The point is that the Hessian H of the quadratic cost functional $\overline{\Phi}$ in Eq. (2.4.2.b) reads as

$$H(\overline{\boldsymbol{p}}) = 2\left[J(\overline{\boldsymbol{p}})^T J(\overline{\boldsymbol{p}}) + Q(\overline{\boldsymbol{p}})\right]$$

with

$$Q(\overline{\boldsymbol{p}}) := \sum_{\mu=1}^{m_\psi} \psi_\mu(\overline{\boldsymbol{p}}) \widetilde{H}_\mu(\overline{\boldsymbol{p}}) \,,$$

where $J(\overline{\boldsymbol{p}})$ is the Jacobian of $\psi(\overline{\boldsymbol{p}}) := [\psi_1, \ldots, \psi_{m_\psi}]^T(\overline{\boldsymbol{p}})$, and $\widetilde{H}_\mu(\overline{\boldsymbol{p}})$ are the Hessians of $\psi_\mu(\overline{\boldsymbol{p}})$, $\mu = 1, \ldots, m_\psi$. This special structure of $H(\overline{\boldsymbol{p}})$ can (and should) be exploited [GMW 95], [Björ 96]. E.g., for the "classic" SQP algorithm NPSOL [GMSW 98] the especially tailored derivate NLSSOL is available [Wrig 02]. For more general information on the solution of nonlinear least squares problems we refer to, e.g., [GiMu 78], [GMW 95], [Björ 96].

2.4.2.c Tying together SQP, Integrator, and Sensitivity Analysis

Before the parameterised optimal control problem Eqs. (2.4a)–(2.4c) can be solved by a standard SQP algorithm this NLP has to be stated in the standard form Eq. (2.5). As easily seen, the equivalences are

$$\begin{aligned}
\overline{\boldsymbol{p}} &:= \boldsymbol{p} \,, \\
\overline{\Phi}(\overline{\boldsymbol{p}}) &:= \widehat{E}(t_{\mathrm{f},j}, \widehat{\boldsymbol{x}}(t_{\mathrm{f},j}; \boldsymbol{p}), \widehat{\boldsymbol{y}}(t_{\mathrm{f},j}; \boldsymbol{p})) \,, \\
\boldsymbol{b}_\mu(\overline{\boldsymbol{p}}) &:= -\widehat{\boldsymbol{c}}(\tau_j^{\mu,\boldsymbol{c}}, \widehat{\boldsymbol{x}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}), \widehat{\boldsymbol{y}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}), \boldsymbol{p}) \,; \mu = 1, \ldots, M_j \,, \\
\boldsymbol{b}_{\mu+M_j}(\overline{\boldsymbol{p}}) &:= -\widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(t_\mu, \widehat{\boldsymbol{x}}(t_\mu; \boldsymbol{p}), \widehat{\boldsymbol{y}}(t_\mu; \boldsymbol{p}), \boldsymbol{p}) \,; \; \mu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ic}}} \,, \\
m_{\mathrm{ic}} &= M_j + m_{\boldsymbol{k}^{\mathrm{ic}}} \,,
\end{aligned}$$

where the state variable trajectories $\widehat{\boldsymbol{x}}(t; \boldsymbol{p})$, $\widehat{\boldsymbol{y}}(t; \boldsymbol{p})$ are determined by the IVP Eq. (2.4c).

As discussed in Section 2.4.2.b above SQP algorithms commonly require for fixed set of optimisation parameters the values of the objective function and of the point constraints $\overline{\Phi}(\overline{\boldsymbol{p}})$ and $\boldsymbol{b}_\mu(\overline{\boldsymbol{p}})$, $\mu = 1, \ldots, m_{\mathrm{ic}}$, as well as their first order derivatives with respect to the optimisation parameters $\mathrm{d}\overline{\Phi}(\overline{\boldsymbol{p}})/\mathrm{d}\overline{\boldsymbol{p}}$ and $\mathrm{d}\boldsymbol{b}_\mu(\overline{\boldsymbol{p}})/\mathrm{d}\overline{\boldsymbol{p}}$, $\mu = 1, \ldots, m_{\mathrm{ic}}$. The evaluation of the objective and of the point constraints basically reduces to the integration of the model equations. For the derivative of the objective we have

$$\frac{\mathrm{d}\overline{\Phi}(\overline{\boldsymbol{p}})}{\mathrm{d}\overline{\boldsymbol{p}}} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{p}} \widehat{E}(t_{\mathrm{f},j}, \widehat{\boldsymbol{x}}(t_{\mathrm{f},j}; \boldsymbol{p}), \widehat{\boldsymbol{y}}(t_{\mathrm{f},j}; \boldsymbol{p})) = \left[\frac{\partial \widehat{E}}{\partial \widehat{\boldsymbol{x}}} \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{p}} + \frac{\partial \widehat{E}}{\partial \widehat{\boldsymbol{y}}} \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{p}} + \frac{\partial \widehat{E}}{\partial \boldsymbol{p}} \right]_{t_{\mathrm{f},j}, \boldsymbol{p}} .$$

The derivatives for the constraints are found accordingly. Assuming that the partial derivatives of objective and constraints with respect to their arguments are known the missing piece of information are the sensitivity functions $\partial \widehat{\boldsymbol{x}}/\partial \boldsymbol{p}(t, \boldsymbol{p})$ and $\partial \widehat{\boldsymbol{y}}/\partial \boldsymbol{p}(t, \boldsymbol{p})$. In Section 2.4.3 we discuss the computation of the sensitivities which generally is one of the most expensive and difficult tasks of the single

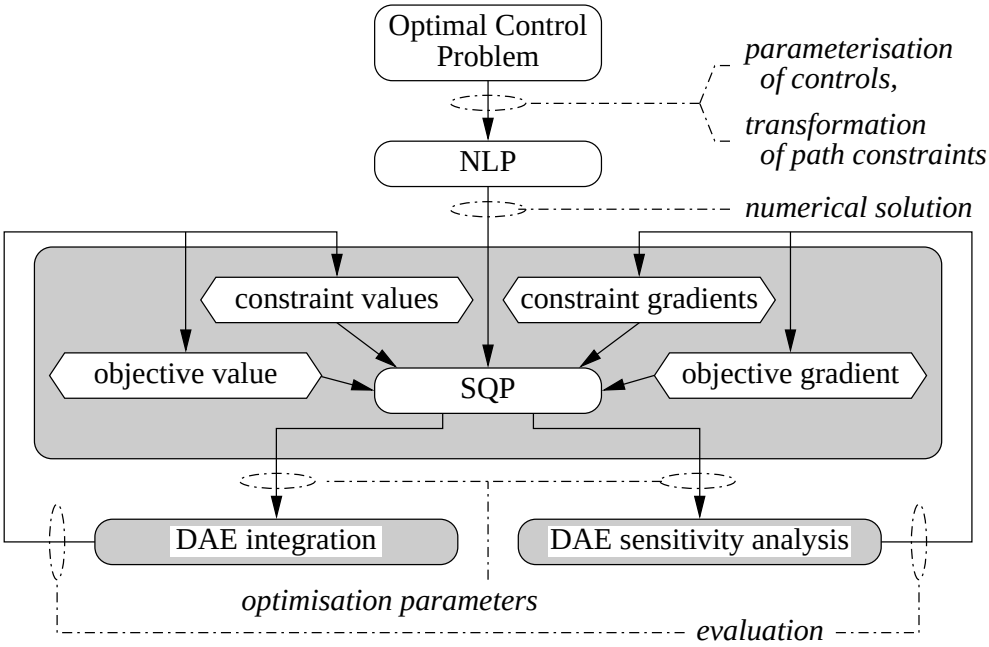

Figure 2.9: Basic structure of the direct single shooting method.

shooting algorithm.

Remark 2.10:
If gradient information is not available some SQP algorithms provide an option to approximate the required first order derivatives by finite differences, e.g., NPSOL [GMSW 98].

On the other hand, some SQP algorithms can additionally use second order derivatives. In this case the Hessian of the Lagrangian can be computed without approximation. The generation of this second order derivative information as well as the potential of its application for the solution of optimal control problems have been discussed, e.g., in [VBB 99], [BBAV 02]. In the context of parameter identification problems [GiMu 78] discusses the utilisation of second order derivative information for the solution of nonlinear least squares problems with modified Gauß-Newton methods. See also [GMW 95]. ◊

Figure 2.9 summarises Section 2.4.2. It depicts the major steps that the direct single shooting method takes in order to obtain the parameterised optimal control for an optimal control problem.

2.4.3 Computation of Parametric Sensitivity Functions by Internal Numerical Differentiation

As discussed in Section 1.4 currently three different numerical approaches are employed in order to compute parametric sensitivity functions for DAE models. These approaches are based on finite difference approximations, integration of the adjoint equations, and integration of the sensitivity equations. For implementation an IND or an END approach may be chosen. Each of these methods has its special

advantages and disadvantages. Unless dictated non-numerical by reasons such as, e.g., the availability of software, in non-standard applications the method of choice is strongly problem dependent.

In our case the boundary conditions are given by the simulation and optimisation environment OPTISIM®, by the (parameterised) optimal control problem Eqs. (2.4a)–(2.4c), and by the chemical engineering application (cf. Section 2.1: load-change of a cryogenic air separation plant). Especially the following four items have to be considered:

a. The model is a very large-scale index-2 DAE system with state and time dependent discontinuities.

b. Inequality constraints are enforced on sampling points within the integration horizon.

c. The dimension of the vector of optimisation parameters is typically small in relation to the size of the state vector.

d. The source code of the (BDF-)integrator within OPTISIM® can be accessed.

From item **a** we conclude that a tailored implementation for the computation of the sensitivities is required. Items **b** and **c** imply that the sensitivities may be needed at many points along the integration horizon but that only a limited number of parameters has to be considered. Finally, direct access to the integrator source code (item **d**) allows development of a robust and efficient IND method. Therefore, our method of choice ([Kron 98], [EKKS 99]) is the integration of the sensitivity DAE of the model by *differentiation of the integrator* in a *staggered direct method* implementation (cf. [LiPe 99]) as proposed in, e.g., [Dunk 84] for ODEs, and by [CaSt 85] and [LeKr 85] in the DAE case.

2.4.3.a The BDF Integrator Method

IND directly refers to the integrator method used. Therefore we consider the parameterised optimal control problem Eqs. (2.4a)–(2.4c) and focus on the model DAE initial value problem Eq. (2.4c)

$$\begin{aligned} \dot{\widehat{\boldsymbol{x}}}(t;\boldsymbol{p}) &= \widehat{\boldsymbol{f}}(t,\widehat{\boldsymbol{x}}(t;\boldsymbol{p}),\widehat{\boldsymbol{y}}(t;\boldsymbol{p}),\boldsymbol{p},\operatorname{sign}\widehat{\boldsymbol{q}})\,;\ t\in[t_{0,j},t_{\mathrm{f},j}]\,,\\ 0 &= \widehat{\boldsymbol{g}}(t,\widehat{\boldsymbol{x}}(t;\boldsymbol{p}),\widehat{\boldsymbol{y}}(t;\boldsymbol{p}),\boldsymbol{p},\operatorname{sign}\widehat{\boldsymbol{q}})\,,\\ 0 &= \widehat{\boldsymbol{k}}^{\mathrm{ini}}(t_{0,j},\widehat{\boldsymbol{x}}(t_{0,j};\boldsymbol{p}),\widehat{\boldsymbol{y}}(t_{0,j};\boldsymbol{p}),\dot{\widehat{\boldsymbol{x}}}(t_{0,j};\boldsymbol{p}),\boldsymbol{p})\,. \end{aligned} \tag{2.9}$$

Given a fixed set of parameters $\boldsymbol{p}$, OPTISIM® [Burr 93] integrates the model equations using a method based on *backward difference formulas* (BDF), cf., e.g., [Gear 71], [BGH 72], [HoLi 83], [LöPe 86], [PeLö 86], [ByHi 87], [HW 91], [DeBo 94], [BCP 96]. BDF methods are implicit linear multi-step methods with variable order and variable step-size suitable for the numerical solution of index-1 DAEs. With a

modified error control criterion they can also be used for the direct integration of semi-explicit index-2 DAEs [BCP 96]. [MäTi 94], [Tisc 95] discuss feasibility and stability of BDF when applied to a class of linear implicit DAEs of tractability index two. In [MäTi 97] the results of [Tisc 95] are specified for DAEs arising from charge-oriented modified nodal analysis of electrical circuits. By a perturbation analysis [Arno 95] arrives at sharper error bounds than were presented in [Tisc 95] and provides additional theoretical justification for the modified error control criterion for index-2 DAEs.

In the n^{th} integration step of a BDF method the interpolating polynomial of the $k-1$ previously computed points $(\widehat{\boldsymbol{x}}_{n-k+1}, \widehat{\boldsymbol{y}}_{n-k+1})$, ..., $(\widehat{\boldsymbol{x}}_n, \widehat{\boldsymbol{y}}_n)$ and of the next point $(\widehat{\boldsymbol{x}}_{n+1}, \widehat{\boldsymbol{y}}_{n+1})$ is formally constructed. The new point $(\widehat{\boldsymbol{x}}_{n+1}, \widehat{\boldsymbol{y}}_{n+1})$ is determined by the condition that the interpolating polynomial has to fulfil the DAE at t_{n+1}. By extrapolation of the interpolating polynomial of $(\widehat{\boldsymbol{x}}_{n-k}, \widehat{\boldsymbol{y}}_{n-k})$, ..., $(\widehat{\boldsymbol{x}}_n, \widehat{\boldsymbol{y}}_n)$ for t_{n+1}, estimates (*predictors*) $\widehat{\boldsymbol{x}}_{n+1}^{\text{pred}}$, $\widehat{\boldsymbol{y}}_{n+1}^{\text{pred}}$, and $\dot{\widehat{\boldsymbol{x}}}_{n+1}^{\text{pred}}$ are obtained. The *fixed leading coefficient BDF* method results in the nonlinear system of equations

$$\begin{aligned} 0 &= \widehat{\boldsymbol{f}}(t_{n+1}, \widehat{\boldsymbol{x}}_{n+1}, \widehat{\boldsymbol{y}}_{n+1}, \boldsymbol{p}, \operatorname{sign} \widehat{\boldsymbol{q}}_n) - \dot{\widehat{\boldsymbol{x}}}_{n+1}^{\text{pred}} + \frac{\alpha_k}{h_{n+1}} (\widehat{\boldsymbol{x}}_{n+1} - \widehat{\boldsymbol{x}}_{n+1}^{\text{pred}}) \,, \\ 0 &= \widehat{\boldsymbol{g}}(t_{n+1}, \widehat{\boldsymbol{x}}_{n+1}, \widehat{\boldsymbol{y}}_{n+1}, \boldsymbol{p}, \operatorname{sign} \widehat{\boldsymbol{q}}_n) \,, \qquad \widehat{\boldsymbol{q}}_n := \widehat{\boldsymbol{q}}(t_n, \widehat{\boldsymbol{x}}_n, \widehat{\boldsymbol{y}}_n, \boldsymbol{p}) \,. \end{aligned} \tag{2.10}$$

which must be solved numerically for $\widehat{\boldsymbol{x}}_{n+1}$ and $\widehat{\boldsymbol{y}}_{n+1}$. The *leading coefficient* $\alpha_k \in \mathbb{R}$ depends on the order k of the method. $h_{n+1} = t_{n+1} - t_n$ is the step-size of the current integration step.

The *corrector system* Eq. (2.10) is solved by a modified Newton algorithm. The initial guess is provided by the evaluation of the BDF predictor polynomial at the new time step

$$\widehat{\boldsymbol{x}}_{n+1}^{[0]} := \widehat{\boldsymbol{x}}_{n+1}^{\text{pred}}, \quad \widehat{\boldsymbol{y}}_{n+1}^{[0]} := \widehat{\boldsymbol{y}}_{n+1}^{\text{pred}} \,.$$

In order to increase the domain of convergence a damped Newton iteration is performed; $m = 1, 2, \ldots$, denotes the current iterate:

$$\begin{bmatrix} \frac{\alpha_k}{h_{n+1}} \boldsymbol{Id} + \widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{x}}} & \widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{y}}} \\ \widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{x}}} & \widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{y}}} \end{bmatrix} \begin{bmatrix} \Delta\widehat{\boldsymbol{x}}^{[m]} \\ \Delta\widehat{\boldsymbol{y}}^{[m]} \end{bmatrix} = - \begin{bmatrix} \widehat{\boldsymbol{f}} - \dot{\widehat{\boldsymbol{x}}}_{n+1}^{\text{pred}} + \frac{\alpha_k}{h_{n+1}} (\widehat{\boldsymbol{x}}_{n+1}^{[m]} - \widehat{\boldsymbol{x}}_{n+1}^{\text{pred}}) \\ \widehat{\boldsymbol{g}} \end{bmatrix} \,, \tag{2.11}$$

$$\begin{bmatrix} \widehat{\boldsymbol{x}}_{n+1}^{[m+1]} \\ \widehat{\boldsymbol{y}}_{n+1}^{[m+1]} \end{bmatrix} := \begin{bmatrix} \widehat{\boldsymbol{x}}_{n+1}^{[m]} \\ \widehat{\boldsymbol{y}}_{n+1}^{[m]} \end{bmatrix} + c \begin{bmatrix} \Delta\widehat{\boldsymbol{x}}^{[m]} \\ \Delta\widehat{\boldsymbol{y}}^{[m]} \end{bmatrix} \;; \quad 0 < c \leq 1 \,.$$

In each step of the Newton iteration the linear system Eq. (2.11) is solved by a hierarchical algorithm efficiently applying modern direct sparse matrix techniques to the large, sparse, and unstructured system matrix [ELBK 97]

$$D = \begin{bmatrix} \frac{\alpha_k}{h_{n+1}} \boldsymbol{Id} + \widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{x}}} & \widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{y}}} \\ \widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{x}}} & \widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{y}}} \end{bmatrix} \,.$$

The treatment of discontinuities is based on the *discontinuity locking* technique as described in Section 1.3.2, cf. Figure 1.1 on page 19. During the solution of the corrector system Eq. (2.10) the switching functions keep their fixed value $\widehat{\boldsymbol{q}}_n$ from the start of the integration step, i.e., switching is suppressed. *After* convergence of the corrector iteration the switching functions are evaluated at the new point $[\widehat{\boldsymbol{x}}_{n+1}^T, \widehat{\boldsymbol{y}}_{n+1}^T]^T$. If at least one switching function changes its sign

$$\operatorname{sign} \widehat{\boldsymbol{q}}(t_n, \widehat{\boldsymbol{x}}_n, \widehat{\boldsymbol{y}}_n, \boldsymbol{p}) \neq \operatorname{sign} \widehat{\boldsymbol{q}}(t_{n+1}, \widehat{\boldsymbol{x}}_{n+1}, \widehat{\boldsymbol{y}}_{n+1}, \boldsymbol{p})$$

the passage of a discontinuity is asserted. In this case the discontinuity is located by inverse interpolation and integration is restarted at this point. In the context of integration with BDF methods the restart may be restricted to cutting down the order of the integrator to one (implicit Euler's method). This is a very simple and in most practical cases successful technique; however, from the numerical point of view *consistent initialisation* is necessary in order to arrive at reliable results. Consistent initialisation is in itself a very complex and difficult task which will be discussed in depth within Chapter 3.

2.4.3.b IND: Differentiation of the Integrator

In the sequel we assume that the integration step from t_n to t_{n+1} has been completed *without* passing a discontinuity; the discontinuous case is subject to investigation in Chapter 4.

Total differentiation of the corrector system Eq. (2.10) with respect to the parameters $\boldsymbol{p}$ yields the *linear* system of equations

$$\begin{aligned} 0 &= \widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{x}}} \boldsymbol{\rho}_{n+1} + \widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{y}}} \boldsymbol{\sigma}_{n+1} + \widehat{\boldsymbol{f}}_{\boldsymbol{p}} - \dot{\boldsymbol{\rho}}^{\text{pred}} + \frac{\alpha_k}{h_{n+1}} (\boldsymbol{\rho}_{n+1} - \boldsymbol{\rho}_{n+1}^{\text{pred}}) \,, \\ 0 &= \widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{x}}} \boldsymbol{\rho}_{n+1} + \widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{y}}} \boldsymbol{\sigma}_{n+1} + \widehat{\boldsymbol{g}}_{\boldsymbol{p}} \,, \end{aligned}$$

or

$$D \begin{bmatrix} \boldsymbol{\rho}_{n+1} \\ \boldsymbol{\sigma}_{n+1} \end{bmatrix} = - \begin{bmatrix} \widehat{\boldsymbol{f}}_{\boldsymbol{p}} - (\dot{\boldsymbol{\rho}}^{\text{pred}} + \frac{\alpha_k}{h_{n+1}} \boldsymbol{\rho}_{n+1}^{\text{pred}}) \\ \widehat{\boldsymbol{g}}_{\boldsymbol{p}} \end{bmatrix} , \tag{2.12}$$

where

$$\boldsymbol{\rho} = \boldsymbol{\rho}(t; \boldsymbol{p}) := \left[\frac{\partial \widehat{\boldsymbol{x}}(t; \boldsymbol{p})}{\partial \boldsymbol{p}} \right] \in \mathbb{R}^{n_{\widehat{\boldsymbol{x}}} \times n_{\boldsymbol{p}}} \text{ and } \boldsymbol{\sigma} = \boldsymbol{\sigma}(t; \boldsymbol{p}) := \left[\frac{\partial \widehat{\boldsymbol{y}}(t; \boldsymbol{p})}{\partial \boldsymbol{p}} \right] \in \mathbb{R}^{n_{\widehat{\boldsymbol{y}}} \times n_{\boldsymbol{p}}}$$

denote the sensitivity matrices.

On the other hand, within an interval where $\operatorname{sign} \widehat{\boldsymbol{q}}$ is constant the sensitivity equations related to the original DAE Eq. (2.9) read as

$$0 = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{p}} \left\{ \begin{bmatrix} \widehat{\boldsymbol{f}}(t, \widehat{\boldsymbol{x}}(t; \boldsymbol{p}), \widehat{\boldsymbol{y}}(t; \boldsymbol{p}), \boldsymbol{p}, \operatorname{sign} \widehat{\boldsymbol{q}}) - \dot{\widehat{\boldsymbol{x}}}(t; \boldsymbol{p}) \\ \widehat{\boldsymbol{g}}(t, \widehat{\boldsymbol{x}}(t; \boldsymbol{p}), \widehat{\boldsymbol{y}}(t; \boldsymbol{p}), \boldsymbol{p}, \operatorname{sign} \widehat{\boldsymbol{q}}) \end{bmatrix} \right\}$$

$$= \begin{bmatrix} \widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{x}}}\boldsymbol{\rho} & +\widehat{\boldsymbol{f}}_{\widehat{\boldsymbol{y}}}\boldsymbol{\sigma} & +\widehat{\boldsymbol{f}}_{\boldsymbol{p}} & -\dot{\boldsymbol{\rho}} \\ \widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{x}}}\boldsymbol{\rho} & +\widehat{\boldsymbol{g}}_{\widehat{\boldsymbol{y}}}\boldsymbol{\sigma} & +\widehat{\boldsymbol{g}}_{\boldsymbol{p}} & \end{bmatrix}_{t,\boldsymbol{p}} . \tag{2.13}$$

Application of the BDF scheme to the sensitivity equations Eq. (2.13) again results in the linear system Eq. (2.12). In both cases, the initial values for the sensitivities are obtained by differentiation of the initial values of the original DAE-IVP

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{p}} \left\{ \widehat{\boldsymbol{k}}^{\mathrm{ini}}(t_{0,j}, \widehat{\boldsymbol{x}}(t_{0,j};\boldsymbol{p}), \widehat{\boldsymbol{y}}(t_{0,j};\boldsymbol{p}), \dot{\widehat{\boldsymbol{x}}}(t_{0,j};\boldsymbol{p}), \boldsymbol{p}) \right\} \\ &= \left[\widehat{\boldsymbol{k}}^{\mathrm{ini}}_{\widehat{\boldsymbol{x}}}\,\boldsymbol{\rho} + \widehat{\boldsymbol{k}}^{\mathrm{ini}}_{\widehat{\boldsymbol{y}}}\,\boldsymbol{\sigma} + \widehat{\boldsymbol{k}}^{\mathrm{ini}}_{\dot{\widehat{\boldsymbol{x}}}}\,\dot{\boldsymbol{\rho}} + \widehat{\boldsymbol{k}}^{\mathrm{ini}}_{\boldsymbol{p}} \right]_{t_{0,j},\boldsymbol{p}} + \left[\widehat{\boldsymbol{k}}^{\mathrm{ini}}_{\widehat{\boldsymbol{x}}}\,\dot{\widehat{\boldsymbol{x}}} + \widehat{\boldsymbol{k}}^{\mathrm{ini}}_{\widehat{\boldsymbol{y}}}\,\dot{\widehat{\boldsymbol{y}}} + \widehat{\boldsymbol{k}}^{\mathrm{ini}}_{\dot{\widehat{\boldsymbol{x}}}}\,\ddot{\widehat{\boldsymbol{x}}} \right]_{t_{0,j},\boldsymbol{p}} \cdot \frac{\partial t_{0,j}}{\partial \boldsymbol{p}} . \end{aligned} \tag{2.14}$$

Here we have treated the more general case, taking into account a possible parametric dependence of the initial time $t_{0,j}$.

Based on these observations (and given that our assumptions on the existence of the sensitivities specified in Section 1.4 hold) the sensitivity matrices can be computed after each integration step for the state variable trajectories by (direct) solution of Eq. (2.12) (*staggered direct method*). This step is computationally cheap since (an approximation of) the matrix D is already available in decomposed form in the modified Newton iteration Eq. (2.11). Moreover, by the equivalence noted above this method corresponds to the (separate) integration of the sensitivity equations with the same step-sizes and order sequence as used for the original DAE which is important in the context of IND.

According to [Feeh 98] there is uncertainty about whether truncation error control must be maintained on both the states *and* the sensitivities. On the one hand, without truncation error control computation of the sensitivities is significantly cheaper and the algorithm is easier to implement than its error controlled counterpart. On the other hand, without a truncation error test the integrator may miss features of the dynamics of the sensitivities. Due to the increased reliability [Feeh 98] recommends the latter choice.

As one part of this work we have extended our sensitivity computation algorithm ([Kron 98], [EKKS 99]) with an option to include the sensitivities in the order and step-size selection strategy, see also Section 5.2. The local truncation error in the sensitivity matrices is estimated basically using the same formulae as employed for the estimation of the local truncation error in the state variables. After a successful integration step in the sensitivity matrices a new step-size and a new order are proposed for the next integration step for the sensitivities. This proposal is based on the local truncation error estimate for the sensitivity matrices and uses the same strategy as employed for the integration of the state variables. Actual order and step-size for the next (combined) integration step in states and sensitivity matrices are determined as the minimum of the order proposals and the minimum of the step-size proposals obtained from state variable and sensitivity error control, respectively. According to our experience the error and step-size

selection strategy including the sensitivities is often more conservative than the purely state variable based strategy. However, (in accordance with [Feeh 98]) the sensitivities appear to be of better quality as in difficult optimal control problems the SQP algorithm converged faster and the optimality criteria could be fulfilled more reliably.

2.5 Neighbouring Extremals

> *Closely associated ... is that [problem] of 'on-line' control. Here the constraint is a novel one mathematically — one not previously encountered in scientific research. We are required to render a decision, perhaps supply a numerical answer, within a specified period of time. It is no longer a question of devising a computationally feasible algorithm; instead we must obtain the best approximation within a specified time.*
>
> Bellman [Bellm 71], Section 14.5

Basically the direct optimal control algorithm in Section 2.4 in connection with an appropriate state estimator is sufficient for setting up a "pure" moving horizon MPC scheme as described in Algorithm 2 on page 40. However, at present for larger processes the time consumed by the state estimation (step **3** in Algorithm 2) and by the computation of the open-loop optimal control on the moving horizon (step **4** in Algorithm 2) is prohibitively long in relation to the process time constants, i.e., the minimum possible control horizon determined by the maximum possible repetition frequency is too large. In other words, due to disturbances the open-loop optimal control policy becomes increasingly inappropriate as the real process dynamics deviate from the model prediction, but a new up-to-date open-loop optimal control policy cannot be provided in appropriate time.

Therefore the validity of a once expensively calculated open-loop control needs to be extended by fast updates in order to compensate for the disturbances. In our context this aim can be achieved by utilising the theory of *neighbouring parameterised extremals*.

The method of linearising neighbouring parameterised extremals has been developed in the 70s and 80s in the context of indirect methods (or boundary value problems arising from the optimal control of ODE systems, respectively), cf., e.g., [BrHo 75], [Kräm 85], [LeBr 89], [Pesc 89a], [Pesc 89b], [KuPe 90a], [KuPe 90b]. The main point is that under some conditions the optimal solution of an optimal control problem smoothly depends on the parameters incorporated in the problem. Given at least $\mathcal{C}^1$ smoothness a first order truncated Taylor's series expansion of the optimal controls (and also of the states) around the original (or *nominal*) parameter set can be established. I.e., the solution for an optimal control problem with a modified parameter set can be linearly approximated based on the solution of the nominal optimal control problem.

The linearisation of neighbouring extremals for parameterised optimal control

problems derived from large-scale discontinuous index-2 DAE process models (cf. Eqs. (2.2a)–(2.2b) in Section 2.2.3) has not been treated previously.

2.5.1 Solution Differentiability

In [MaPes 94], [MaPes 95] solution differentiability of (infinite dimensional) optimal control problems

$$\tilde{\mathcal{J}}[\tilde{\boldsymbol{u}}] = \tilde{E}(\tilde{\boldsymbol{x}}(t_\mathrm{f};\boldsymbol{\pi}),\boldsymbol{\pi}) + \int_{t_0}^{t_\mathrm{f}} \tilde{\Lambda}(\tilde{\boldsymbol{x}}(t;\boldsymbol{\pi}),\tilde{\boldsymbol{u}}(t;\boldsymbol{\pi}),\boldsymbol{\pi})\mathrm{d}t \longrightarrow \min_{\tilde{\boldsymbol{u}}}! \tag{2.15}$$

subject to

$$\begin{aligned}
\dot{\tilde{\boldsymbol{x}}}(t;\boldsymbol{\pi}) &= \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}(t;\boldsymbol{\pi}),\tilde{\boldsymbol{u}}(t;\boldsymbol{\pi}),\boldsymbol{\pi})\,; \quad t \in [t_0,t_\mathrm{f}]\,,\\
0 &= \tilde{\boldsymbol{x}}(0;\boldsymbol{\pi}) - \tilde{\boldsymbol{k}}^{\mathrm{ini}}(\boldsymbol{\pi})\,,\\
0 &\geq \tilde{\boldsymbol{k}}^{\mathrm{ic}}(\tilde{\boldsymbol{x}}(t;\boldsymbol{\pi}),\tilde{\boldsymbol{u}}(t;\boldsymbol{\pi}),\boldsymbol{\pi})\,,\\
0 &= \tilde{\boldsymbol{k}}^{\mathrm{end}}(\tilde{\boldsymbol{x}}(t_\mathrm{f};\boldsymbol{\pi}),\boldsymbol{\pi})\,,
\end{aligned}$$

with respect to the *disturbance parameters* $\boldsymbol{\pi} \in \mathbb{R}^{n_\pi}$ is examined. Please note that the model disturbance parameters $\boldsymbol{\pi}$ are *not* related to the optimisation parameters $\boldsymbol{p}$ as introduced, e.g., by the direct single shooting method discussed in Section 2.4.2. The optimal control task is to find the control $\tilde{\boldsymbol{u}} \in \mathbb{R}^{n_{\tilde{\boldsymbol{u}}}}$ minimising the objective functional $\tilde{\mathcal{J}}$ within the interval in time $t \in [t_0,t_\mathrm{f}]$ subject to the path inequality constraints $\tilde{\boldsymbol{k}}^{\mathrm{ic}} : \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{u}}}+n_{\boldsymbol{\pi}}} \rightarrow \mathbb{R}^{m_{\tilde{\boldsymbol{k}}^{\mathrm{ic}}}}$ and the final point constraints $\tilde{\boldsymbol{k}}^{\mathrm{end}} : \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}+n_{\boldsymbol{\pi}}} \rightarrow \mathbb{R}^{m_{\tilde{\boldsymbol{k}}^{\mathrm{end}}}}$ given a nominal disturbance vector $\boldsymbol{\pi} := \boldsymbol{\pi}_0$. The system dynamics $\tilde{\boldsymbol{x}} \in \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}}$ is determined by the ODE initial value problem $\tilde{\boldsymbol{f}} : \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{u}}}+n_{\boldsymbol{\pi}}} \rightarrow \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}}$, $\tilde{\boldsymbol{k}}^{\mathrm{ini}} : \mathbb{R}^{n_{\boldsymbol{\pi}}} \rightarrow \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}}$.

Examples for disturbance parameters in the context of chemical engineering are, e.g., the composition of the feed for petro-chemical plants, or the ambient temperature in the context of an air separation plant. Also, failure of equipment may be modelled by disturbance parameters, cf., e.g., Section 6.3.4.

The *nominal* (or *unperturbed*) optimal solution

$$\tilde{\boldsymbol{x}}_0^*(t) := \tilde{\boldsymbol{x}}^*(t;\boldsymbol{\pi}_0)\,, \quad \tilde{\boldsymbol{u}}_0^*(t) := \tilde{\boldsymbol{u}}^*(t;\boldsymbol{\pi}_0)\,,$$

is defined as the solution of the optimal control problem Eq. (2.15) for the reference value of the disturbances $\boldsymbol{\pi} = \boldsymbol{\pi}_0$. Associated with the nominal solution are the *adjoint variables* $\boldsymbol{\lambda}_0^*(t) := \boldsymbol{\lambda}^*(t;\boldsymbol{\pi}_0) \in \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}}$.

Based on the BVP derived from the description of the optimal solution of problem Eq. (2.15), *second order sufficient conditions* (*SSC*) (cf., e.g., [MaPi 95]), and a Riccati ODE related to the SSC solution differentiability with respect to the disturbance parameters around the nominal solution can be shown. E.g.,

[BüMa 00] summarise the results in a theorem on solution differentiability for the optimal control problem Eq. (2.15). This theorem states that under some technical conditions (including the SSC) the nominal solution $\tilde{\boldsymbol{x}}_0^*(t), \boldsymbol{\lambda}_0^*(t), \tilde{\boldsymbol{u}}_0^*(t)$ can be embedded into a family of optimal solutions $\tilde{\boldsymbol{x}}^*(t;\boldsymbol{\pi}), \boldsymbol{\lambda}^*(t;\boldsymbol{\pi}), \tilde{\boldsymbol{u}}^*(t;\boldsymbol{\pi})$ of the perturbed problem Eq. (2.15). Moreover, this family is shown to be piecewise of class $\mathcal{C}^1$ for all disturbances $\boldsymbol{\pi}$ in a neighbourhood of the reference parameter $\boldsymbol{\pi}_0$.

The theorem on solution differentiability provides the theoretical justification for the application of neighbouring extremal approaches in order to calculate near optimal approximations to disturbed ($\boldsymbol{\pi} \neq \boldsymbol{\pi}_0$) optimal control problems Eq. (2.15). Given that the conditions of the theorem apply corrections to the nominal solution can be found by truncated Taylor's series expansions. E.g., a first-order correction of the state variable trajectory is obtained by

$$\tilde{\boldsymbol{x}}^*(t;\boldsymbol{\pi}) \doteq \tilde{\boldsymbol{x}}^*(t;\boldsymbol{\pi}_0) + \frac{\partial \tilde{\boldsymbol{x}}^*(t;\boldsymbol{\pi}_0)}{\partial \boldsymbol{\pi}}(\boldsymbol{\pi} - \boldsymbol{\pi}_0)\,.$$

The quantities

$$\frac{\partial \tilde{\boldsymbol{x}}^*(t;\boldsymbol{\pi}_0)}{\partial \boldsymbol{\pi}}, \frac{\partial \tilde{\boldsymbol{u}}^*(t;\boldsymbol{\pi}_0)}{\partial \boldsymbol{\pi}}, \text{ and } \frac{\partial \boldsymbol{\lambda}^*(t;\boldsymbol{\pi}_0)}{\partial \boldsymbol{\pi}}$$

are the *sensitivity differentials* of the solution and of the adjoint variables with respect to the vector of *disturbance parameters* $\boldsymbol{\pi}$ at the nominal solution. The consecutive question is on how to obtain the sensitivity differentials.

2.5.2 BVP-based Methods for ODE Systems

[MaPes 91], [MaPes 93], [MaPes 94], [MaPes 95] show that the sensitivity differentials $\partial(\cdot)/\partial\boldsymbol{\pi}$ satisfy a *linear inhomogeneous boundary value problem* (LBVP). According to the theory of variational calculus the optimal solution of Eq. (2.15) (for any fixed, but otherwise arbitrary value of $\boldsymbol{\pi}$) satisfies necessary conditions which can be reformulated as a BVP. Consider this BVP at the nominal value of the disturbance, $\boldsymbol{\pi} = \boldsymbol{\pi}_0$. Then total differentiation of the BVP for the nominal solution with respect to the disturbance parameters leads to the desired LBVP. A recent discussion of this approach can be found in [MaAu 01].

The BVP-based approach has already been used, e.g., in [Pesc 89a], [Pesc 89b] as a basic ingredient of the *repeated correction method*. In extension of the repeated correction method [KuWe 99] have lately proposed to calculate an update of the nominal path in addition to the near optimal approximation of the optimal solution.

2.5.3 Sensitivity of Parameterised ODE Optimal Control Problems

In [BüMa 98], [BüMa 00], [BüMa 01b] perturbed optimal control problems of the type Eq. (2.15) with a possibly free final time $t_\mathrm{f} \in \mathbb{R}_+$ are considered (for ease of notation we restrict to fixed final time). [BüMa 01b] admit a slightly more general objective as well as general nonlinear boundary constraints instead of the separated boundary constraints in the optimal control problem Eq. (2.15). Especially, the initial state may be subject to optimisation.

In contrast to the previous Section 2.5.2 starting point is the NLP obtained by parameterisation of the controls in the optimal control problem Eq. (2.15). The ODE-IVP is discretised by the implicit Euler's method on a mesh $t_0 < t_1 < \cdots < t_N = t_\mathrm{f}$. Additionally, the Lagrange term in the objective function is approximated by a Riemann sum on the same mesh. The NLP can then be denoted as

$$\begin{aligned} \bar{\mathcal{J}}(\boldsymbol{p}, \boldsymbol{\pi}) =& \bar{E}(\bar{\boldsymbol{x}}_N(\boldsymbol{p}, \boldsymbol{\pi}), t_N, \boldsymbol{\pi}) \\ &+ \sum_{\nu=0}^{N-1} (t_{\nu+1} - t_\nu) \bar{\Lambda}(\bar{\boldsymbol{x}}_\nu(\boldsymbol{p}, \boldsymbol{\pi}), t_\nu, \boldsymbol{\pi}) \longrightarrow \min_{\boldsymbol{p}}! \end{aligned} \tag{2.16}$$

subject to

$$\begin{aligned} 0 &= \bar{G}_\nu(\boldsymbol{p}, \boldsymbol{\pi})\,;\ 1 \le \nu \le r\,, \\ 0 &\le \bar{G}_\nu(\boldsymbol{p}, \boldsymbol{\pi})\,;\ r+1 \le \nu \le m_{\bar{G}}\,. \end{aligned}$$

$\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$ contains the optimisation parameters, i.e., the shape parameters for the parameterised control functions $\tilde{\boldsymbol{u}}(t; \boldsymbol{\pi}) \curvearrowright \bar{\boldsymbol{u}}(t, \boldsymbol{p}; \boldsymbol{\pi})$ (see Section 2.4.2), and eventually the unknown initial state. $\bar{\boldsymbol{x}}_\nu(\boldsymbol{p}, \boldsymbol{\pi}) := \tilde{\boldsymbol{x}}(t_\nu; \boldsymbol{p}, \boldsymbol{\pi})$, $\nu = 0, \ldots, N$, are values of the state variables on the mesh. $\bar{G}_\mu(\boldsymbol{p}, \boldsymbol{\pi})$, $\mu = 1, \ldots, m_{\bar{G}}$, are a collection of both point equality and inequality constraints; the explicit dependency from state and control variables is dropped as both are identified with their parameterised and discretised counterparts.

The parametric sensitivities $\partial \boldsymbol{p}^*(\boldsymbol{\pi}) / \partial \boldsymbol{\pi}$ of the optimal solution $\boldsymbol{p}^*(\boldsymbol{\pi})$ of Eq. (2.16) are obtained applying the sensitivity analysis for NLPs discussed in [Fiac 83]. Starting point is the *Lagrangian function*

$$\bar{L}(\boldsymbol{p}, \bar{\boldsymbol{\mu}}, \boldsymbol{\pi}) := \bar{\mathcal{J}}(\boldsymbol{p}, \boldsymbol{\pi}) + \bar{\boldsymbol{\mu}}^T \cdot \bar{G}(\boldsymbol{p}, \boldsymbol{\pi})$$

associated to the NLP Eq. (2.16). $\bar{\boldsymbol{\mu}} \in \mathbb{R}^{m_{\bar{G}}}$ are the *Lagrangian multipliers*. Now let $[\boldsymbol{p}_0^*, \bar{\boldsymbol{\mu}}_0^*]$ be the solution of the optimisation problem Eq. (2.16) for the nominal value $\boldsymbol{\pi} = \boldsymbol{\pi}_0$ of the disturbances. Further let $\bar{G}^\mathrm{a}$ be the vector of all active constraints $\bar{G}_\nu(\boldsymbol{p}, \boldsymbol{\pi}) = 0$, $\nu = 1, \ldots, m_{\bar{G}}$, and let $\bar{\boldsymbol{\mu}}^\mathrm{a}$ be the vector of associated multipliers. Then, under some conditions (including strong second order sufficient conditions for NLPs), cf., e.g., [Fiac 83], [BüMa 01a], the nominal solution can

be embedded into a $\mathcal{C}^1$ family of solutions $[\boldsymbol{p}^*(\boldsymbol{\pi}), \bar{\boldsymbol{\mu}}^*(\boldsymbol{\pi})]$, with the first order sensitivities given by

$$\begin{bmatrix} \frac{\partial \boldsymbol{p}^*}{\partial \boldsymbol{\pi}} \\ \frac{\partial \bar{\boldsymbol{\mu}}^{\mathrm{a}}}{\partial \boldsymbol{\pi}} \end{bmatrix}_{\boldsymbol{\pi}_0} = - \begin{bmatrix} \frac{\partial^2}{\partial \boldsymbol{p}^2} \bar{L} & \left[\frac{\partial}{\partial \boldsymbol{p}} \bar{G}^{\mathrm{a}} \right]^T \\ \frac{\partial}{\partial \boldsymbol{p}} \bar{G}^{\mathrm{a}} & \mathbf{0} \end{bmatrix}^{-1}_{\substack{\boldsymbol{p}_0^*, \bar{\boldsymbol{\mu}}_0^*, \\ \boldsymbol{\pi}_0}} \begin{bmatrix} \frac{\partial^2}{\partial \boldsymbol{p} \partial \boldsymbol{\pi}} \bar{L} \\ \frac{\partial}{\partial \boldsymbol{\pi}} \bar{G}^{\mathrm{a}} \end{bmatrix}_{\substack{\boldsymbol{p}_0^*, \bar{\boldsymbol{\mu}}_0^*, \\ \boldsymbol{\pi}_0}} . \tag{2.17}$$

Remark 2.11:
Eq. (2.17) is obtained by total differentiation of the KKT-conditions with respect to the disturbances $\boldsymbol{\pi}$ [BüMa 01a]. ◊

If Eq. (2.16) is solved by SQP methods it can be assumed that the constraint Jacobian $\partial \bar{G}(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{p}$ with respect to the optimisation parameters is provided by the user. The Jacobian of the constraints with respect to the disturbance parameters $\partial \bar{G}(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{\pi}$ can be obtained in the same way as the Jacobian with respect to the optimisation parameters. Additionally, the Hessian of the Lagrangian $\partial^2 \bar{L}(\boldsymbol{p}_0^*, \bar{\boldsymbol{\mu}}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{p}^2$ and $\partial^2 \bar{L}(\boldsymbol{p}_0^*, \bar{\boldsymbol{\mu}}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{p} \partial \boldsymbol{\pi}$ are required. Unfortunately, the approximate Hessian $\partial^2 \bar{L}/\partial \boldsymbol{p}^2$ generated by the SQP method cannot be used due to its lack of accuracy in case of BFGS updates [BüMa 01a]. Therefore, [BüMa 98] propose to compute the Hessian explicitly after the NLP has been solved, either by solution of an ODE [Büsk 98] or by approximation via finite differences [Büsk 99]. Similarly, $\partial^2 \bar{L}(\boldsymbol{p}_0^*, \bar{\boldsymbol{\mu}}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{p} \partial \boldsymbol{\pi}$ has to be obtained by finite difference approximations.

Now assume that the sensitivity matrices $\partial \bar{\boldsymbol{u}}(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{p}$, $\partial \bar{\boldsymbol{x}}_\nu(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{p}$, $\partial \bar{\mathcal{J}}(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{p}$, $\partial \bar{\boldsymbol{u}}(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{\pi}$, $\partial \bar{\boldsymbol{x}}_\nu(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{\pi}$, and $\partial \bar{\mathcal{J}}(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0)/\partial \boldsymbol{\pi}$ have been obtained, e.g., by sensitivity analysis of the (discretised) ODE-IVP. Furthermore, assume that the actual values $\boldsymbol{\pi}$ of the disturbances are known from direct measurement or from an estimation (cf. Section 2.3.2.d, dynamic data reconciliation). Then first order updates to the nominal parameterised open-loop optimal control are given by, e.g.,

$$\bar{\boldsymbol{u}}(t, \boldsymbol{p}^*(\boldsymbol{\pi}); \boldsymbol{\pi}) \doteq \bar{\boldsymbol{u}}(t, \boldsymbol{p}_0^*; \boldsymbol{\pi}_0) + \left[\frac{\partial \bar{\boldsymbol{u}}}{\partial \boldsymbol{p}} \cdot \frac{\partial \boldsymbol{p}^*}{\partial \boldsymbol{\pi}} + \frac{\partial \bar{\boldsymbol{u}}}{\partial \boldsymbol{\pi}} \right]_{\substack{\boldsymbol{p}_0^*, \\ \boldsymbol{\pi}_0}} (\boldsymbol{\pi} - \boldsymbol{\pi}_0) , \tag{2.18a}$$

$$\bar{\boldsymbol{x}}_\nu(\boldsymbol{p}^*(\boldsymbol{\pi}), \boldsymbol{\pi}) \doteq \bar{\boldsymbol{x}}_\nu(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0) + \left[\frac{\partial \bar{\boldsymbol{x}}_\nu}{\partial \boldsymbol{p}} \cdot \frac{\partial \boldsymbol{p}^*}{\partial \boldsymbol{\pi}} + \frac{\partial \bar{\boldsymbol{x}}_\nu}{\partial \boldsymbol{\pi}} \right]_{\substack{\boldsymbol{p}_0^*, \\ \boldsymbol{\pi}_0}} (\boldsymbol{\pi} - \boldsymbol{\pi}_0) , \tag{2.18b}$$

$$\bar{\mathcal{J}}(\boldsymbol{p}^*(\boldsymbol{\pi}), \boldsymbol{\pi}) \doteq \bar{\mathcal{J}}(\boldsymbol{p}_0^*, \boldsymbol{\pi}_0) + \left[\frac{\partial \bar{\mathcal{J}}}{\partial \boldsymbol{p}} \cdot \frac{\partial \boldsymbol{p}^*}{\partial \boldsymbol{\pi}} + \frac{\partial \bar{\mathcal{J}}}{\partial \boldsymbol{\pi}} \right]_{\substack{\boldsymbol{p}_0^*, \\ \boldsymbol{\pi}_0}} (\boldsymbol{\pi} - \boldsymbol{\pi}_0) , \text{ and} \tag{2.18c}$$

$$\boldsymbol{p}^*(\boldsymbol{\pi}) \doteq \boldsymbol{p}_0^* + \left. \frac{\partial \boldsymbol{p}^*}{\partial \boldsymbol{\pi}} \right|_{\boldsymbol{\pi}_0} (\boldsymbol{\pi} - \boldsymbol{\pi}_0) . \tag{2.18d}$$

2.6 A Linearisation of the Direct Shooting NLP

2.6.1 The Disturbed Optimal Control Problem

At first we extend our basic optimal control problem specified in Definition 2.1 so as to incorporate *a priori* unknown disturbances which are characteristic in online application. In order to avoid excessive notation we reuse the symbols already introduced in the previous sections.

Definition 2.4 (Disturbed Optimal Control Problem)
Let there be a dynamical process described by a DAE including a parameterised disturbance model

$$\dot{\boldsymbol{x}}(t;\boldsymbol{\pi}) = \boldsymbol{f}(t,\boldsymbol{x}(t;\boldsymbol{\pi}),\boldsymbol{y}(t;\boldsymbol{\pi}),\boldsymbol{u}(t),\operatorname{sign}\boldsymbol{q}(t,\boldsymbol{x}(t;\boldsymbol{\pi}),\boldsymbol{y}(t;\boldsymbol{\pi}),\boldsymbol{u}(t),\boldsymbol{\pi}),\boldsymbol{\pi})\,, \quad (2.19a)$$

$$0 = \boldsymbol{g}(t,\boldsymbol{x}(t;\boldsymbol{\pi}),\boldsymbol{y}(t;\boldsymbol{\pi}),\boldsymbol{u}(t),\operatorname{sign}\boldsymbol{q}(t,\boldsymbol{x}(t;\boldsymbol{\pi}),\boldsymbol{y}(t;\boldsymbol{\pi}),\boldsymbol{u}(t),\boldsymbol{\pi}),\boldsymbol{\pi})\,, \quad (2.19b)$$

where $\boldsymbol{f} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}+m_{\boldsymbol{q}}+n_{\boldsymbol{\pi}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$ *and* $\boldsymbol{g} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}+m_{\boldsymbol{q}}+n_{\boldsymbol{\pi}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$ *together with the switching functions* $\boldsymbol{q} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}+n_{\boldsymbol{\pi}}} \to \mathbb{R}^{m_{\boldsymbol{q}}}$ *describe the possibly discontinuous dynamics.* $\boldsymbol{\pi} \in n_{\boldsymbol{\pi}}$ *is the vector of disturbance parameters. Standard model parameters are included in the vector of external controls* $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^{n_{\boldsymbol{u}}}$ *as the special constant case.*

Given a fixed *vector of disturbance parameters* $\boldsymbol{\pi}$*, an interval in time* $[t_{0,j}, t_{f,j}] \subset \mathbb{R}$*, an objective function* $\mathcal{J} : \mathcal{C}_p^0 \to \mathbb{R}$ *in Mayer form (represented by* $E : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}$*), path inequality constraints* $\boldsymbol{c} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}+n_{\boldsymbol{\pi}}} \to \mathbb{R}^{m_{\boldsymbol{c}}}$*, point inequality constraints* $\boldsymbol{k}_\mu^{ic} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{u}}+n_{\boldsymbol{\pi}}} \to \mathbb{R}^{m_{\boldsymbol{k}_\mu^{ic}}}$*,* $\mu = 1,\ldots,m_{\boldsymbol{k}^{ic}}$*, and initial conditions* $\boldsymbol{k}^{ini} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}+n_{\boldsymbol{u}}+n_{\boldsymbol{\pi}}} \to \mathbb{R}^{m_{\boldsymbol{k}^{ini}}}$ *the* disturbed optimal control problem *is to find optimal controls* $\boldsymbol{u}_{(\boldsymbol{\pi})} \in \mathcal{C}_p^0(\mathbb{R},\mathbb{R}^{n_{\boldsymbol{u}}})$ *such that*

$$\mathcal{J}[\boldsymbol{u}_{(\boldsymbol{\pi})}] = E(t_{f,j},\boldsymbol{x}(t_{f,j};\boldsymbol{\pi}),\boldsymbol{y}(t_{f,j};\boldsymbol{\pi})) \longrightarrow \min_{\boldsymbol{u}_{(\boldsymbol{\pi})}}! \quad (2.20)$$

subject to

$$\begin{aligned}
\dot{\boldsymbol{x}}(t;\boldsymbol{\pi}) &= \boldsymbol{f}(t,\boldsymbol{x}(t;\boldsymbol{\pi}),\boldsymbol{y}(t;\boldsymbol{\pi}),\boldsymbol{u}_{(\boldsymbol{\pi})}(t),\operatorname{sign}\boldsymbol{q},\boldsymbol{\pi})\,;\ t \in [t_{0,j},t_{f,j}]\,,\\
0 &= \boldsymbol{g}(t,\boldsymbol{x}(t;\boldsymbol{\pi}),\boldsymbol{y}(t;\boldsymbol{\pi}),\boldsymbol{u}_{(\boldsymbol{\pi})}(t),\operatorname{sign}\boldsymbol{q},\boldsymbol{\pi})\,,\\
0 &\geq \boldsymbol{c}(t,\boldsymbol{x}(t;\boldsymbol{\pi}),\boldsymbol{y}(t;\boldsymbol{\pi}),\boldsymbol{u}_{(\boldsymbol{\pi})}(t),\boldsymbol{\pi})\,,\\
0 &\geq \boldsymbol{k}_\mu^{ic}(t_\mu,\boldsymbol{x}(t_\mu;\boldsymbol{\pi}),\boldsymbol{y}(t_\mu;\boldsymbol{\pi}),\boldsymbol{u}_{(\boldsymbol{\pi})}(t_\mu),\boldsymbol{\pi})\,;\ \mu = 1,\ldots,m_{\boldsymbol{k}^{ic}}\,,\\
0 &= \boldsymbol{k}^{ini}(t_{0,j},\boldsymbol{x}(t_{0,j};\boldsymbol{\pi}),\boldsymbol{y}(t_{0,j};\boldsymbol{\pi}),\dot{\boldsymbol{x}}(t_{0,j};\boldsymbol{\pi}),\boldsymbol{u}_{(\boldsymbol{\pi})}(t_{0,j}),\boldsymbol{\pi})\,.
\end{aligned}$$

Similar to Section 2.4.2 we employ the direct single shooting technique in order to discretise the infinite dimensional optimal control problem Eq. (2.20). The

resulting NLP reads as

$$\widehat{\mathcal{J}}[\boldsymbol{p}] = \widehat{E}(t_{\mathrm{f},j}, \widehat{\boldsymbol{x}}(t_{\mathrm{f},j}; \boldsymbol{p}, \boldsymbol{\pi}), \widehat{\boldsymbol{y}}(t_{\mathrm{f},j}; \boldsymbol{p}, \boldsymbol{\pi})) \longrightarrow \min_{\boldsymbol{p}}! \tag{2.21a}$$

subject to

$$\begin{aligned} 0 &\geq \widehat{\boldsymbol{c}}(\tau_j^{\mu,\boldsymbol{c}}, \widehat{\boldsymbol{x}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}, \boldsymbol{\pi}), \widehat{\boldsymbol{y}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}, \boldsymbol{\pi}), \boldsymbol{p}, \boldsymbol{\pi})\,;\ \mu = 1, \ldots, M_j\,, \\ 0 &\geq \widehat{\boldsymbol{k}}_{\mu}^{\mathrm{ic}}(t_{\mu}, \widehat{\boldsymbol{x}}(t_{\mu}; \boldsymbol{p}, \boldsymbol{\pi}), \widehat{\boldsymbol{y}}(t_{\mu}; \boldsymbol{p}, \boldsymbol{\pi}), \boldsymbol{p}, \boldsymbol{\pi})\,;\ \mu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ic}}}\,, \end{aligned} \tag{2.21b}$$

and

$$\begin{aligned} \dot{\widehat{\boldsymbol{x}}}(t; \boldsymbol{p}, \boldsymbol{\pi}) &= \widehat{\boldsymbol{f}}(t, \widehat{\boldsymbol{x}}(t; \boldsymbol{p}, \boldsymbol{\pi}), \widehat{\boldsymbol{y}}(t; \boldsymbol{p}, \boldsymbol{\pi}), \boldsymbol{p}, \operatorname{sign} \widehat{\boldsymbol{q}}, \boldsymbol{\pi})\,;\ t \in [t_{0,j}, t_{\mathrm{f},j}]\,, \\ 0 &= \widehat{\boldsymbol{g}}(t, \widehat{\boldsymbol{x}}(t; \boldsymbol{p}, \boldsymbol{\pi}), \widehat{\boldsymbol{y}}(t; \boldsymbol{p}, \boldsymbol{\pi}), \boldsymbol{p}, \operatorname{sign} \widehat{\boldsymbol{q}}, \boldsymbol{\pi})\,, \\ 0 &= \widehat{\boldsymbol{k}}^{\mathrm{ini}}(t_{0,j}, \widehat{\boldsymbol{x}}(t_{0,j}; \boldsymbol{p}, \boldsymbol{\pi}), \widehat{\boldsymbol{y}}(t_{0,j}; \boldsymbol{p}, \boldsymbol{\pi}), \dot{\widehat{\boldsymbol{x}}}(t_{0,j}; \boldsymbol{p}, \boldsymbol{\pi}), \boldsymbol{p}, \boldsymbol{\pi})\,. \end{aligned} \tag{2.21c}$$

2.6.2 A New Correction Method

Similar to the optimisation problem Eq. (2.16) the solution of the discretised optimal control problem Eqs. (2.21a)–(2.21c) depends on the disturbances, i.e., $\boldsymbol{p}^* = \boldsymbol{p}^*(\boldsymbol{\pi})$.

Assuming that the mapping $\boldsymbol{\pi} \mapsto \boldsymbol{p}^*(\boldsymbol{\pi})$ is differentiable the method described in Section 2.5.3 can be applied to Eqs. (2.21a)–(2.21c). Once $\partial \boldsymbol{p}^*(\boldsymbol{\pi})/\partial \boldsymbol{\pi}$ has been generated corrected optimisation parameters are obtained according to Eq. (2.18d) in first order approximation. Unfortunately, computation of a part of the derivative information required for the generation of the sensitivity information $\partial \boldsymbol{p}^*(\boldsymbol{\pi})/\partial \boldsymbol{\pi}$ based on Eq. (2.17) is costly for large-scale problems, cf. Section 2.5.3. Therefore, we propose a new optimisation based disturbance rejection technique which takes into account the requirements of our special problem setting.

Remark 2.12:
In Section 2.5.1 – Section 2.5.3 we have presented differentiability conditions for the mapping $\boldsymbol{\pi} \mapsto \boldsymbol{p}^*(\boldsymbol{\pi})$ and methods for the computation of the first order derivatives $\partial \boldsymbol{p}^*(\boldsymbol{\pi})/\partial \boldsymbol{\pi}$. However, the theoretical results have been obtained for optimal control problems with ODE models and not for higher index DAEs as in our case. Thus these theoretical results can only be applied with care. ◊

Our main idea is that the predominant goal in the load-change problem consists in the fulfilment of the (nonlinear) constraints, e.g., in order to ensure a safe load-change for an air separation plant. Therefore, especially the constraints $\widehat{\boldsymbol{c}}$ and $\widehat{\boldsymbol{k}}^{\mathrm{ic}}$ in the disturbed parameterised optimal control problem Eqs. (2.21a)–(2.21c) have to be considered. Optimality with respect to the objective function is advantageous, but it is not as strictly required.

Assume that the optimal solution $\boldsymbol{p}_0^* = \boldsymbol{p}^*(\boldsymbol{\pi}_0)$ of Eqs. (2.21a)–(2.21c) corresponding to the nominal disturbance $\boldsymbol{\pi} = \boldsymbol{\pi}_0$ is known. If a small deviation

$\Delta\boldsymbol{\pi} \in \mathbb{R}^{n_{\boldsymbol{\pi}}}$ from the nominal case is introduced, i.e., $\boldsymbol{\pi} = \boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}$, then in first order approximation the corrected optimisation parameters are given by

$$\boldsymbol{p}^*(\boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}) \doteq \boldsymbol{p}_0^* + \Delta\boldsymbol{p}\,, \tag{2.22}$$

where

$$\Delta\boldsymbol{p} := \frac{\partial \boldsymbol{p}^*(\boldsymbol{\pi}_0)}{\partial \boldsymbol{\pi}} \Delta\boldsymbol{\pi}\,,\ \Delta\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}\,.$$

Similarly, a linearised estimate for the values of the constraints Eq. (2.21b) subject to the same disturbance is given by

$$\begin{aligned}
&\widehat{\boldsymbol{c}}(\tau_j^{\mu,\boldsymbol{c}}, \widehat{\boldsymbol{x}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}^*(\boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}), \boldsymbol{\pi} + \Delta\boldsymbol{\pi}),\\
&\qquad \widehat{\boldsymbol{y}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}^*(\boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}), \boldsymbol{\pi} + \Delta\boldsymbol{\pi}), \boldsymbol{p}^*(\boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}), \boldsymbol{\pi} + \Delta\boldsymbol{\pi}) \doteq \left[\widehat{\boldsymbol{c}}(\cdot)\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}}\\
&+ \left[\frac{\partial \widehat{\boldsymbol{c}}(\cdot)}{\partial \boldsymbol{x}}\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \left[\frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} + \left[\frac{\partial \widehat{\boldsymbol{c}}(\cdot)}{\partial \boldsymbol{y}}\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \left[\frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}}\\
&\qquad + \left[\frac{\partial \widehat{\boldsymbol{c}}(\cdot)}{\partial \boldsymbol{p}}\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{p} + \left[\frac{\partial \widehat{\boldsymbol{c}}(\cdot)}{\partial \boldsymbol{\pi}}\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{\pi}\,;\ \mu = 1, \ldots, M_j\,,
\end{aligned} \tag{2.23a}$$

and

$$\begin{aligned}
&\widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(t_\mu, \widehat{\boldsymbol{x}}(t_\mu; \boldsymbol{p}^*(\boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}), \boldsymbol{\pi} + \Delta\boldsymbol{\pi}),\\
&\qquad \widehat{\boldsymbol{y}}(t_\mu; \boldsymbol{p}^*(\boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}), \boldsymbol{\pi} + \Delta\boldsymbol{\pi}), \boldsymbol{p}^*(\boldsymbol{\pi}_0 + \Delta\boldsymbol{\pi}), \boldsymbol{\pi} + \Delta\boldsymbol{\pi}) \doteq \left[\widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(\cdot)\right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}}\\
&+ \left[\frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(\cdot)}{\partial \boldsymbol{x}}\right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \left[\frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} + \left[\frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(\cdot)}{\partial \boldsymbol{y}}\right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \left[\frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}}\\
&\qquad + \left[\frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(\cdot)}{\partial \boldsymbol{p}}\right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{p} + \left[\frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(\cdot)}{\partial \boldsymbol{\pi}}\right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{\pi}\,;\ \mu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ic}}}\,.
\end{aligned} \tag{2.23b}$$

Now *assume* that the optimiser $\boldsymbol{p}^*$ is *independent* from the disturbances $\boldsymbol{\pi}$. Then the right hand sides of Eqs. (2.23a)–(2.23b) can be interpreted as the linearisation of the constraints Eq. (2.21b) around $[(\boldsymbol{p}_0^*)^T, \boldsymbol{\pi}_0^T]^T$ in the direction of the sufficiently small but otherwise *arbitrary* vector $[\Delta\boldsymbol{p}^T, \Delta\boldsymbol{\pi}^T]^T \in \mathbb{R}^{n_{\boldsymbol{p}}+n_{\boldsymbol{\pi}}}$, i.e.,

$$\begin{aligned}
&\left[\widehat{\boldsymbol{c}} + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{x}}\left[\frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right]\right.\\
&\qquad \left. + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{y}}\left[\frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right] + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{p}}\Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \doteq
\end{aligned}$$

$$\widehat{\boldsymbol{c}}(\tau_j^{\mu,\boldsymbol{c}}, \widehat{\boldsymbol{x}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}_0^* + \Delta\boldsymbol{p}, \boldsymbol{\pi} + \Delta\boldsymbol{\pi}), \widehat{\boldsymbol{y}}(\tau_j^{\mu,\boldsymbol{c}}; \boldsymbol{p}_0^* + \Delta\boldsymbol{p}, \boldsymbol{\pi} + \Delta\boldsymbol{\pi}), \boldsymbol{p}_0^* + \Delta\boldsymbol{p}, \boldsymbol{\pi} + \Delta\boldsymbol{\pi}) \,; \quad \mu = 1, \ldots, M_j \,, \tag{2.24a}$$

and

$$\begin{aligned} &\left[\widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}} + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{x}} \left[\frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{\pi}} \Delta\boldsymbol{\pi} \right] \right. \\ &\qquad \left. + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{y}} \left[\frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{\pi}} \Delta\boldsymbol{\pi} \right] + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{\pi}} \Delta\boldsymbol{\pi} \right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \doteq \\ &\widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}(t_\mu, \widehat{\boldsymbol{x}}(t_\mu; \boldsymbol{p}_0^* + \Delta\boldsymbol{p}, \boldsymbol{\pi} + \Delta\boldsymbol{\pi}), \widehat{\boldsymbol{y}}(t_\mu; \boldsymbol{p}_0^* + \Delta\boldsymbol{p}, \boldsymbol{\pi} + \Delta\boldsymbol{\pi}), \boldsymbol{p}_0^* + \Delta\boldsymbol{p}, \boldsymbol{\pi} + \Delta\boldsymbol{\pi}) \,; \\ &\qquad \mu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ic}}} \,. \end{aligned} \tag{2.24b}$$

Since our primary interest is in the feasibility of the constraints of the disturbed NLP Eqs. (2.21a)–(2.21c) we disregard the original objective Eq. (2.21a) and state the new optimisation problem

$$\widehat{\mathcal{J}}[\Delta\boldsymbol{p}] := \sum_{\nu=1}^{n_{\boldsymbol{p}}} \left(\frac{\Delta\boldsymbol{p}_\nu}{\boldsymbol{w}_\nu^{\Delta\boldsymbol{p}}} \right)^2 \longrightarrow \min_{\Delta\boldsymbol{p}}! \tag{2.25a}$$

subject to

$$\begin{aligned} &-\widehat{\boldsymbol{c}}\Big|_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} - \left[\frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{x}} \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{\pi}} + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{y}} \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{\pi}} + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{\pi}} \right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{\pi} \geq \\ &\qquad \left[\frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{x}} \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{p}} + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{y}} \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{p}} + \frac{\partial \widehat{\boldsymbol{c}}}{\partial \boldsymbol{p}} \right]_{\substack{\tau_j^{\mu,\boldsymbol{c}} \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{p} \,; \; \mu = 1, \ldots, M_j \,, \end{aligned} \tag{2.25b}$$

$$\begin{aligned} &-\widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}\Big|_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} - \left[\frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{x}} \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{\pi}} + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{y}} \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{\pi}} + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{\pi}} \right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{\pi} \geq \\ &\qquad \left[\frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{x}} \frac{\partial \widehat{\boldsymbol{x}}}{\partial \boldsymbol{p}} + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{y}} \frac{\partial \widehat{\boldsymbol{y}}}{\partial \boldsymbol{p}} + \frac{\partial \widehat{\boldsymbol{k}}_\mu^{\mathrm{ic}}}{\partial \boldsymbol{p}} \right]_{\substack{t_\mu \\ \boldsymbol{p}_0^*, \boldsymbol{\pi}_0}} \cdot \Delta\boldsymbol{p} \,; \; \mu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ic}}} \,. \end{aligned} \tag{2.25c}$$

where $\boldsymbol{w}_\nu^{\Delta\boldsymbol{p}} \in \mathbb{R}_+ \backslash \{0\}$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$, are weighting factors, e.g., $\boldsymbol{w}_\nu^{\Delta\boldsymbol{p}} := 1 + \left| \boldsymbol{p}_{0,\nu}^* \right|$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$. As above (cf. Eq. (2.22)) the corrected parameters are given by $\boldsymbol{p}_0^* + \Delta\boldsymbol{p}$. The optimisation problem formulated in Eqs. (2.25a)–(2.25c) is a convex linearly constrained *quadratic programming problem* (QP) of moderate size which can be solved very efficiently. Objective and constraint evaluations

reduce to simple multiplication and addition of precalculated information. Here, precalculated means that the parametric sensitivity information with respect to $\boldsymbol{p}_0^*$ is already computed during the synthesis of the nominal open-loop optimal control, and that the parametric sensitivities of the nominal optimal control with respect to $\boldsymbol{\pi}_0$ can be obtained by straightforward sensitivity analysis, e.g., immediately after computation of the nominal trajectory. Thus the solution of this QP provides a real-time capable short-cut method for the generation of updates $\Delta\boldsymbol{p}$ to an open-loop optimal set of parameters $\boldsymbol{p}_0^*$ with the aim to preserve feasibility of the constraints. The progress of time in the real-time application has to be accounted for by restriction of the linearised point constraints in Eqs. (2.25b)–(2.25c) to those which have not been passed at the present time t^{actual}, i.e., to $\tau^{\mu,\boldsymbol{c}} \geq t^{\text{actual}}$, $\mu = 1, \ldots, M_j$, and t_μ, $\mu = 1, \ldots, m_{\boldsymbol{k}^{\text{ic}}}$, respectively.

The proposed update algorithm is no longer based on the theory of neighbouring extremals as it drops the dependency of the optimal set of optimisation parameters from the disturbances present in the original NLP Eqs. (2.21a)–(2.21c). The price to be paid for this simplification is that the adapted set of optimisation parameters obtained via Eqs. (2.25a)–(2.25c) in general looses its optimality properties with respect to Eqs. (2.21a)–(2.21c). On the other hand, the proposed method is computationally feasible even for larger problems in an MPC framework as the computation of the sensitivities $\partial\boldsymbol{p}(\boldsymbol{\pi})/\partial\boldsymbol{\pi}$ is not required. Additionally, the problem of establishing differentiability for the mapping $\boldsymbol{\pi} \mapsto \boldsymbol{p}^*(\boldsymbol{\pi})$ induced by Eqs. (2.21a)–(2.21c) is avoided.

By construction a controller based on Eqs. (2.25a)–(2.25c) is passive unless a violation of the (linearised) constraints is predicted. However, trajectory tracking control behaviour may be incorporated by using the augmented objective

$$\begin{aligned}\widehat{\mathcal{J}}[\Delta\boldsymbol{p}] := &\sum_{\nu=1}^{n_{\boldsymbol{p}}} \left(\frac{\Delta\boldsymbol{p}_\nu}{\boldsymbol{w}_\nu^{\Delta\boldsymbol{p}}}\right)^2 \\ &+ \sum_{\substack{\nu=1,\ldots,n_{\text{mmt}},\\ t_\nu^{\text{mmt}}\in[t_{0,j},t_{\text{f},j}]}} \sum_{\mu=1}^{n_{\boldsymbol{z}}} \frac{1}{\boldsymbol{w}_\mu} \left(\widehat{\boldsymbol{z}}_{\mu,(\nu)}^{\text{mmt}} - \left[\widehat{\boldsymbol{z}}_\mu + \frac{\partial\widehat{\boldsymbol{z}}_\mu}{\partial\boldsymbol{\pi}}\Delta\boldsymbol{\pi}\right]_{\substack{t_\nu^{\text{mmt}}\\ \boldsymbol{p}_0^*,\boldsymbol{\pi}_0}} - \left[\frac{\partial\widehat{\boldsymbol{z}}_\mu}{\partial\boldsymbol{p}}\Delta\boldsymbol{p}\right]_{\substack{t_\nu^{\text{mmt}}\\ \boldsymbol{p}_0^*,\boldsymbol{\pi}_0}}\right)^2\end{aligned} \tag{2.26}$$

instead of Eq. (2.25a). With this modified formulation the controller tracks the nominal trajectory for a subset of the state variables $\widehat{\boldsymbol{z}} \subset \{\widehat{\boldsymbol{x}}_1, \ldots, \widehat{\boldsymbol{x}}_{n_{\boldsymbol{x}}}, \widehat{\boldsymbol{y}}_1, \ldots, \widehat{\boldsymbol{y}}_{n_{\boldsymbol{y}}}\}$, $\widehat{\boldsymbol{z}} \in \mathbb{R}^{n_{\boldsymbol{z}}}$, given by the values $\widehat{\boldsymbol{z}}_{(\nu)}^{\text{mmt}}$ at the sample times $t_\nu^{\text{mmt}} \in [t_{0,j}, t_{\text{f},j}]$, $\nu = 1, \ldots, n_{\text{mmt}}$. The values $\widehat{\boldsymbol{z}}_{(\nu)}^{\text{mmt}}$ are obtained by *simulation* during the computation of the nominal trajectory. $\boldsymbol{w}_\nu \in \mathbb{R}_+\backslash\{0\}$, $\nu = 1, \ldots, n_{\boldsymbol{z}}$, are weighting factors. The advantage of this modification is that less deviation from the open-loop optimal solution improves the linear prediction based on the first order sensitivity information.

The objective Eq. (2.26) has the same structure as the objective function used,

e.g., in QDMC, cf. [MoLe 91]. From the LMPC point of view the penalty on the optimisation parameters is equivalent to penalising the control moves of the new controller. Thus in effect this penalty improves the robustness of the MPC-type control scheme [BiRa 91], [MeRa 97].

We emphasise that the algorithm introduced in this section is not intended as a stand-alone controller but is designed as a part of our overall control concept. The controller relies on data computed during and immediately after the solution of the full open-loop optimal control problem. Especially, it requires the reference trajectory and the parametric sensitivity information for this reference trajectory. Moreover, the algorithm is restricted to a sufficiently small neighbourhood of the nominal solution due to the inherent limitations of the linear approximations employed.

2.6.3 Comparison with Related Work

Linearisation approaches to NMPC have already been mentioned in Section 2.3.2.c. Either they are employed in order to extend LMPC methods to the nonlinear case, or they are used in order to simplify the NMPC task such that the online problem becomes numerically tractable, e.g., in order to avoid the necessity for a fast solution of a difficult NLP under consideration of nonlinear system dynamics. The extended Newton-type controller summarised in [LBEM 90], [BiRa 91] is one of these linearisation based approaches. In the course of our investigations we found that the technique introduced in Section 2.6.2 above is related to this controller.

Originally, the extended Newton-type control approach of [BiRa 91] is based on the Newton-type control law which has been proposed by [EMP 86] in order to extend the linear *internal model control* (*IMC*) method to the nonlinear case. [LiBi 88] formulate a QP based on this control law which allows them to incorporate state and control constraints into the controller. Starting point is the first order approximation of the system outputs around a nominal trajectory using sensitivity information. The QP then proposes a correction to the nominal controls. Additionally, a line-search is included which on the one hand guarantees convergence to the setpoint of the controller and on the other hand often extends its stability region [BiRa 91]. In [LiBi 89] this one-step algorithm (i.e., both prediction and control horizon cover one time step) is extended to an MPC-like multistep algorithm. Finally, disturbances are addressed in [LiBi 90]. Several stability properties of this method can be shown.

The standard problem considered is the regulator task with state and control constraints. It can be denoted as [LiBi 89]

$$\mathcal{J}[\boldsymbol{u}] = \int_{t_0}^{t_f} \left\| \boldsymbol{y}(t) - \boldsymbol{y}_{sp} \right\|^2 \mathrm{d}t \longrightarrow \min_{\boldsymbol{u}}! \tag{2.27}$$

subject to $(\forall t \in [t_0, t_\mathrm{f}])$

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))\,, \\
\boldsymbol{y}(t) &= \boldsymbol{g}(\boldsymbol{x}(t))\,, \\
\boldsymbol{y}^{\mathrm{lwr}} \leq \boldsymbol{y}(t) &\leq \boldsymbol{y}^{\mathrm{upr}}\,, \\
\boldsymbol{u}^{\mathrm{lwr}} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}^{\mathrm{upr}}\,, \dot{\boldsymbol{u}}^{\mathrm{lwr}} &\leq \dot{\boldsymbol{u}}(t) \leq \dot{\boldsymbol{u}}^{\mathrm{upr}}\,, \\
\boldsymbol{x}(t_0) &= \boldsymbol{x}_0\,,
\end{aligned}$$

where in standard control theory notation $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$ are the system states, $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$ are the system outputs, and $\boldsymbol{u} \in \mathbb{R}^{n_{\boldsymbol{u}}}$ are the system inputs. $\boldsymbol{f} : \mathbb{R}^{n_{\boldsymbol{x}}+n_{\boldsymbol{u}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$ describes the system dynamics while $\boldsymbol{g} : \mathbb{R}^{n_{\boldsymbol{x}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$ is the system output map. Simple lower and upper bounds for the system outputs, for the controls, as well as for the control changes are specified by $\boldsymbol{y}^{\mathrm{lwr}}, \boldsymbol{y}^{\mathrm{upr}} \in \mathbb{R}^{n_{\boldsymbol{y}}}$, and $\boldsymbol{u}^{\mathrm{lwr}}, \boldsymbol{u}^{\mathrm{upr}}, \dot{\boldsymbol{u}}^{\mathrm{lwr}}, \dot{\boldsymbol{u}}^{\mathrm{upr}} \in \mathbb{R}^{n_{\boldsymbol{u}}}$. The initial system state $\boldsymbol{x}_0 \in \mathbb{R}^{n_{\boldsymbol{x}}}$ as well as the constant setpoint $\boldsymbol{y}_{\mathrm{sp}} \in \mathbb{R}^{n_{\boldsymbol{y}}}$ have to be given. The system inputs are restricted to the class of piecewise constant controls with a fixed sampling interval.

The control task is solved within a moving horizon framework; however, in order to ease notation we drop indices for the respective moving horizon step. Assume that a nominal control trajectory $\bar{\boldsymbol{u}}$ and a corresponding nominal state output trajectory $\bar{\boldsymbol{y}}$ for the current prediction horizon are given. Then the task is to obtain corrections $\Delta\boldsymbol{u}$ to the nominal control, i.e., the actual control is then given by $\boldsymbol{u} = \bar{\boldsymbol{u}} + \Delta\boldsymbol{u}$. As the piecewise constant controls are parameterised by their values at the beginning of the respective sampling intervals parametric sensitivities of the system outputs with respect to the controls can be defined straightforwardly. Based on these sensitivities [LiBi 89] set up the QP

$$\widehat{\mathcal{J}}[\Delta\boldsymbol{u}] = \sum_{n=1}^{\mu} \left\| \widehat{\boldsymbol{y}}^{n+1} - \boldsymbol{y}_{\mathrm{sp}} \right\|^2 \longrightarrow \min_{\Delta\boldsymbol{u}}! \tag{2.28}$$

subject to $(n = 1, \ldots, \mu)$

$$\begin{aligned}
\widehat{\boldsymbol{y}}^{n+1} &= \bar{\boldsymbol{y}}^{n+1} + \sum_{\nu=1}^{\mu} \boldsymbol{\sigma}_\nu^{n+1} \Delta\boldsymbol{u}^\nu\,, \\
\boldsymbol{y}^{\mathrm{lwr}} \leq \widehat{\boldsymbol{y}}^{n+1} &\leq \boldsymbol{y}^{\mathrm{upr}}\,, \\
\boldsymbol{u}^{\mathrm{lwr}} \leq \boldsymbol{u}^n \leq \boldsymbol{u}^{\mathrm{upr}}\,, \dot{\boldsymbol{u}}^{\mathrm{lwr}} &\leq \Delta\boldsymbol{u}^n \leq \dot{\boldsymbol{u}}^{\mathrm{upr}}\,,
\end{aligned}$$

corresponding to the optimal control problem Eq. (2.27) in *discrete time formulation* with a fixed sampling interval (e.g., $\widehat{\boldsymbol{y}}^\nu = \widehat{\boldsymbol{y}}(t_\nu)$, $t_\nu = t_0 + h\nu$, $\nu = 0, 1, 2, \ldots, \mu$). The matrices $\boldsymbol{\sigma}_\nu^{n+1}$, $\nu = 1, \ldots, \mu$, $n = 1, \ldots, \mu$, contain the sensitivity information for the system outputs with respect to the controls. $\widehat{\boldsymbol{y}}^{n+1}$, $n = 1, \ldots, \mu$, is then the linearised approximation to the system outputs. $\mu \in \mathbb{N}$ denotes the number of

sampling intervals used as prediction horizon. For a more detailed explanation of the notation we refer to, e.g., [LiBi 89]. The QP Eq. (2.28) is one of the central parts of Algorithm 3, cf., e.g., [LiBi 89].

Algorithm 3 (Multistep, Newton-Type Controller)

1. *Initialise.*
2. *Set the control output of the previous control move as the nominal control for the entire prediction horizon. Compute the corresponding nominal system output as well as the sensitivities.*
3. (a) *Solve the QP Eq.* (2.28) *to obtain a (Newton-)correction* $\Delta \boldsymbol{u}$.
   (b) *Perform a line-search to obtain a (Newton-)step-size* $\lambda \in]0,1]$ *such that* $\sum_{n=1}^{\mu} \left\| \widehat{\boldsymbol{y}}^{n+1} - \boldsymbol{y}_{sp} \right\|^2$ *decreases sufficiently.*
4. *Update the nominal controls* $\bar{\boldsymbol{u}} + \lambda \Delta \boldsymbol{u} \rightarrow \bar{\boldsymbol{u}}$.

   If the difference to the previous iteration is below a given threshold employ $\bar{\boldsymbol{u}} \rightarrow \boldsymbol{u}$ *as control and goto step* **5**

   Otherwise, relinearise (i.e., obtain new sensitivities) and go back to step **3**.
5. *Implement the control on the control horizon, i.e., the first sampling interval considered, advance the horizons and restart with step* **1**.

Our technique developed in Section 2.6.2 and Algorithm 3 correspond in employing a first order approximation to the system dynamics based on sensitivity functions. The main difference is in the generation of the next control move. Here, our proposal does not contain the Newton-type iteration implemented in the steps **3** and **4** of Algorithm 3 in order to obtain the final controls. However, as pointed out in [LiBi 89], [BiRa 91] this iteration need not be made until full convergence; even a single step including the line-search can be sufficient. In this case the main algorithmic difference in the two approaches reduces to the line-search. Another difference is our approximation to the influence of disturbances based on the parametric sensitivities of the states with respect to the disturbance parameters.

We consider the similarities between both techniques an interesting point as these two approaches have evolved starting from rather different origins. On the one hand, the extended Newton-type approach [BiRa 91] was intended as an extension to an optimisation-free linear control law. On the other hand, our method discussed in Section 2.6.2 has been originally derived from the neighbouring extremals technique as a second level short-cut alternative to full NMPC.

Chapter 3

Consistent Initial Conditions

Hearing much and selecting what is good and following it;
seeing much and keeping it in memory:
– this is the second style of knowledge.

Confucius: Analects (7,27)

3.1 Review of Previous Work

A considerable amount of research concerning DAEs has been spent in the development of algorithms for computing consistent initial conditions. Some of this research will be discussed in this section.

One of our intentions connected with the broad review provided in this section is to point out that in general the calculation of consistent initial conditions is a demanding task, especially for higher index DAEs (index ≥ 2) of larger scale. According to [Marq 91] no satisfactory answer to the consistent initialisation problem was available in the early 1990's. Thus our second intention is to demonstrate that although one decade of research has passed since the statement of [Marq 91] a final and fully general method for the solution of the consistent initialisation problem is still pending. Finally, we employ some of the ideas collected in this section for the development of our own tailored algorithm in the subsequent Section 3.2.

3.1.1 Solution of the Consistency Equations

[CaMo 94], [CKY 96], [CMZ 96] develop a method for the integration of a broad class of DAEs Eq. (1.1a) of finite, but otherwise (in principle) arbitrary index based on the integration of the derivative array equations Eq. (1.5) by multistep methods. The numerical computation of consistent initial conditions is addressed especially in [CKY 96]. The DAE is assumed to be available in symbolic form, which allows the construction of the derivative array equations by formula manipulation

programs such as MAPLE™. A further possibility mentioned in [CaMo 94] is the application of automatic differentiation programs such as ADOL-C [GJU 96].

Given the index ι_d of a DAE $\boldsymbol{F}$ (Eq. (1.1a)) consistent initial conditions are calculated according to Definition 1.12 and Definition 1.13 as a solution of the consistency equations

$$\begin{aligned}
\boldsymbol{F}_{[0]}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t))\Big|_{t=t_0} &= 0\,, \\
\boldsymbol{F}_{[1]}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t), \ddot{\boldsymbol{\xi}}(t))\Big|_{t=t_0} &= 0\,, \\
&\vdots \\
\boldsymbol{F}_{[\iota_\mathrm{d}]}(t, \boldsymbol{\xi}^{(0)}(t), \boldsymbol{\xi}^{(1)}(t), \ldots, \boldsymbol{\xi}^{(\iota_\mathrm{d}+1)}(t))\Big|_{t=t_0} &= 0\,, \\
\boldsymbol{k}^{\mathrm{ini}}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t))\Big|_{t=t_0} &= 0\,,
\end{aligned} \tag{3.1}$$

where $\boldsymbol{F}_{[\mu]}(\cdots) := d^\mu \boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t))/dt^\mu$ and $\boldsymbol{\xi}^{(\mu)}(t) := d^\mu \boldsymbol{\xi}(t)/dt^\mu$. The values $\boldsymbol{\xi}^{(\mu)} := \boldsymbol{\xi}^{(\mu)}(t)\big|_{t=t_0}$, $\mu = 0, \ldots, \iota_\mathrm{d}+1$, are then the (independent) unknowns. Initial conditions $\boldsymbol{k}^{\mathrm{ini}}$ (Eq. (1.1b)) are not necessarily required.

As the structure of the DAE is not further investigated system Eq. (3.1) may be determined, overdetermined, or underdetermined according to $\boldsymbol{k}^{\mathrm{ini}}$. In order to address this problem special iterative solution methods for nonlinear systems have been considered in [CKY 96]. They range from plain, damped, or truncated Gauß-Newton schemes where the linear systems are solved in a generalised sense (application of the Moore-Penrose pseudo-inverse generated by singular value decomposition), over steepest descend residual minimisation to the Levenberg-Marquardt method. The sequential linear programming approach proposed by [GoBi 99] (cf. Section 3.1.2) is also mentioned.

All these numerical methods for the solution of Eq. (3.1) are numerically expensive, especially for larger problems. Therefore, [CKY 96] restrict the applicability of their approach to initialisation problems of moderate size. Additionally, we see that the absence of a structural investigation of the DAE – which appears to be an advantage – can lead to somewhat arbitrary results as the problem of assignment of the dynamic degrees of freedom is not addressed. *Arbitrary* means that the resulting consistent initial conditions are not determined by known properties of the DAE, but by the initial guess and by the convergence behaviour of the algorithm employed for the solution of the nonlinear system of equations Eq. (3.1).

3.1.2 An SLP-Formulation for the Solution of the Consistency Equations

As seen in Section 3.1.1 the direct solution of the consistency equations Eq. (3.1) is a difficult task which requires the application of numerically expensive solvers

for rank-deficient problems. As an alternative [GoBi 99] propose to formulate the consistency equations as an NLP that can be treated using nonlinear optimisation methods. Similar to Section 3.1.1 they do not need to know the number of dynamic degrees of freedom. Additionally, the requirements that have to be imposed on the quality of the user given transition conditions can be relaxed.

In the sequel we consider the DAE

$$0 = \boldsymbol{F}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \,, \tag{3.2}$$

$\boldsymbol{F} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}} \to \mathbb{R}^{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}$, of differential index ι_d and its derivative array equations

$$\vec{\boldsymbol{F}}_{\iota_\mathrm{d}}(t, \boldsymbol{x}(t), \ldots, \boldsymbol{x}^{(\iota_\mathrm{d}+1)}(t), \boldsymbol{y}(t), \ldots, \boldsymbol{y}^{(\iota_\mathrm{d})}(t)) := \begin{bmatrix} \boldsymbol{F}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \\ \vdots \\ \frac{\mathrm{d}^{\iota_\mathrm{d}}}{\mathrm{d}t^{\iota_\mathrm{d}}} \boldsymbol{F}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \end{bmatrix} . \tag{3.3}$$

Commonly, (cf., e.g., Section 3.1.4, Section 3.1.5, Section 3.1.11, and the discussion in Section 3.1.13) the user is asked for appropriate initial conditions $\boldsymbol{k}^{\mathrm{ini}}$ fixing the dynamic degrees of freedom of the DAE Eq. (3.2). Then consistent initial conditions are determined by the (numerically) exactly solution of the consistency equations composed of the derivative array equations Eq. (3.3) together with the initial conditions, e.g., Eq. (1.1b).

[GoBi 99] start from a different point of view: Given conditions $\boldsymbol{k}^{\mathrm{ini}}$ specifying the initial state they calculate consistent initial conditions that satisfy the derivative array equations and these specifications as exact *as possible*. Applied to Eq. (3.2) or Eq. (3.3), respectively, this idea leads to the optimisation formulation Eq. (3.4):

$$\sum_{\nu=1}^{m_{\boldsymbol{k}^{\mathrm{ini}}}} \boldsymbol{\omega}_\nu \boldsymbol{\delta}_\nu \longrightarrow \min_{\substack{\boldsymbol{x}, \ldots, \boldsymbol{x}^{(\iota_\mathrm{d}+1)}; \\ \boldsymbol{y}, \ldots, \boldsymbol{y}^{(\iota_\mathrm{d})}; \, \boldsymbol{\delta}}} ! \tag{3.4}$$

subject to

$$\begin{aligned} 0 &= \vec{\boldsymbol{F}}_{\iota_\mathrm{d}}(t, \boldsymbol{x}, \ldots, \boldsymbol{x}^{(\iota_\mathrm{d}+1)}, \boldsymbol{y}, \ldots, \boldsymbol{y}^{(\iota_\mathrm{d})}) \,, \\ -\boldsymbol{\delta}_\nu &\le \boldsymbol{k}^{\mathrm{ini}}_\nu(\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}, \boldsymbol{x}_0, \boldsymbol{y}_0, \dot{\boldsymbol{x}}_0) \le \boldsymbol{\delta}_\nu \,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}} \,, \\ 0 &\le \boldsymbol{\delta}_\nu \,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}} \,, \end{aligned}$$

where $\boldsymbol{\omega}_\nu \in \mathbb{R}_+ \backslash \{0\}$, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$, are weights. $\boldsymbol{k}^{\mathrm{ini}}_\nu$, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$, are relations of the type "$\boldsymbol{z} - \boldsymbol{z}_0$" each of them fixing a certain variable or its derivative. The values given in the corresponding $m_{\boldsymbol{k}^{\mathrm{ini}}}$ elements of the vectors $\boldsymbol{x}_0$, $\boldsymbol{y}_0$, and $\dot{\boldsymbol{x}}_0$ are the desired initial values. Additionally, box constraints for the optimisation variables, especially for $\boldsymbol{x}, \ldots, \boldsymbol{x}^{(\iota_\mathrm{d}+1)}$ and $\boldsymbol{y}, \ldots, \boldsymbol{y}^{(\iota_\mathrm{d})}$, can be specified.

By assumption it may not not be possible to satisfy all initial conditions $\boldsymbol{k}^{\mathrm{ini}}_\nu$,

$\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$. Therefore the objective of the NLP Eq. (3.4) is to minimise the weighted sum of the infeasibilities $\boldsymbol{\delta}_\nu \in \mathbb{R}_+$, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$, of the initial conditions. However, this minimisation of the weighted 1-norm of the violation of the initial conditions $\boldsymbol{k}^{\mathrm{ini}}$ is the main drawback of the formulation of the consistent initialisation problem given by Eq. (3.4) as the number of variables actually set *equal* to their values specified by $\boldsymbol{k}^{\mathrm{ini}}$ at the numerically obtained solution of the optimisation problem cannot be explicitly controlled [GoBi 99].

On the other hand, if the number of dynamic degrees of freedom n_{ddf} is known it appears advantageous to enforce exactly n_{ddf} of the initial conditions $\boldsymbol{k}^{\mathrm{ini}}$ (given that $\boldsymbol{k}^{\mathrm{ini}}$ contains a subset of suitable initial conditions). This idea leads to the mixed-integer nonlinear programming problem (MINLP)

$$\sum_{\nu=1}^{m_{\boldsymbol{k}^{\mathrm{ini}}}} \boldsymbol{\omega}_\nu \boldsymbol{\delta}_\nu \longrightarrow \min_{\substack{\boldsymbol{x},\ldots,\boldsymbol{x}^{(\iota_\mathrm{d}+1)};\\ \boldsymbol{y},\ldots,\boldsymbol{y}^{(\iota_\mathrm{d})};\, \boldsymbol{\delta},\boldsymbol{\lambda}}} !$$

subject to

$$\begin{aligned}
0 &= \vec{\boldsymbol{F}}_{\iota_\mathrm{d}}(t, \boldsymbol{x}, \ldots, \boldsymbol{x}^{(\iota_\mathrm{d}+1)}, \boldsymbol{y}, \ldots, \boldsymbol{y}^{(\iota_\mathrm{d})}) \,, \\
-\boldsymbol{\delta}_\nu \le \boldsymbol{k}_\nu^{\mathrm{ini}}(\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}, \boldsymbol{x}_0, \boldsymbol{y}_0, \dot{\boldsymbol{x}}_0) &\le \boldsymbol{\delta}_\nu \,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}} \,, \\
0 \le \boldsymbol{\delta}_\nu &\le M_\nu (1 - \boldsymbol{\lambda}_\nu) \,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}} \,, \\
0 &= n_{\mathrm{ddf}} - \sum_{\nu=1}^{m_{\boldsymbol{k}^{\mathrm{ini}}}} \boldsymbol{\lambda}_\nu \,, \\
\boldsymbol{\lambda}_\nu &\in \{0, 1\} \,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}} \,,
\end{aligned}$$

with the real optimisation variables $\boldsymbol{x}, \ldots, \boldsymbol{x}^{(\iota_\mathrm{d}+1)}$, $\boldsymbol{y}, \ldots, \boldsymbol{y}^{(\iota_\mathrm{d})}$, $\boldsymbol{\delta}_\nu$, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$, and the binary variables $\boldsymbol{\lambda}_\nu$, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$. M_ν, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$, are large positive numbers (method of *big-M constraints*, cf., e.g., [Kels 95]). However, the solution of MINLPs is nontrivial and computationally demanding. The idea of [GoBi 99] is to consider the *approximated* (or relaxed) problem Eq. (3.5):

$$\sum_{\nu=1}^{m_{\boldsymbol{k}^{\mathrm{ini}}}} \boldsymbol{\omega}_\nu \boldsymbol{\delta}_\nu \longrightarrow \min_{\substack{\boldsymbol{x},\ldots,\boldsymbol{x}^{(\iota_\mathrm{d}+1)};\\ \boldsymbol{y},\ldots,\boldsymbol{y}^{(\iota_\mathrm{d})};\, \boldsymbol{\delta},\boldsymbol{\lambda}}} ! \tag{3.5}$$

subject to

$$\begin{aligned}
0 &= \vec{\boldsymbol{F}}_{\iota_\mathrm{d}}(t, \boldsymbol{x}, \ldots, \boldsymbol{x}^{(\iota_\mathrm{d}+1)}, \boldsymbol{y}, \ldots, \boldsymbol{y}^{(\iota_\mathrm{d})}) \,, \\
-\boldsymbol{\delta}_\nu \le \boldsymbol{k}_\nu^{\mathrm{ini}}(\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}, \boldsymbol{x}_0, \boldsymbol{y}_0, \dot{\boldsymbol{x}}_0) &\le \boldsymbol{\delta}_\nu \,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}} \,, \\
0 &= \boldsymbol{\delta}_\nu - \beta \cdot \log\left(1 + \exp\left(-(\boldsymbol{\lambda}_\nu - \boldsymbol{\delta}_\nu)/\beta\right)\right) \,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}} \,, \\
0 &= n_{\mathrm{ddf}} - \sum_{\nu=1}^{m_{\boldsymbol{k}^{\mathrm{ini}}}} \boldsymbol{\lambda}_\nu \,,
\end{aligned}$$

$$0 \leq \boldsymbol{\lambda}_\nu \leq 1\,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}\,,$$
$$0 \leq \boldsymbol{\delta}_\nu\,, \qquad \nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}\,,$$

where $\boldsymbol{\lambda}_\nu \in [0;1]$, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$, are continuous substitutes for the binary $\{0,1\}$-variables. $\beta \in \mathbb{R}_+ \backslash \{0\}$ is a parameter used for the smooth approximation of the max-function, which is approached for $\beta \to 0$:

$$(\boldsymbol{\lambda}_\nu - \boldsymbol{\delta}_\nu) + \beta \cdot \log\left(1 + \exp\left(-(\boldsymbol{\lambda}_\nu - \boldsymbol{\delta}_\nu)/\beta\right)\right) \overset{\beta \to 0}{\longrightarrow} \max\left\{0, (\boldsymbol{\lambda}_\nu - \boldsymbol{\delta}_\nu)\right\}\,.$$

This relation is applied in order to gradually enforce complementary conditions $\boldsymbol{\lambda}_\nu \boldsymbol{\delta}_\nu = 0$, $\nu = 1, \ldots, m_{\boldsymbol{k}^{\mathrm{ini}}}$, present in the MINLP by

$$0 = \boldsymbol{\lambda}_\nu \cdot \boldsymbol{\delta}_\nu \Leftrightarrow 0 = (\boldsymbol{\lambda}_\nu - \boldsymbol{\delta}_\nu)_+ - \boldsymbol{\lambda}_\nu = \max\left\{0, (\boldsymbol{\lambda}_\nu - \boldsymbol{\delta}_\nu)\right\} - \boldsymbol{\lambda}_\nu\,.$$

This technique can be used for the computation of consistent initial conditions of a DAE at both, the initial point t_0, and after a discontinuity. In the latter case [GoBi 99] examine the automatic determination of a set of variables which can be assigned continuity conditions across the discontinuity in order to obtain appropriate transition conditions. Especially, they note that the assumption of continuity in *all* differential variables $\boldsymbol{x}$ is not always true. More specifically, in the case of discontinuous *forcing functions* (i.e., *input variables*, or from the optimal control point of view *control variables*) [GoBi 99] show that only those variables can be assumed to be continuous whose UODEs are independent from any derivative of the discontinuous input. Their further examinations regarding transition conditions at discontinuities are restricted to step-discontinuities in forcing functions of linear DAEs. For this special case they develop two methods for the automatic generation of transition conditions. The first method is based on the analysis of the Jacobian of the derivative array equations. The second method uses sensitivity information from an additional *sequential linear programming problem* (SLP). Only the latter method can be safely applied to nonlinear problems.

[CKY 96] (cf. Section 3.1.1) mention the SLP approach as proposed in an earlier article related to [GoBi 99]. [CKY 96] employ a Gauß-Newton variant for two reasons. On the one hand, they intend to use the same data for the computation of consistent initial conditions as is used in their integrator algorithm for higher index DAEs. On the other hand, they consider the Gauß-Newton solver as the more robust method.

3.1.3 Approximations to the Consistency Equations

For the construction of the consistency equations Eq. (3.1) (or more precisely, for the construction of the derivative array equations), higher order total time derivatives of the DAE are required. If the DAE is avaliable in symbolic form then the derivatives can be generated very efficiently, cf., e.g., [Feeh 98]. On the other hand, if the equations are given as executable routines automatic code differen-

tiation tools such as ADIFOR [BCCGH 92], ADOL-F [GrSh 96], or ADOL-C [GJU 96] may be applicable. E.g., [CVSB 01] employ ADOL-C in this context. However, the two approaches mentioned above are not always suitable. Symbolic manipulations are cancelled out if the equations are not available in symbolic form. Automatic differentiation of the program code may not be applicable due to lack of additional resources for implementation as it may require significant reorganisation of large programs. These two conditions can apply to real-life applications where the DAE is formed numerically by very complex simulation tools. Then numerical differentiation provides a remedy for the calculation of the required time derivatives. But this approach can be computationally expensive and bears numerical difficulties. Therefore, in [Leim 88], [LPG 91] a refined technique for the efficient and reliable numerical approximation of the required derivatives is developed. This method is discussed in the sequel.

In order to simplify notation we define $\Xi(t) := [t, \boldsymbol{\xi}(t)^T, \dot{\boldsymbol{\xi}}(t)^T]^T \in \mathbb{R}^{2n_{\boldsymbol{\xi}}+1}$. Then the general DAE initial value problem Eqs. (1.1a)–(1.1b) reads as

$$\begin{aligned} \overline{\boldsymbol{F}}(\Xi(t)) &= 0\,; \quad t \in [t_0, t_\mathrm{f}] \subset \mathbb{R}, \\ \overline{\boldsymbol{k}^\mathrm{ini}}(\Xi(t_0)) &= 0\,. \end{aligned}$$

The primary idea in [LPG 91] is to choose points $\Xi_1, \ldots, \Xi_{\lambda_\mathrm{s}}$ in state space which are truncated Taylor's series expansions of the solution $\Xi(t)$ at $t_0 + c_1 h, \ldots, t_0 + c_{\lambda_\mathrm{s}} h$, $c_\nu \in \mathbb{R}_+ \backslash \{0\}$, $\nu = 1, \ldots, \lambda_\mathrm{s}$, $h \in \mathbb{R}_+ \backslash \{0\}$, and then build linear combinations of $\overline{\boldsymbol{F}}(\Xi_\nu)$, $\nu = 1, \ldots, \lambda_\mathrm{s}$, in such a way that certain higher order terms in the Taylor's series expansion of $\overline{\boldsymbol{F}}(\Xi(t_0))$ are annihilated. The restriction to positive c_ν is imposed as due to the time order of the problem forward differencing may be allowed only. These considerations lead to the family of approximations for the k^th order total time derivative of $\overline{\boldsymbol{F}}(\Xi(t))$ at t_0

$$\begin{aligned} D_h^k \overline{\boldsymbol{F}} := \frac{k!}{h^k} \Bigg\{ \sum_{\nu=1}^{\lambda_\mathrm{s}} \alpha_\nu \overline{\boldsymbol{F}} \left(\Xi_0 + c_\nu h \dot{\Xi}_0 + \frac{(c_\nu h)^2}{2!} \ddot{\Xi}_0 + \ldots + \frac{(c_\nu h)^{\lambda_\mathrm{t}}}{\lambda_\mathrm{t}!} \Xi_0^{(\lambda_\mathrm{t})} \right) \\ - \left(\sum_{\nu=1}^{\lambda_\mathrm{s}} \alpha_\nu \right) \overline{\boldsymbol{F}}(\Xi_0) \Bigg\}\,, \end{aligned} \tag{3.6}$$

where $\Xi_0^{(\mu)} := d^\mu \Xi(t)/dt^\mu \big|_{t=t_0}$, $\mu = 1, \ldots, \lambda_\mathrm{t}$. λ_t is the order of the Taylor's series expansion of $\Xi(t)$, and $\alpha_\nu \in \mathbb{R}$, $\nu = 1, \ldots, \lambda_\mathrm{s}$, are coefficients that have to be selected appropriately (see Eq. (3.8) below). Defining $c_0 := 0$ and $\alpha_0 := -\sum_{\nu=1}^{\lambda_\mathrm{s}} \alpha_\nu$, Eq. (3.6) can be formulated more concisely as

$$D_h^k \overline{\boldsymbol{F}} = \frac{k!}{h^k} \left\{ \sum_{\nu=0}^{\lambda_\mathrm{s}} \alpha_\nu \overline{\boldsymbol{F}} \left(\sum_{\mu=0}^{\lambda_\mathrm{t}} \frac{(c_\nu h)^\mu}{\mu!} \Xi_0^{(\mu)} \right) \right\}. \tag{3.7}$$

The approximation properties of the family $D_h^k \overline{\boldsymbol{F}}$ are characterised by the following

theorem [LPG 91]:

Theorem 3.1 (Truncation Error of Derivative Approximation)
Let $\overline{\boldsymbol{F}} : \mathbb{R}^{n_{\boldsymbol{\Xi}}} \to \mathbb{R}^{m_{\overline{\boldsymbol{F}}}}$, $\boldsymbol{\Xi} : \mathbb{R} \to \mathbb{R}^{n_{\boldsymbol{\Xi}}}$. *Suppose* $\lambda_t \geq k$. *If* $\boldsymbol{\Xi}$ *is* $\lambda_t + 1$ *times differentiable in a neighbourhood of* $t = t_0$ *and* $\overline{\boldsymbol{F}}$ *is* $\mathcal{C}^{p+1}$ *in a neighbourhood of* $\boldsymbol{\Xi}(t_0)$ *for some* $p \geq \lambda_t$, *then the truncation error of Eq.* (3.6), *as an approximation to the* k^{th} *derivative, satisfies*

$$D_h^k \overline{\boldsymbol{F}} - \left.\frac{d^k \overline{\boldsymbol{F}}(\boldsymbol{\Xi}(t))}{dt^k}\right|_{t=t_0} = \mathcal{O}(h^l)\,,$$

provided that $p \geq k + l - 1$, *and*

$$\sum_{\nu=1}^{\lambda_s} \alpha_\nu (c_\nu)^\mu = \begin{cases} 1\,; & \mu = k\,, \\ 0\,; & \mu \neq k\,, \end{cases} \quad \mu = 1, \ldots, k + l - 1\,. \tag{3.8}$$

The maximum attainable order of approximation l is specified below in Theorem 3.2. Beforehand, the existence of the approximations introduced in Theorem 3.1 has to be shown. This means that the existence of appropriate coefficients $\alpha_\nu, \nu = 1, \ldots, \lambda_s$, determined by Eq. (3.8) has to be guaranteed [LPG 91]:

Lemma 3.1 (Existence of the Expansion Coefficients)
Let $c_1, c_2, \ldots, c_{\lambda_s}$ *be* λ_s *distinct, positive real numbers. For* $\nu = k, k+1, \ldots$ *let* $\boldsymbol{e}_k^\nu$ *be the* k^{th} *standard basis vector in* $\mathbb{R}^\nu$ *and define* $\alpha := [\alpha_1, \ldots, \alpha_{\lambda_s}]^T \in \mathbb{R}^{\lambda_s}$ *and*

$$M := \begin{bmatrix} c_1 & c_2 & \ldots & c_{\lambda_s} \\ (c_1)^2 & (c_2)^2 & \ldots & (c_{\lambda_s})^2 \\ \vdots & \vdots & & \vdots \\ (c_1)^\nu & (c_2)^\nu & \ldots & (c_{\lambda_s})^\nu \end{bmatrix} \in \mathbb{R}^{\nu \times \lambda_s}\,.$$

Then the system

$$M\alpha = \boldsymbol{e}_k^\nu$$

has a solution if and only if $\nu \leq \lambda_s$.

For $c_\nu = 0, \ldots, \lambda_s$, $\lambda_s = k, \ldots, 7$, [Leim 88] gives the corresponding coefficients α_ν determined by Eq. (3.8) providing the maximum order of approximation in the $k = 1^{\text{st}}$ to $k = 3^{\text{rd}}$ total time derivative for Eq. (3.7). In Table 3.1 the coefficients for the 1^{st} total time derivative are enlisted as they will be used in Section 3.2.3 for setting up the derivative array equations.

Finally, we are interested in the maximum order of an approximation that can be expected from Eq. (3.6) [LPG 91]:

Theorem 3.2 (Maximum Attainable Order of Approximation)
Let $\boldsymbol{\Xi}$, $\overline{\boldsymbol{F}}$ *be as in Theorem 3.1, then the maximum attainable order of the* k^{th} *derivative approximation Eq.* (3.6) *is* $\min\{p - k + 1, \lambda_s - k + 1\}$.

Table 3.1: Coefficients for approximation of 1st order total time derivative providing the maximum order of approximation with $c_\nu = 0, \ldots, \lambda_s$, $\lambda_s = k, \ldots, 7$, according to [Leim 88].

λ_s	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7
1	-1	1						
2	-3/2	2	-1/2					
3	-11/6	3	-3/2	1/3				
4	-25/12	4	-3	4/3	-1/4			
5	-137/60	5	-5	10/3	-5/4	1/5		
6	-49/20	6	-15/2	20/3	-15/4	6/5	-1/6	
7	-363/140	7	-21/2	35/3	-35/4	21/5	-7/6	1/7

After approximation of the consistency equations the problem of solving this nonlinear system of equations still remains. As already noted in Section 3.1.1 and Section 3.1.2 the full system of consistency equations Eq. (3.1) – here set up using the numerical approximation specified in Eq. (3.6) – seen as one nonlinear system of equations in the state variables and their (higher order) derivatives is a rank-deficient problem. Moreover, inappropriate transition conditions given by the user can lead to an unsolvable system. The *additional* difficulty introduced by the numerical approximation of the derivative array equations is that the resulting system may not be solvable even if a symbolically derived system of consistency equations owns a well-defined solution. Therefore, [LPG 91] propose to solve the approximated consistency equations in a least-squares sense.

Remark 3.1:
For some special cases [LPG 91] directly derive reductions to full-column rank problems which are easier to solve. The construction of a full rank system of reduced consistency equations for semi-explicit index-2 DAEs is one of the core steps in our algorithm described in Section 3.2. ◊

3.1.4 Semi-Explicit Index-2 DAEs

[Grit 90], [GPS 95] address optimal control of semi-explicit index-2 DAEs arising from chemical engineering. In order to solve the optimal control problem a direct shooting method is employed where the control functions are approximated by parametrised basis functions on a control mesh. Although no implicitly defined discontinuities are considered the piecewise structure of the controls causes a consistent initialisation problem at every node of the control mesh.

Remark 3.2:
As we have already noted in Section 2.4.2.a, [Gerd 01] has shown that these consistent initialisation problems can be avoided if sufficiently smooth controls are chosen. ◊

In this section we summarise the analysis given in [GPS 95] as it elaborates some properties of semi-explicit index-2 DAEs. For ease of notation we suppress the additional argument $\boldsymbol{p}$ in all functions representing the vector of optimisation parameters. Now consider the semi-explicit index-2 DAE

$$0 = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \,, \tag{3.9a}$$

$$0 = \boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{y}(t)) \,, \tag{3.9b}$$

$$0 = \boldsymbol{k}^{\mathrm{ini}}(t_0, \boldsymbol{x}(t_0), \boldsymbol{y}(t_0), \dot{\boldsymbol{x}}(t_0)) \,, \tag{3.9c}$$

with the differential equations $\boldsymbol{f} \in \mathcal{C}^2\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}}, \mathbb{R}^{n_{\boldsymbol{x}}}\right)$, the algebraic equations $\boldsymbol{g} \in \mathcal{C}^2\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}, \mathbb{R}^{n_{\boldsymbol{y}}}\right)$, and appropriate initial conditions $\boldsymbol{k}^{\mathrm{ini}} \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}}, \mathbb{R}^{n_{\mathrm{ddf}}}\right)$. The number of dynamic degrees of freedom n_{ddf} at the initial time t_0 is assumed to be known as well as the differential index of the DAE.

As the DAE is semi-explicit by assumption, we have $\det([\partial \boldsymbol{f}/\partial \dot{\boldsymbol{x}}]) \neq 0$ along the solution. Additionally, $\det([\partial \boldsymbol{g}/\partial \boldsymbol{y}]) = 0$ holds along the solution trajectory as otherwise Eqs. (3.9a)–(3.9b) represent an index-1 problem (the case of a locally changing index is not considered). Thus due to the restriction to index-2 problems the derivative array equations

$$\left.\begin{aligned} 0 &= \boldsymbol{f} \,, \\ 0 &= \boldsymbol{g} \,, \end{aligned}\right\} \tag{3.10a}$$

$$\left.\begin{aligned} 0 &= \tfrac{\partial \boldsymbol{f}}{\partial t} + \tfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} + \tfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}} + \tfrac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}} \,, \\ 0 &= \tfrac{\partial \boldsymbol{g}}{\partial t} + \tfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} + \tfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}} \,, \end{aligned}\right\} \tag{3.10b}$$

$$\left.\begin{aligned} 0 &= \left(\tfrac{\partial}{\partial t}\tfrac{\partial \boldsymbol{f}}{\partial t}\right) + \left(\tfrac{\partial}{\partial t}\tfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)\dot{\boldsymbol{x}} + \left(\tfrac{\partial}{\partial t}\tfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right)\dot{\boldsymbol{y}} + \left(\left(\tfrac{\partial}{\partial t}\tfrac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\right) + \tfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)\ddot{\boldsymbol{x}} + \tfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\ddot{\boldsymbol{y}} + \tfrac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}}\dddot{\boldsymbol{x}} \,, \\ 0 &= \left(\tfrac{\partial}{\partial t}\tfrac{\partial \boldsymbol{g}}{\partial t}\right) + \left(\tfrac{\partial}{\partial t}\tfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right)\dot{\boldsymbol{x}} + \left(\tfrac{\partial}{\partial t}\tfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\right)\dot{\boldsymbol{y}} + \tfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\ddot{\boldsymbol{x}} + \tfrac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\ddot{\boldsymbol{y}} \,, \end{aligned}\right\} \tag{3.10c}$$

uniquely determine $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{y}}$ as a function of $\boldsymbol{x}$, $\boldsymbol{y}$, and t according to Definition 1.9. In Eqs. (3.10a)–(3.10c) the arguments of $\boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t))$, $\boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{y}(t))$, $\boldsymbol{x}(t)$, $\boldsymbol{y}(t)$, and of their derivatives have been omitted for the sake of brevity as will be in the sequel.

As $\det([\partial \boldsymbol{f}/\partial \dot{\boldsymbol{x}}]) \neq 0$, $\dot{\boldsymbol{x}}$ can be determined from $\boldsymbol{f} = 0$ in Eq. (3.10a). Thus by definition the initialisation problem is solved when an appropriate value of $\dot{\boldsymbol{y}}$ is available. Let $r_{\boldsymbol{g}} < n_{\boldsymbol{y}}$ be the rank of the matrix $[\partial \boldsymbol{g}/\partial \boldsymbol{y}] \in \mathbb{R}^{n_{\boldsymbol{y}} \times n_{\boldsymbol{y}}}$. Then it is possible to find a Gaussian elimination $L = L(t, \boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}})$, $L : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}} \to \mathbb{R}^{(n_{\boldsymbol{x}}+n_{\boldsymbol{y}}) \times (n_{\boldsymbol{x}}+n_{\boldsymbol{y}})}$, regular and lower-triangular, and a partitioning $\bar{\boldsymbol{y}} \in \mathbb{R}^{r_{\boldsymbol{g}}}$, $\bar{\boldsymbol{z}} \in \mathbb{R}^{n_{\boldsymbol{y}}-r_{\boldsymbol{g}}}$ of the algebraic variables $\boldsymbol{y}$ such that

$$L \cdot \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} \\ \boldsymbol{0} & \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}} & \bar{\boldsymbol{f}}_{\boldsymbol{y}} \\ \boldsymbol{0} & \bar{\boldsymbol{g}}_{\boldsymbol{y}} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}} & \bar{\boldsymbol{f}}_{\bar{\boldsymbol{y}}} & \bar{\boldsymbol{f}}_{\bar{\boldsymbol{z}}} \\ \boldsymbol{0} & \bar{\boldsymbol{g}}_{\bar{\boldsymbol{y}}} & \bar{\boldsymbol{g}}_{\bar{\boldsymbol{z}}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{matrix} \} n_{\boldsymbol{x}} \\ \} r_{\boldsymbol{g}} \\ \} n_{\boldsymbol{y}} - r_{\boldsymbol{g}} \end{matrix} \,, \tag{3.11}$$

where $[\bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}}]$ and $[\bar{\boldsymbol{g}}_{\bar{\boldsymbol{y}}}]$ are unit upper-triangular. Thus $[\bar{\boldsymbol{g}}_{\bar{\boldsymbol{y}}}]$ is of full rank $r_{\boldsymbol{g}}$. Using $[L]$ transformed functions are *defined* by

$$\begin{bmatrix}\bar{\boldsymbol{f}}\\ \bar{\boldsymbol{g}}\\ \bar{\boldsymbol{s}}\end{bmatrix} := L \cdot \begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{g}\end{bmatrix}, \quad \begin{bmatrix}\bar{\boldsymbol{f}}_{\boldsymbol{x}}\\ \bar{\boldsymbol{g}}_{\boldsymbol{x}}\\ \bar{\boldsymbol{s}}_{\boldsymbol{x}}\end{bmatrix} := L \cdot \begin{bmatrix}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\\ \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\end{bmatrix}, \quad \begin{bmatrix}\bar{\boldsymbol{f}}'_{\boldsymbol{y}}\\ \bar{\boldsymbol{g}}'_{\boldsymbol{y}}\\ \bar{\boldsymbol{s}}'_{\boldsymbol{y}}\end{bmatrix} := L \cdot \begin{bmatrix}\frac{\partial}{\partial t}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\\ \frac{\partial}{\partial t}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\end{bmatrix}, \ \ldots$$

Please note that this notation is not compatible with the notation employed in the remainder of this treatise, e.g., the function denoted with the *symbol* $\bar{\boldsymbol{f}}_{\boldsymbol{x}}$ is in general not equal to the *partial derivative* $\partial \bar{\boldsymbol{f}}/\partial \boldsymbol{x}$

$$\begin{bmatrix}\frac{\partial}{\partial \boldsymbol{x}}\bar{\boldsymbol{f}}\\ \frac{\partial}{\partial \boldsymbol{x}}\bar{\boldsymbol{g}}\\ \frac{\partial}{\partial \boldsymbol{x}}\bar{\boldsymbol{s}}\end{bmatrix} = \frac{\partial L}{\partial \boldsymbol{x}} \cdot \begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{g}\end{bmatrix} + L \cdot \begin{bmatrix}\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{f}\\ \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{g}\end{bmatrix} = \frac{\partial L}{\partial \boldsymbol{x}} \cdot \begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{g}\end{bmatrix} + \begin{bmatrix}\bar{\boldsymbol{f}}_{\boldsymbol{x}}\\ \bar{\boldsymbol{g}}_{\boldsymbol{x}}\\ \bar{\boldsymbol{s}}_{\boldsymbol{x}}\end{bmatrix},$$

as in general $\partial L/\partial \boldsymbol{x} \neq \boldsymbol{0}$. Thus the functions representing the transformed derivatives are not equal to the derivatives of the transformed functions.

Using this newly defined functions multiplication of Eqs. (3.10a)–(3.10c) with $[L]$ from the left gives

$$0 = \bar{\boldsymbol{f}}, \tag{3.12a}$$
$$0 = \bar{\boldsymbol{g}}, \tag{3.12b}$$
$$0 = \bar{\boldsymbol{s}}, \tag{3.12c}$$
$$0 = \bar{\boldsymbol{f}}_t + \bar{\boldsymbol{f}}_{\boldsymbol{x}}\dot{\boldsymbol{x}} + \bar{\boldsymbol{f}}_{\boldsymbol{y}}\dot{\boldsymbol{y}} + \bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}}, \tag{3.12d}$$
$$0 = \bar{\boldsymbol{g}}_t + \bar{\boldsymbol{g}}_{\boldsymbol{x}}\dot{\boldsymbol{x}} + \bar{\boldsymbol{g}}_{\boldsymbol{y}}\dot{\boldsymbol{y}}, \tag{3.12e}$$
$$0 = \bar{\boldsymbol{s}}_t + \bar{\boldsymbol{s}}_{\boldsymbol{x}}\dot{\boldsymbol{x}}, \tag{3.12f}$$
$$0 = \bar{\boldsymbol{f}}'_t + \bar{\boldsymbol{f}}'_{\boldsymbol{x}}\dot{\boldsymbol{x}} + \bar{\boldsymbol{f}}'_{\boldsymbol{y}}\dot{\boldsymbol{y}} + \left(\bar{\boldsymbol{f}}'_{\dot{\boldsymbol{x}}} + \bar{\boldsymbol{f}}_{\boldsymbol{x}}\right)\ddot{\boldsymbol{x}} + \bar{\boldsymbol{f}}_{\boldsymbol{y}}\ddot{\boldsymbol{y}} + \bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}}\dddot{\boldsymbol{x}}, \tag{3.12g}$$
$$0 = \bar{\boldsymbol{g}}'_t + \bar{\boldsymbol{g}}'_{\boldsymbol{x}}\dot{\boldsymbol{x}} + \bar{\boldsymbol{g}}'_{\boldsymbol{y}}\dot{\boldsymbol{y}} + \bar{\boldsymbol{g}}_{\boldsymbol{x}}\ddot{\boldsymbol{x}} + \bar{\boldsymbol{g}}_{\boldsymbol{y}}\ddot{\boldsymbol{y}}, \tag{3.12h}$$
$$0 = \bar{\boldsymbol{s}}'_t + \bar{\boldsymbol{s}}'_{\boldsymbol{x}}\dot{\boldsymbol{x}} + \bar{\boldsymbol{s}}'_{\boldsymbol{y}}\dot{\boldsymbol{y}} + \bar{\boldsymbol{s}}_{\boldsymbol{x}}\ddot{\boldsymbol{x}}, \tag{3.12i}$$

as $[\bar{\boldsymbol{s}}_{\boldsymbol{y}}] = \boldsymbol{0}$ everywhere. [GPS 95] show the following properties of the transformed system (and thus of the original DAE):

a) As Eqs. (3.9a)–(3.9b) form an index-2 DAE $\dot{\boldsymbol{y}}(t_0)$ must be uniquely determined by Eqs. (3.12d)–(3.12e) and Eq. (3.12i). A *sufficient* condition is

$$\det\left(\begin{bmatrix}\bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}} & \bar{\boldsymbol{f}}_{\boldsymbol{y}}\\ \boldsymbol{0} & \bar{\boldsymbol{g}}_{\boldsymbol{y}}\\ \bar{\boldsymbol{s}}_{\boldsymbol{x}} & \bar{\boldsymbol{s}}'_{\boldsymbol{y}}\end{bmatrix}_t\right) \neq 0 \quad \forall t \in [t_0, t_{\mathrm{f}}]. \tag{3.13}$$

In the sequel it is assumed that Eq. (3.13) holds.

b) The initial conditions $\boldsymbol{k}^{\mathrm{ini}}$ specified in Eq. (3.9c) have to determine $\dot{\boldsymbol{x}}(t_0)$, $\boldsymbol{x}(t_0)$,

and $\boldsymbol{y}(t_0)$ locally unique together with Eqs. (3.12a)–(3.12c) and Eq. (3.12f). Therefore the corresponding Jacobian has to be of full rank, i.e.,

$$\det\left(\begin{bmatrix} \bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}} & \bar{\boldsymbol{f}}_{\boldsymbol{x}} & \bar{\boldsymbol{f}}_{\boldsymbol{y}} \\ \mathbf{0} & \bar{\boldsymbol{g}}_{\boldsymbol{x}} & \bar{\boldsymbol{g}}_{\boldsymbol{y}} \\ \mathbf{0} & \bar{\boldsymbol{s}}_{\boldsymbol{x}} & \mathbf{0} \\ \bar{\boldsymbol{s}}_{\boldsymbol{x}} & \bar{\boldsymbol{s}}'_{\boldsymbol{x}} & \bar{\boldsymbol{s}}'_{\boldsymbol{y}} \\ \frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{y}} \end{bmatrix}_{t_0}\right) = \det\left(\left[\begin{array}{cc|c} \bar{\boldsymbol{f}}_{\dot{\boldsymbol{x}}} & \bar{\boldsymbol{f}}_{\boldsymbol{y}} & \bar{\boldsymbol{f}}_{\boldsymbol{x}} \\ \mathbf{0} & \bar{\boldsymbol{g}}_{\boldsymbol{y}} & \bar{\boldsymbol{g}}_{\boldsymbol{x}} \\ \bar{\boldsymbol{s}}_{\boldsymbol{x}} & \bar{\boldsymbol{s}}'_{\boldsymbol{y}} & \bar{\boldsymbol{s}}'_{\boldsymbol{x}} \\ \hline \mathbf{0} & \mathbf{0} & \bar{\boldsymbol{s}}_{\boldsymbol{x}} \\ \frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{x}} \end{array}\right]_{t_0}\right) \neq 0\,. \quad (3.14)$$

Under assumption Eq. (3.13) the upper left part of the matrix on the right hand side has full rank $n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$ and $\bar{\boldsymbol{s}}_{\boldsymbol{x}}$ is of full rank $n_{\boldsymbol{y}} - r_{\boldsymbol{g}}$. Thus, the number of dynamic degrees of freedom that can (and have to) be specified by $\boldsymbol{k}^{\mathrm{ini}}$ is

$$n_{\mathrm{ddf}} = (2n_{\boldsymbol{x}} + n_{\boldsymbol{y}}) - (n_{\boldsymbol{x}} + n_{\boldsymbol{y}} + (n_{\boldsymbol{y}} - r_{\boldsymbol{g}})) = n_{\boldsymbol{x}} - (n_{\boldsymbol{y}} - r_{\boldsymbol{g}})\,.$$

If the initial conditions $\boldsymbol{k}^{\mathrm{ini}}$ are chosen such that $\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \dot{\boldsymbol{x}}} \equiv \mathbf{0}$ and $\frac{\partial \boldsymbol{k}^{\mathrm{ini}}}{\partial \boldsymbol{y}} \equiv \mathbf{0}$ then the matrix on the right hand side of Eq. (3.14) has an upper right block triangular structure as indicated by the horizontal and vertical lines. In this case only the lower right block of this block triangular matrix requires further analysis. The most simple choice is to fix components of the state vector $\boldsymbol{x}(t_0)$ by equations $\boldsymbol{k}^{\mathrm{ini}}_{\mu}(t_0, \boldsymbol{x}(t_0), \boldsymbol{y}(t_0), \dot{\boldsymbol{x}}(t_0))$, $\mu = 1, \ldots, n_{\mathrm{ddf}}$, of the type

$$\boldsymbol{x}_{\nu(\mu)}(t_0) - \boldsymbol{x}_{0,\nu(\mu)} = 0\,, \quad (3.15)$$

where $\nu : \{1, \ldots, n_{\mathrm{ddf}}\} \to \{1, \ldots, n_{\boldsymbol{x}}\}$ is an injective mapping. The mapping ν has to be defined appropriately, i.e., in a way that the regularity condition

$$\det\left(\left[\begin{array}{c} \bar{\boldsymbol{s}}_{\boldsymbol{x}}(t, \boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}) \\ \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{k}^{\mathrm{ini}}(t, \boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}) \end{array}\right]_{t=t_0}\right) \neq 0 \quad (3.16)$$

holds. This condition can be easily checked.

c) The transition conditions connecting the stages (the restrictions of the dynamical system to the subintervals of the prediction horizon introduced by the control mesh) depend on the physical properties of the system being modelled. Similar to Eq. (3.15) transition conditions of the form

$$\boldsymbol{k}^{\mathrm{ini}}_{n}(t_n, \boldsymbol{x}^{+}(t_n), \boldsymbol{x}^{-}(t_n)) = 0\,; \quad n = 1, 2, \ldots \quad (3.17)$$

are chosen where $\boldsymbol{x}^{-}(t_n) = \lim_{t \nearrow t_n} \boldsymbol{x}(t)$ and $\boldsymbol{x}^{+}(t_n) = \lim_{t \searrow t_n} \boldsymbol{x}(t)$. E.g., continuity in a subset of the differential variables may be demanded (see the discussion in Section 3.1.13.a). Together with Eq. (3.12c) these transition conditions have to determine $\boldsymbol{x}^{+}(t_n)$ uniquely. Thus

$$\det\left(\left[\begin{array}{c} \bar{\boldsymbol{s}}_{\boldsymbol{x}}(t^{+}, \boldsymbol{x}^{+}(t), \boldsymbol{y}^{+}(t), \dot{\boldsymbol{x}}^{+}(t)) \\ \frac{\partial}{\partial \boldsymbol{x}^{+}} \boldsymbol{k}^{\mathrm{ini}}_{n}(t, \boldsymbol{x}^{+}(t), \boldsymbol{x}^{-}(t)) \end{array}\right]_{t=t_n}\right) \neq 0\,; \quad n = 1, 2, \ldots\,, \quad (3.18)$$

has to hold (cf. Eq. (3.16)) at each switching point t_n^+.

In Algorithm 4 we summarise the computation of a set of consistent initial conditions $\dot{\boldsymbol{x}}(t)$, $\boldsymbol{x}(t)$, and $\boldsymbol{y}(t)$ for semi-explicit index-2 DAEs based on the results discussed in this section. It can be used at the start ($t = t_0$) or at a switching point ($t = t_n^+$, $n = 1, 2, \dots$). For the latter case in step **4** the initial conditions $\boldsymbol{k}^{\text{ini}}(\cdot)$ have to be replaced by the transition conditions $\boldsymbol{k}_n^{\text{ini}}(\cdot)$.

Algorithm 4 (Consistent Initial Conditions, Index-2 DAE)

1. *Build the derivative array equations Eqs.* (3.10a)–(3.10c).

2. *Compute the Gaussian elimination* $[L]$ *according to Eq.* (3.11).

3. *Generate the (nonlinearly) transformed system of derivative array equations Eqs.* (3.12a)–(3.12c), *Eq.* (3.12f).

4. *Solve the square and full rank* reduced consistency equations *at* $t = t_0$ *(or, with minor modifications of notation, at* $t = t_n^+$, $n = 1, 2, \dots$*), which is set up using Eqs.* (3.12a)–(3.12c), *Eq.* (3.12f)

$$
\begin{aligned}
0 &= \bar{\boldsymbol{f}}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \,, \\
0 &= \bar{\boldsymbol{g}}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \,, \\
0 &= \bar{\boldsymbol{s}}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \,, \\
0 &= \bar{\boldsymbol{s}}_t(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) + \bar{\boldsymbol{s}}_{\boldsymbol{x}}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \cdot \dot{\boldsymbol{x}}(t) \,, \\
0 &= \boldsymbol{k}^{ini}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \,,
\end{aligned}
$$

   in the unknowns $\boldsymbol{x}(t)$, $\boldsymbol{y}(t)$, *and* $\dot{\boldsymbol{x}}(t)$. *Appropriately specified initial / transition conditions* $\boldsymbol{k}^{ini}$ *have to be available for each consistent initialisation problem, e.g., as specified in Eqs.* (3.15)–(3.16) *or Eqs.* (3.17)–(3.18).

5. *Optionally, solve Eqs.* (3.12d)–(3.12e), *Eq.* (3.12i) *for* $\ddot{\boldsymbol{x}}(t)$ *and* $\dot{\boldsymbol{y}}(t)$.

Core of Algorithm 4 is the calculation of the Gaussian elimination $[L]$ and of the corresponding partitioning of the algebraic variables in step **2** in order to reveal the hidden constraints Eq. (3.12f). This calculation is based on the numerical values of the Jacobian of the original DAE. Thus, the result $[L]$ can be sensitive to round-off errors in the Jacobian of the original DAE, and it directly depends on the estimated start values for $\boldsymbol{x}$, $\boldsymbol{y}$, and $\dot{\boldsymbol{x}}$. Additionally, the index of the DAE has to be known *a priori*.

3.1.5 The Algorithm of Pantelides

One of the major problems encountered in Section 3.1.1, Section 3.1.2, and Section 3.1.3 is that indiscriminate differentiation of the entire DAE in accordance with the definition of the derivative array equations (Definition 1.8 and Eq. (1.5)) in

general leads to a system of rank-deficient consistency equations. In order to solve such a system special, computationally expensive algorithms have to be employed.

The alternative approach is to determine subsets of equations in a DAE that need to be differentiated in order to obtain a minimal, full rank system of consistency equations. Given appropriately specified transition conditions this system can then be solved by standard numerical methods. We call this (full rank) system the *reduced consistency equations*. Accordingly, we introduce the *reduced derivative array equations* as the set of functions that are present in the reduced consistency equations as total time differentials of the original DAE equations.

In the previous Section 3.1.4 reduced consistency equations have been derived for the special case of a semi-explicit index-2 DAE (cf. Algorithm 4). The approach is based on numerical properties of the DAE (basically, on a Gaussian elimination) and requires a priori knowledge of the index. In contrast, [Pant 88a] proposes a technique which applies to a large class of DAE systems of practical interest. The *Algorithm of Pantelides* is a structurally oriented method which solely requires data that can be directly obtained from the original DAE. Therefore it is suitable for general, large-scale problems as arising in our industrial application.

Again, we distinguish differential variables $\boldsymbol{x}$ and algebraic variables $\boldsymbol{y}$, rewriting Eqs. (1.1a)–(1.1b) as

$$\begin{bmatrix} \boldsymbol{F}_1(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \\ \vdots \\ \boldsymbol{F}_{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t)) \end{bmatrix} = 0\,; \quad t \in [t_0, t_\mathrm{f}] \subset \mathbb{R}, \tag{3.19a}$$

$$\boldsymbol{k}^{\mathrm{ini}}(t_0, \boldsymbol{x}(t_0), \boldsymbol{y}(t_0), \dot{\boldsymbol{x}}(t_0)) = 0\,, \tag{3.19b}$$

where $\boldsymbol{F}_\mu : \mathbb{R}^{1+2n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}, \quad \mu = 1, \ldots, n_{\boldsymbol{x}}+n_{\boldsymbol{y}}$, and $\boldsymbol{k}^{\mathrm{ini}} : \mathbb{R}^{1+2n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{m_{\boldsymbol{k}^{\mathrm{ini}}}}$. The aim is to find a set of consistent initial conditions $[\boldsymbol{x}(t_0), \boldsymbol{y}(t_0), \dot{\boldsymbol{x}}(t_0)]$.

Remark 3.3:
In Section 2.4.3.a we have discussed the widely used BDF integrator method. When starting from lowest order, i.e., with an implicit Euler step, only in $\boldsymbol{x}$, $\boldsymbol{y}$, and $\dot{\boldsymbol{x}}$ consistent values need to be provided in order to allow for a smooth start of the integration while $\dot{\boldsymbol{y}}$ can be neglected in the first step. Moreover, $\dot{\boldsymbol{y}}$ may be obtained given consistent values for $\boldsymbol{x}$, $\boldsymbol{y}$, and $\dot{\boldsymbol{x}}$ and the information required to set up the reduced derivative array equations, cf., e.g., Eq. (4.42) in Section 4.3.3.a. ◊

[Pant 88a] starts with the observation that not every subset of equations of a DAE actually gives *new* constraints upon differentiation. In order to see this consider an arbitrary nonempty subset of $m_{\bar{\boldsymbol{F}}}$ equations of the DAE (3.19a)

$$\emptyset \neq \left\{\bar{\boldsymbol{F}}_1, \ldots, \bar{\boldsymbol{F}}_{m_{\bar{\boldsymbol{F}}}}\right\} \subset \left\{\boldsymbol{F}_1, \ldots, \boldsymbol{F}_{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}\right\},$$

containing a subset of the state variables

$$\bar{\boldsymbol{x}} := \{\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_{n_{\bar{\boldsymbol{x}}}}\} \subset \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_{\boldsymbol{x}}}\}\,, \quad \text{and}$$

$$\bar{\boldsymbol{z}} := \{\bar{\boldsymbol{z}}_1, \ldots, \bar{\boldsymbol{z}}_{n_{\bar{\boldsymbol{z}}}}\} \subset \left\{\dot{\boldsymbol{x}}_1, \ldots, \dot{\boldsymbol{x}}_{n_{\boldsymbol{x}}}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_{\boldsymbol{y}}}\right\} .$$

I.e., $\bar{\boldsymbol{z}}$ is the vector of the highest order time derivatives of all original state variables present. This set of functions defines a multi-dimensional function

$$\bar{\boldsymbol{F}}(t, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) := \left[\bar{\boldsymbol{F}}_1, \ldots, \bar{\boldsymbol{F}}_{m_{\bar{\boldsymbol{F}}}}\right]^T (t, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) ,$$

$\bar{\boldsymbol{F}} : \mathbb{R}^{1+n_{\bar{\boldsymbol{x}}}+n_{\bar{\boldsymbol{z}}}} \to \mathbb{R}^{m_{\bar{\boldsymbol{F}}}}$. In the sequel it is assumed that the Jacobian of $\bar{\boldsymbol{F}}$ at t_0 with respect to $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{z}}$ has full row rank $m_{\bar{\boldsymbol{F}}}$:

$$\operatorname{rank}\left(\left[\frac{\partial \bar{\boldsymbol{F}}}{\partial \bar{\boldsymbol{x}}} \quad \frac{\partial \bar{\boldsymbol{F}}}{\partial \bar{\boldsymbol{z}}}\right]_{\substack{t_0, \\ \bar{\boldsymbol{x}}(t_0), \bar{\boldsymbol{z}}(t_0)}}\right) = m_{\bar{\boldsymbol{F}}} .$$

Total differentiation of $\bar{\boldsymbol{F}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}, t)$ with respect to time t not only gives a set of new equations, but also introduces new variables $\dot{\bar{\boldsymbol{z}}}$ that have not been present in the DAE Eq. (3.19a) before

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{\boldsymbol{F}}(t, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) = \frac{\partial}{\partial t}\bar{\boldsymbol{F}}(t, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) + \frac{\partial}{\partial \bar{\boldsymbol{x}}}\bar{\boldsymbol{F}}(t, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) \cdot \dot{\bar{\boldsymbol{x}}} + \frac{\partial}{\partial \bar{\boldsymbol{z}}}\bar{\boldsymbol{F}}(t, \bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) \cdot \dot{\bar{\boldsymbol{z}}} = 0 ; \quad (3.20)$$

the equality to zero is due to Eq. (3.20) being part of the derivative array equations of the DAE Eq. (3.19a). But if

$$\operatorname{rank}\left(\left[\frac{\partial \bar{\boldsymbol{F}}}{\partial \bar{\boldsymbol{z}}}\right]_{\substack{t_0, \\ \bar{\boldsymbol{x}}(t_0), \bar{\boldsymbol{z}}(t_0)}}\right) = m_{\bar{\boldsymbol{F}}} ,$$

then the new equations can be satisfied for any choice of $\{\boldsymbol{x}(t_0), \boldsymbol{y}(t_0), \dot{\boldsymbol{x}}(t_0)\}$ as by the implicit function theorem it is possible to solve for a corresponding value of $\dot{\bar{\boldsymbol{z}}}$. Thus no additional constraints on $\bar{\boldsymbol{x}}$ or $\bar{\boldsymbol{z}}$ have been created. However, it is exactly the additional constraints we are looking for as they set up the hidden constraints. In other words, detection of hidden constraints means to locate all subsets of equations that impose *new* constraints upon differentiation. According to the previous considerations this is equivalent to the determination of all subsets of equations $\bar{\boldsymbol{F}}$ satisfying

$$\operatorname{rank}\left(\left[\frac{\partial \bar{\boldsymbol{F}}}{\partial \bar{\boldsymbol{z}}}\right]_{\substack{t_0, \\ \bar{\boldsymbol{x}}(t_0), \bar{\boldsymbol{z}}(t_0)}}\right) < m_{\bar{\boldsymbol{F}}} . \quad (3.21)$$

The difficulty inherent in this criterion is that *all* nonempty subsets $\bar{\boldsymbol{F}}$ of the DAE have to be examined, which is a combinatorial problem. Thus the criterion of Eq. (3.21) cannot be applied in practice (apart from small scale DAE systems).

At this point the idea of [Pant 88a] is to employ a *necessary* condition for a set of equations to have property Eq. (3.21) as a (weak) substitute. This necessary

condition is

$$n_{\bar{\boldsymbol{z}}} < m_{\bar{\boldsymbol{F}}}\,, \tag{3.22}$$

as

$$\operatorname{rank}\left(\left[\frac{\partial \bar{\boldsymbol{F}}}{\partial \bar{\boldsymbol{z}}}\right]_{\substack{t_0,\\ \bar{\boldsymbol{x}}(t_0),\bar{\boldsymbol{z}}(t_0)}}\right) \leq n_{\bar{\boldsymbol{z}}} \quad \overset{n_{\bar{\boldsymbol{z}}}<m_{\bar{\boldsymbol{F}}}}{\Rightarrow} \quad \operatorname{rank}\left(\left[\frac{\partial \bar{\boldsymbol{F}}}{\partial \bar{\boldsymbol{z}}}\right]_{\substack{t_0,\\ \bar{\boldsymbol{x}}(t_0),\bar{\boldsymbol{z}}(t_0)}}\right) < m_{\bar{\boldsymbol{F}}}\,,$$

i.e., the number of the variables which occur in their highest order time derivative in a set of functionally independent equations has to be smaller than the number of equations considered. Eq. (3.22) gives rise to [Pant 88a]

Definition 3.1 (Structural Singularity of Sets of Equations)

- *A (functionally independent) subset of equations $\bar{\boldsymbol{F}}$ that satisfies criterion Eq.* (3.22) *is called* structurally singular with respect to the variable subset $\bar{\boldsymbol{z}}$ *($\bar{\boldsymbol{F}}$ and $\bar{\boldsymbol{z}}$ as defined above).*
- *A structurally singular subset of equations is called* minimally structurally singular *(MSS) if none of its proper subsets is structurally singular.*
- *A system of equations is called* structurally singular with respect to a certain set of variables *if it contains a structurally singular subset with respect to this set of variables.*

The advantage of criterion Eq. (3.22) is that it allows the application of efficient algorithms originating from combinatorics. Basis is

Theorem 3.3 (Systems of Distinct Representatives)
Let $V = \{V_1, V_2, \ldots, V_m\}$ be a set of objects and $S = \{S_1, S_2, \ldots, S_n\}$ a set of subsets of V.

Then each element of S can be assigned a different element of V if and only if every collection of $k(\leq n)$ elements of S contains at least k distinct elements of V.

In the context of the consistent initialisation problem we have the following interpretation of V and S:

- Let $\boldsymbol{z} := [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}]^T := [\dot{\boldsymbol{x}}_1, \ldots, \dot{\boldsymbol{x}}_{n_{\boldsymbol{x}}}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_{\boldsymbol{y}}}]^T$. The variables $\boldsymbol{z}_\nu$ are the members $V_\nu \in V$, $\nu = 1, \ldots, n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$.
- Each equation $\boldsymbol{F}_\mu$, $\mu = 1, \ldots, n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$, is associated with a set $S_\mu \in S$ containing the elements of V assigned to the variables occurring in this equation.

In this nomenclature criterion Eq. (3.22) defining minimally structurally singular sets of equations is satisfied for all subsets of those elements in S that do *not* fulfil the necessary and sufficient condition for an *assignment* stated in Theorem 3.3.

Remark 3.4:
The variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_x}$ do not enter the investigations. ◊

Assignments are advantageously *constructed* by graph-theoretical methods. Here, such an algorithm is employed for the detection of minimally singular subsets. At first, a bipartite graph consisting of *E-nodes* and *V-nodes* is defined by the following identification:

- E-node $\mu \leftrightarrow S_\mu \leftrightarrow \boldsymbol{F}_\mu$,
- V-node $\nu \leftrightarrow V_\nu \leftrightarrow \boldsymbol{z}_\nu$,
- edge (μ, ν) exists if $V_\nu \in S_\mu$, i.e, if equation μ contains variable ν.

For the further discussion more concepts from graph-theory are required, cf., e.g. [Pant 88a]:

Definition 3.2 ((Partial/Complete) Assignment, Augmenting Path)

- *An* assignment *is a set of edges (μ, ν) such that no node μ or ν appears in more than one edge in the set.*
- *Edges in an assignment are called* matching edges.
- *A node is* exposed *if it does not appear in any matching edge.*
- *An* augmenting path *is a path with exposed nodes at both ends and alternating nonmatching and matching edges between them.*
- *An assignment is* complete *if it leaves no E-node exposed; otherwise it is called a* partial assignment.

A complete assignment can be constructed starting from a partial assignment (trivially the empty set) and then successively adding new edges to the partial assignment by the construction of an augmenting path from an exposed E-node to an exposed V-node. If the search for an augmenting path is successful then the new assignment can be obtained by *reassignment*. In the reassignment the matching edges in the augmenting path are set as the nonmatching edges of the new assignment. The matching edges in the new assignment are given by the nonmatching edges in the augmenting path. In this way the assignment is enhanced by an additional edge. An efficient algorithm for the construction of augmenting paths has been proposed by [Duff 81]. We will refer to this algorithm as AUGMENTPATH.

Remark 3.5:
Algorithms for obtaining a maximum transversal [Duff 81] are a common tool for the treatment of sparse matrices. An implementation of AUGMENTPATH is found in the HSL routine MC21 [AEA 93]. ◊

The connection between (in)complete assignments and minimally singular subsets is given by [Pant 88a]:

Lemma 3.2 (Augmenting Paths & Minimally Structurally Singular Subsets)
If AUGMENTPATH *cannot find an augmenting path, then the set of equations represented by all E-nodes visited in the search is structurally singular with respect to all variables* $\{\dot{\boldsymbol{x}}_1, \ldots, \dot{\boldsymbol{x}}_{n_{\boldsymbol{x}}}\}$ *and* $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_{\boldsymbol{y}}}\}$ *contained in that set of equations.*

Lemma 3.2 provides the basis for Pantelides' Algorithm. Given a procedure AUGMENTPATH for the construction of augmenting paths the entire algorithm can be formulated as in Algorithm 5.

Algorithm 5 (Algorithm of Pantelides (outline))

1. *Make initialisations.*
2. *Start with an empty assignment.*
3. *Loop over all E-nodes (i.e., equations):*
   (a) *Delete all V nodes (and their edges) corresponding to variables which total time derivatives are present in the* current *system of equations represented by the E- and V-nodes.*
   (b) *Search for an augmenting path emanating from the actual E-node.*
   (c) *If* AUGMENTPATH *failed, then*
      i. *Differentiate all equations with respect to time that have been visited during the search for an augmenting path.*
      ii. *Add the* new *equations and time derivatives of variables introduced in step* ***3(c)i*** *to the* current *system of equations, i.e., add E-nodes, V-nodes, and edges.*
      iii. *Set that E-node as actual E-node which was generated by differentiation of the actual E-node.*
      iv. *Continue with* ***3a****.*

In the course of Algorithm 5 all equations of the reduced derivative array equations are examined as root of an augmenting path. In this way all minimally structurally subsets of equations can be detected. Step **3(c)iii** initiates a *depth-first search.* Termination of Algorithm 5 is not given a priori due to the loop **3a** — **3(c)iv** which is only left by condition **3c** if an augmenting path is found in step **3b**. Indeed, termination of the algorithm can be guaranteed for well-posed DAEs only:

Theorem 3.4 (Termination of the Algorithm of Pantelides)
The Algorithm of Pantelides terminates if and only if the reduced derivative array equations are structurally nonsingular. Furthermore, if the reduced derivative array equations are structurally singular, then the DAE system examined is structurally inconsistent.

Structural inconsistency of a DAE is specified in the following Definition 3.3.

Definition 3.3 (Structurally Inconsistent DAE)
A DAE system Eq. (3.19a) *is said to be* structurally inconsistent *if it can become structurally singular with respect to* all *occurring variables by the addition of the time differentials of a (possibly empty) subset of its equations.*

An important feature of the algorithm as proposed by Pantelides is that step **3(c)i** is *not* executed symbolically. Instead, *structurally nonlinear differentiation* is employed (cf. Definition 1.19). Consequently, the only input required is the pattern of the Jacobian of the DAE system Eq. (3.19a)

$$\operatorname{pat}\left(\left[\begin{array}{ccc} \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_1}{\partial \dot{\boldsymbol{x}}} \\ \vdots & \vdots & \vdots \\ \frac{\partial \boldsymbol{F}_{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}}{\partial \dot{\boldsymbol{x}}} \end{array}\right]_{\substack{t_0,\\ \boldsymbol{x}(t_0),\boldsymbol{y}(t_0),\dot{\boldsymbol{x}}(t_0)}}\right),$$

which defines the edges in the initial bipartite graph.

Remark 3.6:
Strictly speaking only the Jacobian with respect to $\dot{\boldsymbol{x}}$ and $\boldsymbol{y}$ is required. All V-nodes (and their incident edges) corresponding to variables $\boldsymbol{x}$ are deleted in step **3a** before the search for an augmenting path is initiated for the first time. See also Remark 3.4. ◊

In the sequel $\boldsymbol{f}_\mu$ is used as a symbol for any equation of the original DAE or any of the additional equations created in the course of the algorithm. Analogously, $\boldsymbol{\xi}_\nu$ is a symbol for any variable in the original system or for any new variable arising. Thus initially $\boldsymbol{f}_\mu \equiv \boldsymbol{F}_\mu$, $\mu = 1, \ldots, n_{\boldsymbol{x}} + n_{\boldsymbol{y}} =: m_{\boldsymbol{f}}$ and $\boldsymbol{\xi} := [\boldsymbol{x}^T, \boldsymbol{y}^T, \dot{\boldsymbol{x}}^T]^T$, $n_{\boldsymbol{\xi}} = 2n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$ holds. Apart from the bipartite graph describing the occurrence of the highest order derivatives of the variables in the derivative array equations, Pantelides' Algorithm uses an equation association list $\mathtt{EAL} \in \mathbb{N}^{m_{\boldsymbol{f}}}$ and a variable association list $\mathtt{VAL} \in \mathbb{N}^{m_{\boldsymbol{\xi}}}$. The μ^{th} element of these vectors is defined as

$$\mathtt{EAL}_{(\mu)} = \begin{cases} \nu\,; & \text{if } \boldsymbol{f}_\nu = \frac{\mathrm{d}\boldsymbol{f}_\mu}{\mathrm{d}t}\,, \\ 0\,; & \text{otherwise}, \end{cases} \qquad \text{and} \qquad \mathtt{VAL}_{(\mu)} = \begin{cases} \nu\,; & \text{if } \boldsymbol{\xi}_\nu = \frac{\mathrm{d}\boldsymbol{\xi}_\mu}{\mathrm{d}t}\,, \\ 0\,; & \text{otherwise}, \end{cases}$$

pointing to the corresponding derived equation or variable, respectively. In the course of the algorithm $m_{\boldsymbol{f}}$ and $n_{\boldsymbol{\xi}}$ increase. Upon termination Pantelides' Algorithm returns in `EAL` the information required to set up the reduced system of derivative array equations together with its size $m_{\boldsymbol{f}}$. Furthermore, `VAL` contains the variables present in this system and their differential dependencies, and the number of all variables $n_{\boldsymbol{\xi}}$ in the reduced derivative array equations is available.

In common there are more variables than equations in the reduced derivative equations, i.e., they constitute an underdetermined system of equations with $n_{\text{ddf}} = n_{\boldsymbol{\xi}} - m_{\boldsymbol{f}}$ dynamic degrees of freedom (cf. Definition 1.14). Unfortunately, Pantelides' Algorithm *cannot determine* which subset of n_{ddf} of the variables in the reduced system of derivative array equations may be specified by additional n_{ddf} equations in order to arrive at a square system of consistency equations. For

a discussion of this topic we refer to Section 3.1.13.

Pantelides' Algorithm has been designed for the detection of a reduced system of derivative array equations that can be used for the computation of consistent initial conditions $[\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}]$ for the DAE Eqs. (3.19a)–(3.19b). With some care taken for special cases Pantelides' Algorithm can also be used to determine the structural index (cf. Definition 1.20) of the DAE [Unge 90]. From the theoretical point of view we are in general interested in the differential index of the DAE. Again some care is required if Pantelides' Algorithm is employed to assess the differential index, as there are cases in which the structural (differential) index and the differential index differ (cf. the discussion in Section 1.2).

Remark 3.7:
[Otte 95], [Otte 96] obtains the *differential-algebraic index* by a slight modification of the definition of the differential index. Especially, he considers $\boldsymbol{x}$, $\boldsymbol{y}$, and $\dot{\boldsymbol{x}}$ instead of $\boldsymbol{x}$, $\boldsymbol{y}$, $\dot{\boldsymbol{x}}$, and $\dot{\boldsymbol{y}}$ (cf. Definition 1.7). Thus by construction Pantelides' Algorithm is more directly related to the differential-algebraic index than to the differential index. However, the differential-algebraic index suffers from theoretical shortcomings that limit its applicability, e.g., the differential-algebraic index is not invariant against state transformations [Otte 02]. ◊

Pantelides' Algorithm has been implemented in the FORTRAN code PALG [Unge 90], [UnMa 91]. Its sparse version SPALG [UKM 95] is especially suited for the analysis of the very large-scale DAEs which are the subject of our interest. Indeed, SPALG is a core ingredient of our algorithm for consistent initialisation (cf. Section 3.2.1). The extensions PALGU and SPALGU additionally trace the derivatives of external inputs (controls) in the reduced derivative array equations [Krön 02].

3.1.6 Dummy Derivatives

The structural information obtained from Pantelides' Algorithm (cf. Section 3.1.5) can be used in two different ways for the construction of a DAE with an index of at most one, starting from a DAE with higher (structural) index [MaSö 93]:

1. Replace each equation by its highest order total derivative with respect to time as requested from Pantelides' Algorithm. By construction the resulting system is a determined system of index less or equal one (in the index-0 case an UODE of the original DAE has been constructed). This approach is proposed in the original paper [Pant 88a].

   The main drawback of this method is that during an integration the replaced algebraic constraints are only satisfied implicitly via their time derivatives. Therefore the numerical integration of the new system in general suffers from *drift* effects which are inevitably introduced by discretisation round-off errors. The problem is that these round-off errors lead to a gradual deviation from the solution manifold of the original DAE. In order to avoid such effects

stabilisation techniques or projection methods can to be applied, cf., e.g., [CaMo 95].

2. Use the entire system of reduced derivative array equations as new index ≤ 1 DAE, i.e., in contrast to approach **1** keep all lower time derivatives of all equations. In this way the index reduced DAE explicitly contains all constraints of the original DAE.

   However, the reduced system of derivative array equations forms an overdetermined dynamical system. Therefore special projection techniques are required for their numerical integration.

In order to combine the advantages of both techniques, i.e., determinacy and preservation of the solution manifold, [MaSö 92], [MaSö 93], and independently [CeEl 93] introduce the dummy derivative method. Similar to method **2** above in a first step Pantelides' Algorithm is employed in order to investigate the reduced derivative array equations. In a second step a subset of the time derivatives of state variables is substituted by (or reinterpreted as) *dummy algebraic variables*, resulting in a determined system. In practice the selection of the dummy derivatives has to be monitored and adapted dynamically during the course of an integration (*dummy derivative pivoting*).

From the consistent initialisation point of view the index-1 DAE derived by the dummy derivative method is equivalent to the original reduced system of derivative array equations, apart from two minor differences:

- [MaSö 93] check for unnecessary differentiations demanded from Pantelides' Algorithm that lead to index-0 systems in some cases.
- Some of the former state derivatives become algebraic variables.

As these differences are not relevant in our context we skip a more detailed discussion of the dummy derivative algorithm.

[Feeh 98] applies the method of *dummy derivatives* as a means for deriving index-1 DAEs in order to integrate higher index DAEs. The availability of the index reduced system further allows him to apply his algorithm for the transfer of sensitivities at discontinuities in index-1 DAEs also in the context of higher index DAEs (cf. Section 4.1.4.a). In his implementation into the simulation and optimisation package ABACUSS [Feeh 98] executes the necessary differentiations *symbolically* using automatic differentiation techniques.

3.1.7 Hybrid Methods

The generation of the derivative array equations is a major obstacle for every algorithm based on the solution of the consistency equations unless the higher order time derivatives can be obtained symbolically. As we have seen in Section 3.1.3 the technique proposed by [LPG 91] is suitable for the numerical approximation of

the higher order time derivatives in a very general setting. However, this special approach exhibits some drawbacks:

1. The index of the DAE has to be known.
2. The user has to provide *appropriate* transition conditions.
3. In general, the solution of the resulting overdetermined rank-deficient nonlinear system of equations is difficult and numerically expensive.

As a partial remedy [KMG 92] propose to analyse the *structural* properties of a DAE by Pantelides' Algorithm at first. Based on this structural information a *square* system of reduced consistency equations is constructed that can be solved for consistent initial conditions. Similar to [LPG 91] the numerical approximation of the total time derivatives is performed by the formulae of [Leim 88] (cf. Section 3.1.3). Thus the combined structural/numerical approach solves problems **1** and **3** which are present in [LPG 91].

[KMG 92] have tested this *hybrid technique* on smaller index-2 and index-3 systems, in [Krön 02] the application to smaller index-1 and index-2 systems is discussed.

In Section 3.2 we develop a tailored algorithm for the consistent initialisation of large scale semi-explicit index-2 DAEs extending this technique. Especially, in Section 3.2.4 we develop an algorithm which automatically generates appropriate transition conditions addressing problem **2** noted above.

3.1.8 Elimination and Substitution Methods

In Section 1.1 we have introduced Algorithm 1 which computes the differential index of a DAE. On its way the UODE of the DAE is generated as well as the hidden constraints. Using this information the corresponding extended system of the DAE can be built. Together with an appropriate set of transition conditions consistent initial conditions can then be computed according to Definition 1.11 [UKM 95]. This approach originates from a method proposed by [Gear 88]. [BBMP 90] interpret the method in a different way and develop an algorithm for linear constant coefficient DAEs. [Moe 95] extends the algorithm of [BBMP 90] to the nonlinear case. In parallel to [BBMP 90] in [Unge 90], [UnMa 91], [UKM 95] an algorithm for the analysis of linear-implicit DAEs based on Gear's approach is developed. This algorithm is further extended in [KMG 97].

After the UODE of the DAE and the hidden equations have been generated, [Gear 88] considers the corresponding extended system as the new dynamical system in order to avoid the otherwise inevitable drift effects that make integration of the plain UODE prohibitive (but: the UODE contains the full *dynamics* of the original DAE and thus is algebraically equivalent to it; drift is a purely numerical effect). The drawback is in the overdeterminacy of the corresponding extended system (as a dynamical system). Therefore, [Gear 88] introduces additional slack

variables. In contrast, [BBMP 90] successively replace differential equations by the hidden constraints finally ending up with an index-1 DAE. This system is equivalent to the original DAE with the same consistent initial conditions and the same number of dynamic degrees of freedom.

In order to demonstrate the fundamental difference of this approach to Pantelides' Algorithm we consider Algorithm 6 as given in [UKM 95]:

Algorithm 6 (Index Reduction: Elimination and Substitution)
Let there be a DAE $\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t))$, $t \in [t_0, t_f] \subset \mathbb{R}$, *with* $\boldsymbol{\xi} \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$ *and* $\boldsymbol{F} : \mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{\xi}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$.

1. *Initialise*
   $i := 0$,
   $\hat{\boldsymbol{F}}^{(0)} := \boldsymbol{F}$, $\tilde{\boldsymbol{\xi}}^{(0)} := \boldsymbol{\xi}$, $n_{\boldsymbol{\xi}}^{(0)} := n_{\boldsymbol{\xi}}$, $n_{ddf} := n_{\boldsymbol{\xi}}$,
   $\boldsymbol{X}^{(-1)} := \emptyset$, $\boldsymbol{\mathcal{F}}^{(-1)} := \emptyset$.

2. *Determine the size of the subspace carrying the system dynamics*
   $\mu^{(i)} := \operatorname{rank}\left(\left[\frac{\partial \hat{\boldsymbol{F}}^{(i)}(t, \boldsymbol{\xi}, \dot{\tilde{\boldsymbol{\xi}}}^{(i)})}{\partial \dot{\tilde{\boldsymbol{\xi}}}^{(i)}}\right]\right)$, $\tilde{\boldsymbol{\xi}}^{(i)} \in \mathbb{R}^{n_{\boldsymbol{\xi}}^{(i)}}$,
   and adapt the counter for the number of dynamic degrees of freedom
   $n_{ddf} \to n_{ddf} - (n_{\boldsymbol{\xi}}^{(i)} - \mu^{(i)})$.

3. *Partition*
   $$\begin{pmatrix} \hat{\boldsymbol{f}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)}) \\ \hat{\boldsymbol{g}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)}) \end{pmatrix} = P \cdot \hat{\boldsymbol{F}}^{(i)}(t, \boldsymbol{\xi}, \dot{\tilde{\boldsymbol{\xi}}}^{(i)}),\ \{\dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)}\} = \{\dot{\tilde{\boldsymbol{\xi}}}^{(i)}\},$$
   such that
   $$\det\left(\left[\frac{\partial \hat{\boldsymbol{f}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)})}{\partial \dot{\boldsymbol{x}}^{(i)}}\right]\right) \neq 0,$$
   with $\dot{\boldsymbol{x}}^{(i)} \in \mathbb{R}^{\mu^{(i)}}$, $\dot{\boldsymbol{y}}^{(i)} \in \mathbb{R}^{n_{\boldsymbol{\xi}}^{(i)} - \mu^{(i)}}$, $\hat{\boldsymbol{f}}^{(i)} \in \mathbb{R}^{\mu^{(i)}}$, $\hat{\boldsymbol{g}}^{(i)} \in \mathbb{R}^{n_{\boldsymbol{\xi}}^{(i)} - \mu^{(i)}}$, *and the permutation matrix* $P \in \mathbb{R}^{n_{\boldsymbol{\xi}}^{(i)} \times n_{\boldsymbol{\xi}}^{(i)}}$.

4. *Obtain explicit expressions for* $\dot{\boldsymbol{x}}^{(i)}$
   by solving $\hat{\boldsymbol{f}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)})$ *for* $\dot{\boldsymbol{x}}^{(i)} = \phi^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{y}}^{(i)})$,
   and append this expression to the collection of UODEs
   $\dot{\boldsymbol{X}}^{(i-1)}(t) := \left[\dot{\boldsymbol{x}}^{(0)}, \ldots, \dot{\boldsymbol{x}}^{(i-1)}\right]^T(t) = \boldsymbol{\mathcal{F}}^{(i-1)}(t, \boldsymbol{\xi}, \dot{\tilde{\boldsymbol{\xi}}}^{(i)})$,
   to obtain
   $\dot{\boldsymbol{X}}^{(i)}(t) := \left[\dot{\boldsymbol{x}}^{(0)}, \ldots, \dot{\boldsymbol{x}}^{(i-1)}, \dot{\boldsymbol{x}}^{(i)}\right]^T(t) = \tilde{\boldsymbol{\mathcal{F}}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)})$.

5. *Generate the new collection of UODEs by substitution of*
   $\dot{\boldsymbol{x}}^{(i)} = \phi^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{y}}^{(i)})$ *into* $\tilde{\boldsymbol{\mathcal{F}}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)})$,
   leading to
   $\dot{\boldsymbol{X}}^{(i)}(t) := \left[\dot{\boldsymbol{x}}^{(0)}, \ldots, \dot{\boldsymbol{x}}^{(i-1)}, \dot{\boldsymbol{x}}^{(i)}\right]^T(t) = \boldsymbol{\mathcal{F}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{y}}^{(i)})$.

6. *If* $\mu^{(i)} = n_{\boldsymbol{\xi}}^{(i)}$ *then terminate as both UODE and the hidden equations have been completely detected. The differential index of the DAE is* $\iota_d = i$.

7. *In order to obtain the hidden equations substitute*
   $\dot{\boldsymbol{x}}^{(i)} = \phi^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{y}}^{(i)})$ *into* $\hat{\boldsymbol{g}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{x}}^{(i)}, \dot{\boldsymbol{y}}^{(i)})$,
   giving[(i)]
   $\boldsymbol{\mathcal{G}}^{(i)}(t, \boldsymbol{\xi}) = 0$.

8. *Differentiate the new hidden equations* $\boldsymbol{\mathcal{G}}^{(i)}(t, \boldsymbol{\xi}) = 0$ *with respect to time, giving*
   $0 := \frac{\partial \boldsymbol{\mathcal{G}}^{(i)}(t,\boldsymbol{\xi})}{\partial t} + \frac{\partial \boldsymbol{\mathcal{G}}^{(i)}(t,\boldsymbol{\xi})}{\partial \boldsymbol{X}^{(i)}} \dot{\boldsymbol{X}}^{(i)} + \frac{\partial \boldsymbol{\mathcal{G}}^{(i)}(t,\boldsymbol{\xi})}{\partial \boldsymbol{y}^{(i)}} \dot{\boldsymbol{y}}^{(i)}$.

9. *Eliminate the already known differentials* $\dot{\boldsymbol{X}}^{(i)}$ *by substituting the UODEs* $\dot{\boldsymbol{X}}^{(i)}(t) = \boldsymbol{\mathcal{F}}^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{y}}^{(i)})$ *into the result of step* **8**. *This gives a new set of DAEs*
   $\hat{\boldsymbol{F}}^{(i+1)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{y}}^{(i)}) = 0$.

10. *Prepare the next loop by*
    $\tilde{\boldsymbol{\xi}}^{(i+1)} := \boldsymbol{y}^{(i)}$, $n_{\boldsymbol{\xi}}^{(i+1)} := n_{\boldsymbol{\xi}}^{(i)} - \mu^{(i)}$, $i \to i+1$.

11. *Restart from step* **2**.

Remark 3.8:
After termination of Algorithm 6 the hidden equations are given by the equations

$$0 = \boldsymbol{\mathcal{G}}(t, \boldsymbol{\xi}) := \left[\boldsymbol{\mathcal{G}}^{(0)}, \ldots, \boldsymbol{\mathcal{G}}^{(\iota_d - 1)}\right]^T (t, \boldsymbol{\xi}) .$$

◊

In difference to Pantelides' Algorithm (Algorithm 5) Gear's Approach explicitly generates all UODEs and hidden equations. On the one hand, it solves for the differential variables (steps **2**–**4**) and eliminates the known time differentials in order reveal the UODEs (step **5**) as well as the hidden equations (step **7**). On the other hand, it eliminates the known time differentials in the first total derivatives with respect to time of the hidden equations (step **9**) thus avoiding the generation of higher order time derivatives of the state variables. In the general (nonlinear) case step **4** can lead to difficulties in practical application as it is based on the implicit function theorem which holds only locally. However, if the DAE is of linear implicit type (as is the case with the chemical engineering applications of our interest) step **4** reduces to the inversion of a regular matrix which can always be performed [UKM 95].

[(i)] *Substitution of* $\dot{\boldsymbol{x}}^{(i)}$ *in* $\hat{\boldsymbol{g}}^{(i)}$ *results in* $0 = \hat{\boldsymbol{g}}^{(i)}(t, \boldsymbol{\xi}, \phi^{(i)}(t, \boldsymbol{\xi}, \dot{\boldsymbol{y}}^{(i)}), \dot{\boldsymbol{y}}^{(i)})$. *However it can be shown that this equation does not explicitly depend on* $\dot{\boldsymbol{y}}^{(i)}$ *([Unge 90], Proposition 2.1.1), i.e.,* $\partial \boldsymbol{\mathcal{G}}^{(i)} / \partial \dot{\boldsymbol{y}}^{(i)} \equiv 0$.

In [Unge 90], [UnMa 91], [UKM 95] the implementation of a structural version of Algorithm 6 (ALGO) suitable for linear-implicit DAEs is described. This approach overcomes the difficulties connected to symbolical/numerical rank determination and construction of a suitable partitioning in steps **2**–**3**, at the price of the limitations of structural calculus discussed in Section 1.2. An interesting point is that – although Algorithm 5 and Algorithm 6 are conceptually different – it can be shown that their structural counterparts SPALG (or PALG) and ALGO return the same (structural) index as well as the same number of degrees of freedom if provided with the same data. In [KMG 97] ALGO is further extended for non-autonomous DAEs. Their algorithm ALGOU provides additional information about the derivatives of the external inputs (controls) present in the corresponding extended system.

A drawback of Gear's approach is that a sparse version of ALGO is possible in principle, but difficult to implement [UKM 95]. However, in our opinion a sparse version of ALGO may not be suitable for application to practical problems as in the elimination and substitution steps sparsity is in general not preserved. Pantelides' Algorithm can be implemented in sparse form without fundamental difficulties (SPALG, [UKM 95]). Thus in the context of our application Pantelides' Algorithm is the method of choice as large-scale systems of DAEs have to be examined.

3.1.9 Projector Based Techniques

Based on the notion of the *tractability index* ([März 89], [März 90], [März 92], [März 97]) several authors have developed methods for the consistent initialisation of index-2 tractable DAEs, see, e.g., [Hans 92], [Lamo 97], [EsLa 99], [Este 00]. In the sequel we follow [EsLa 99] who consider quasi-linear index-2 DAEs

$$\boldsymbol{F}(t,\boldsymbol{\xi}(t),\dot{\boldsymbol{\xi}}(t)) := A(t,\boldsymbol{\xi}(t))\cdot\dot{\boldsymbol{\xi}}(t) + \boldsymbol{f}(t,\boldsymbol{\xi}(t)) = 0\,; \quad t\in[t_0,t_\mathrm{f}]\,, \tag{3.23a}$$

$$P_\mathrm{ini}(t_0,\boldsymbol{\xi}(t_0))\cdot[\boldsymbol{\xi}(t_0)-\boldsymbol{\xi}_0] = 0\,, \tag{3.23b}$$

where $A : \mathbb{R}^{1+n_\xi} \to \mathbb{R}^{n_\xi\times n_\xi}$, and $\boldsymbol{f} : \mathbb{R}^{1+n_\xi} \to \mathbb{R}^{n_\xi}$. $P_\mathrm{ini}(t,\boldsymbol{\xi}) \in \mathbb{R}^{n_\xi\times n_\xi}$ is a projector selecting those state variables $\boldsymbol{\xi}$ which are chosen as the dynamic degrees of freedom; it will be defined below. $\boldsymbol{\xi}_0 \in \mathbb{R}^{n_\xi}$ are appropriate user given transition conditions.

To start with, we define the matrices

$$A_0(t,\boldsymbol{\xi}) := \frac{\partial}{\partial\dot{\boldsymbol{\xi}}}\boldsymbol{F}(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) = A(t,\boldsymbol{\xi})\,,$$

$$B_0(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) := \frac{\partial}{\partial\boldsymbol{\xi}}\boldsymbol{F}(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) = \frac{\partial}{\partial\boldsymbol{\xi}}\left[A(t,\boldsymbol{\xi})\dot{\boldsymbol{\xi}}\right] + \frac{\partial}{\partial\boldsymbol{\xi}}\boldsymbol{f}(t,\boldsymbol{\xi})\,.$$

In the DAE case given the coefficient matrix $A_0(t,\boldsymbol{\xi})(= A(t,\boldsymbol{\xi}))$ is singular. Addi-

tionally the following assumptions on A_0 are made:

$$\ker\left(A_0(t,\boldsymbol{\xi})\right) = \text{constant}\,, \quad \operatorname{im}\left(A_0(t,\boldsymbol{\xi})\right) = \text{constant}\,. \tag{3.24}$$

Now define the space $N_0 := \ker\left(A_0(t,\boldsymbol{\xi})\right)$, let Q_0 be a (constant) projector on N_0

$$\forall \bar{\boldsymbol{\xi}} \in \mathbb{R}^{n_\xi} : A_0(t,\boldsymbol{\xi})Q_0\bar{\boldsymbol{\xi}} = 0\,, \quad [Q_0]^2 = Q_0\,,$$

let $P_0 := \boldsymbol{Id} - Q_0$, and let W_0 be a constant projector along $\operatorname{im}\left(A_0(\boldsymbol{\xi},t)\right)$

$$\forall \bar{\boldsymbol{\xi}} \in \operatorname{im}\left(A_0(t,\boldsymbol{\xi})\right) : W_0\bar{\boldsymbol{\xi}} = 0\,, \quad [W_0]^2 = W_0\,.$$

I.e., P_0 filters the differential variables, whereas W_0 if multiplied from the left gives the algebraic constraints, nullifying all differential contributions in Eq. (3.23a). Furthermore, the space

$$\begin{aligned} S_0(t,\boldsymbol{\xi}) &:= \left\{ \bar{\boldsymbol{\xi}} \in \mathbb{R}^{n_\xi} : W_0 \frac{\partial}{\partial \boldsymbol{\xi}} \boldsymbol{f}(t,\boldsymbol{\xi})\bar{\boldsymbol{\xi}} = 0 \right\} \\ &= \left\{ \bar{\boldsymbol{\xi}} \in \mathbb{R}^{n_\xi} : W_0 B_0(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})\bar{\boldsymbol{\xi}} = 0 \right\} \end{aligned}$$

is introduced. With these prerequisites at hand index-1 tractability can be defined in adaptation of [EsLa 99]:

Definition 3.4 (Index-1 Tractability)
If $A_0(t,\boldsymbol{\xi})$ is singular, then Eq. (3.23a) *has* tractability index-1 *if and only if*

$$N_0 \cap S_0(t,\boldsymbol{\xi}) = \{0\}\,,$$

or, equivalently, if and only if

$$A_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) := A_0(t,\boldsymbol{\xi}) + B_0(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})Q_0 \textit{ is nonsingular}.$$

Suppose that both $A_0(t,\boldsymbol{\xi})$ and $A_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})$ are singular, i.e., Eq. (3.23a) does not fulfil the conditions for index-1 tractability. In this case, define

$$B_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) := B_0(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})P_0\,.$$

Then, assuming that both

$$\ker(A_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})) \text{ and } \operatorname{im}(A_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})) \text{ are independent of } \dot{\boldsymbol{\xi}}\,, \tag{3.25}$$

the considerations of above can be repeated with $N_1(t,\boldsymbol{\xi}) := \ker(A_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}))$, a projector $Q_1(t,\boldsymbol{\xi})$ on $N_1(t,\boldsymbol{\xi})$, $P_1(t,\boldsymbol{\xi}) := \boldsymbol{Id} - Q_1(t,\boldsymbol{\xi})$, a projector $W_1(t,\boldsymbol{\xi})$ along

$\text{im}\left(A_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})\right)$, and the space

$$S_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) := \left\{\bar{\boldsymbol{\xi}} \in \mathbb{R}^{n_\xi} : W_1(t,\boldsymbol{\xi})B_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})\bar{\boldsymbol{\xi}} = 0\right\} .$$

Similar to the notion of index-1 tractability (Definition 3.4) index-2 tractability is then introduced by

Definition 3.5 (Index-2 Tractability)
If $A_0(t,\boldsymbol{\xi})$ *is singular, and if Eq.* (3.23a) *is not tractable with index 1, and if* $\dim\left(N_0 \cap S_0(t,\boldsymbol{\xi})\right)$ *is constant, then Eq.* (3.23a) *has* tractability index 2 *if and only if*

$$N_1(t,\boldsymbol{\xi}) \cap S_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) = \{0\} ,$$

or, equivalently, if and only if

$$A_2(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) := A_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) + B_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})Q_1(t,\boldsymbol{\xi}) \text{ is nonsingular.}$$

Remark 3.9:
For quasi-linear and semi-explicit DAEs with the property that differentiation of the algebraic part gives a *transferable DAE*, i.e, a DAE for which the index-reduced system is uniformly of index-1, index-2 tractability is essentially equivalent to a differential index of 2 [März 89]. ◊

Remark 3.10:
The previous definitions for index-1 and index-2 tractability indicate a recursive nature of the tractability index. A general recursive definition of index-ι_t tractability, $\iota_t \in \mathbb{N}$, is given in [März 92] for linear DAEs. ◊

It can be shown that the tractability index is independent from the special choice of the projectors. [EsLa 99] choose $Q_1(t,\boldsymbol{\xi})$ as the *canonical projector* onto $N_1(t,\boldsymbol{\xi})$ along $S_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})$ which has the properties

$$\begin{aligned} Q_1(t,\boldsymbol{\xi}) &= Q_1(t,\boldsymbol{\xi})A_2^{-1}(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})B_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}}) , \quad \text{and} \\ N_0 \cap S_0(t,\boldsymbol{\xi}) &= \text{im}\left(Q_0Q_1(t,\boldsymbol{\xi})\right) \neq \{0\} . \end{aligned}$$

In [EsLa 99] consistent initial conditions in the sense of Definition 1.15 for a DAE Eq. (3.23a) with tractability index 2 are obtained by

1. generation of an equivalent index-1 system,
2. appropriate selection of a subset of the state variables $\boldsymbol{\xi}$ that are assigned user given values,
3. and solution of the resulting full rank system of nonlinear equations.

The first two steps of this method are based on the projectors introduced in the analysis above. In the sequel it is assumed that it is possible to find a suitable projector $W_1(t,\boldsymbol{\xi})$ such that $\text{im}(W_1(t,\boldsymbol{\xi}))$ = constant. This is the case, e.g., for

Hessenberg systems and for systems Eq. (3.23a) arising from the modelling of electric circuits using *Modified Nodal Analysis* (MNA). Further the constant diagonal matrix I_{W_1} is introduced as

$$[I_{W_1}]_{\mu,\mu} = \begin{cases} 1\,; & \text{if} \quad \exists \nu \in \{1,\ldots,n_{\boldsymbol{\xi}}\} : [W_1(t,\boldsymbol{\xi})]_{\mu,\nu} \neq 0\,, \\ 0\,; & \text{otherwise}\,. \end{cases}$$

I_{W_1} is used to select those equations that are to be differentiated with respect to time, whereas W_1 describes how to combine the differentiated equations (by construction, $W_1(t,\boldsymbol{\xi})I_{W_1} = W_1(t,\boldsymbol{\xi})$). It should be noted that the dependency of I_{W_1} from $(t,\boldsymbol{\xi})$ has been dropped. Then, under the additional assumptions that

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[I_{W_1} W_0 \boldsymbol{F}(t,\boldsymbol{\xi}(t),\dot{\boldsymbol{\xi}}(t))\right] \quad \text{exists}\,, \tag{3.26a}$$

$$\ker\left\{\frac{\partial}{\partial \boldsymbol{\xi}}\left(W_1(t,\boldsymbol{\xi})\cdot\left[\frac{\partial (I_{W_1} W_0 \boldsymbol{f})(t,\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\cdot\dot{\boldsymbol{\xi}} + \frac{\partial (I_{W_1} W_0 \boldsymbol{f})(t,\boldsymbol{\xi})}{\partial t}\right]\right)\right\} \subset \{N_0 \cap S_0(t,\boldsymbol{\xi})\}\,, \quad \text{and} \tag{3.26b}$$

$$\forall \bar{\boldsymbol{\xi}} \in \mathbb{R}^{n_{\boldsymbol{\xi}}} : \frac{\partial([P_0 P_1(t,\boldsymbol{\xi})]\cdot[\boldsymbol{\xi}-\boldsymbol{\xi}_0])}{\partial \boldsymbol{\xi}}\cdot[\boldsymbol{Id} - P_0 Q_1(t,\boldsymbol{\xi})]\cdot\bar{\boldsymbol{\xi}} = 0 \;\Rightarrow\; P_0 P_1(t,\boldsymbol{\xi})\bar{\boldsymbol{\xi}} = 0\,, \tag{3.26c}$$

the system

$$0 = \left[\begin{array}{c} \boldsymbol{F}(t,\boldsymbol{\xi},\dot{\boldsymbol{x}}) \\ [P_0 P_1(t,\boldsymbol{\xi})]\cdot[\boldsymbol{\xi}-\boldsymbol{\xi}_0] \\ Q_0\dot{\boldsymbol{x}} \\ W_1(t,\boldsymbol{\xi})\cdot\left[\frac{\partial (I_{W_1} W_0 \boldsymbol{f})(t,\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\cdot\dot{\boldsymbol{x}}(t_0) + \frac{\partial (I_{W_1} W_0 \boldsymbol{f})(t,\boldsymbol{\xi})}{\partial t}\right] \end{array}\right]_{\substack{t=t_0 \\ \boldsymbol{\xi}=\boldsymbol{\xi}(t_0), \\ \dot{\boldsymbol{x}}=\dot{\boldsymbol{x}}(t_0)}}, \tag{3.27}$$

with $\dot{\boldsymbol{x}} := P_0\dot{\boldsymbol{\xi}}$, represents a DAE of tractability index 1 equivalent to a given index-2 tractable DAE Eq. (3.23a) at $t = t_0$.

Remark 3.11:
In Eq. (3.23b) the projector $P_{\mathrm{ini}}(t,\boldsymbol{\xi})$ selecting the dynamic degrees of freedom was intermediately introduced. In Eq. (3.27) it is specified as $P_{\mathrm{ini}}(t,\boldsymbol{\xi}) := P_0 P_1(t,\boldsymbol{\xi})$. ◊

Theorem 3.5 shows that the nonlinear system of equations Eq. (3.27) in the unknowns $\boldsymbol{\xi}(t_0)$ and $\dot{\boldsymbol{x}}(t_0)$ can be solved for consistent initial conditions [EsLa 99]:

Theorem 3.5 (Regularity of Projected Consistency Equations)
Let the assumptions Eq. (3.24), Eq. (3.25), and Eqs. (3.26a)–(3.26c) be valid. Then the system Eq. (3.27) has a full rank Jacobian in a neighbourhood of a solution.

Eq. (3.24) and Eq. (3.25) are dependency conditions, Eq. (3.26a) is a smoothness

condition. Eq. (3.26b) can be shown to hold for linear time-dependent and Hessenberg systems as well as for a class of DAE models arising from Modified Nodal Analysis. Eq. (3.26c) is fulfilled, e.g., if $[P_0P_1(t,\boldsymbol{\xi})]$ is constant or purely time-dependent, or if $[P_0Q_1(t,\boldsymbol{\xi})]$ is constant (given for MNA), or for some mechanical systems.

By the special choice $W_1(t,\boldsymbol{\xi}) := A_2(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})P_0Q_1(t,\boldsymbol{\xi})A_2^{-1}(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}})$ [EsLa 99] transform Eq. (3.27) into the nonlinear system of equations

$$0 = \left\{\boldsymbol{F}(t,\boldsymbol{\xi},\dot{\boldsymbol{x}})\right\}\Big|_{\substack{t=t_0\\ \boldsymbol{\xi}=\boldsymbol{\xi}(t_0),\\ \dot{\boldsymbol{x}}=\dot{\boldsymbol{x}}(t_0)}}, \tag{3.28}$$

$$0 = \Big\{[P_0P_1(t,\boldsymbol{\xi})]\cdot[\boldsymbol{\xi}-\boldsymbol{\xi}_0] + P_0Q_1(t,\boldsymbol{\xi})\left[\dot{\boldsymbol{x}} + A_2^{-1}(t,\boldsymbol{\xi},\dot{\boldsymbol{x}})\frac{\partial \boldsymbol{F}(t,\boldsymbol{\xi},\dot{\boldsymbol{x}})}{\partial t}\right] + Q_0\dot{\boldsymbol{x}}\Big\}\Big|_{\substack{t=t_0\\ \boldsymbol{\xi}=\boldsymbol{\xi}(t_0),\\ \dot{\boldsymbol{x}}=\dot{\boldsymbol{x}}(t_0)}}. \tag{3.29}$$

System Eqs. (3.28)–(3.29) is solved by a Newton-type algorithm with an inexact Jacobian dropping the dependencies of the projectors $P_0P_1(t,\boldsymbol{\xi})$, $P_0Q_1(t,\boldsymbol{\xi})$, and $A_2(t,\boldsymbol{\xi},\dot{\boldsymbol{x}})$. The projectors can be calculated by Householder decomposition (QR) or by Singular Value Decomposition (SVD). [EsLa 99] employ the first method due to computational costs. Still the expensive generation of the projectors restricts this technique to small systems or systems where the projectors can be obtained analytically [HaLa 01]. [HaLa 01] implement a modified version of the method. Their implementation addresses DAEs arising from the discretisation of PDEs by the method of lines. On the one hand, this means that sparse matrix techniques have to be employed due to the problem dimensions (as an example problem [HaLa 01] initialise a DAE with 5029 variables). On the other hand, special properties of these systems resulting from PDE discretisation can be utilised.

Based on the projector technique [Este 99b] proposes an algorithm tailored for the computation of consistent initial values for a class of DAEs arising from MNA. Her algorithm employs a topological analysis of the electrical network as described in [Este 99a]. [Este 99b] also addresses the problem of an appropriate choice of the value of the vector $\boldsymbol{\xi}_0$ of initial conditions. For the special application treated the values are assumed to be given either as the DC operating point of the electrical network or as (the state vector part of) a solution of the DAE before a discontinuity.

In [Este 00] a two-step method for the computation of consistent initial conditions motivated by the projector technique is proposed. In a first step an intermediate value for the initial conditions is computed which fulfils the explicitly given DAE only. In a second step this value is modified such that a set of consistent initial conditions is obtained, i.e., a set which also satisfies the hidden constraints. As a main advantage of this method the determination of a set of appropriate dynamic degrees of freedom (here corresponding to $[P_0P_1(t,\boldsymbol{\xi})]\cdot[\boldsymbol{\xi}-\boldsymbol{\xi}_0]$) is not

required. At the same time, user given specifications for the initial values in the differential variables are observed exactly in as many components as possible. A drawback of this method is that special assumptions on the structure of the DAE have to be fulfilled.

3.1.10 The Back-Tracing Method

In this section we consider a numerical method that can be used to obtain "numerically consistent initial conditions" *without* explicit determination of the consistency equations and of the dynamic degrees of freedom. It is based on a special property of BDF integration codes, a class of implicit linear multi-step integrators (cf., e.g., [Gear 71], [DeBo 94], and the discussion in Section 2.4.3.a). Given initial conditions that are not too bad the BDF integrator may arrive at the solution manifold within several integration steps. Then integration proceeds in a normal way. The idea of the *back-tracing* method proposed by [SEYE 81] is to revert the direction of integration after the solution manifold has been reached until arriving at the initial time again. In this way consistent initial conditions can be obtained that are related to the numerically computed trajectory.

[SEYE 81] examine linear constant coefficient DAEs of the form

$$A\dot{\boldsymbol{\xi}}(t) + B\boldsymbol{\xi}(t) + \boldsymbol{u}(t) = 0\,; \quad t \in [t_0, t_\mathrm{f}]\,, \tag{3.30a}$$

$$\boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0\,, \tag{3.30b}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$, and $A, B \in \mathbb{R}^{n_\xi \times n_\xi}$. $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^{n_\xi}$ is assumed to be sufficiently smooth. If A is singular the IVP Eqs. (3.30a)–(3.30b) does not own a solution for general $\boldsymbol{\xi}_0 \in \mathbb{R}^{n_\xi}$ due to the algebraic components of the DAE (in case of a regular matrix A Eq. (3.30a) is an ODE). Vectors $\boldsymbol{\xi}_0$ for which a solution of the IVP exists are called *admissible initial conditions* [SEYE 81].

Definition 3.6 (Admissible Initial Conditions)
An initial condition $\boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0 \in \mathbb{R}^{n_\xi}$ Eq. (3.30b) is said to be admissible *if and only if a unique solution $\boldsymbol{\xi}(t)$ for Eq. (3.30a) exists (in $[t_0, t_0 + \epsilon[$, $\epsilon > 0$) such that $\boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0$.*

Gear's k-step method applied to Eq. (3.30a) reads as

$$(A + h\beta_0 B)\,\boldsymbol{\xi}_n = \left(\sum_{\nu=1}^{k} \alpha_\nu A \boldsymbol{\xi}_{n-\nu}\right) - h\beta_0 \boldsymbol{u}(t_n)\,, \tag{3.31}$$

where $\boldsymbol{\xi}_n$, $\nu = 0, 1, 2, \ldots$, are the numerical approximations to the solution trajectory at t_n, $\nu = 0, 1, 2, \ldots$. The coefficients β_0 and α_ν, $\nu = 1, \ldots, k$ are given

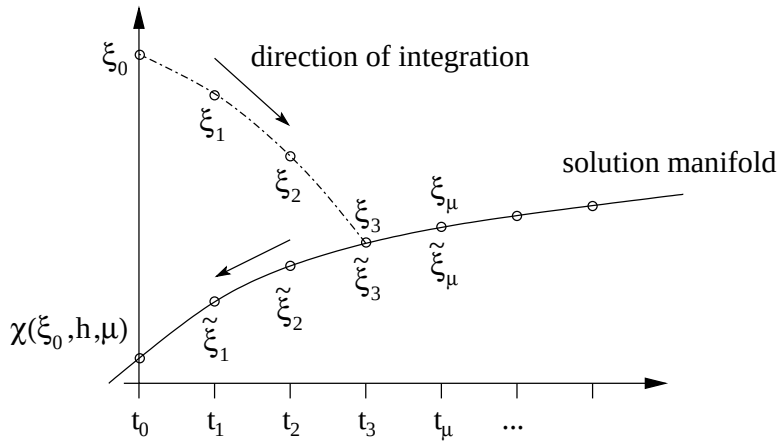

Figure 3.1: The back-tracing function (sketch), cf. Definition 3.7.

by

$$\beta_0 := \left(\sum_{\nu=1}^{k} \frac{1}{\nu}\right)^{-1} \quad \text{and} \quad \alpha_\nu := (-1)^{\nu+1}\beta_0 \sum_{\mu=\nu}^{k} \frac{1}{\mu}\binom{\mu}{\nu}\,, \quad \nu = 1,\ldots,k\,.$$

As shown in [SEYE 81], starting from not too bad (i.e., the integrator must not fail) but otherwise arbitrary – i.e., even inadmissible – initial values $\boldsymbol{\xi}_0$, rule Eq. (3.31) applied to DAEs Eq. (3.30a) generates a numerical solution equivalent to some admissible initial condition $\widehat{\boldsymbol{\xi}_0}$ after sufficiently many integration steps. This result is summarised in Theorem 3.6.

Theorem 3.6 (Gear's k-Step Method on Index-ι_d Linear DAEs)
Suppose $\mathcal{I} \subset \mathbb{R}^{n_\xi}$ is the set of admissible initial conditions for Eq. (3.30a)*. With a fixed step-size $h \in \mathbb{R}_+$ let $\boldsymbol{\xi}_n(\boldsymbol{\xi}_0) \in \mathbb{R}^{n_\xi}$, $n = 1, 2, \ldots$, be the approximation obtained for $\boldsymbol{\xi}(nh)$ by Gear's k-step method*[ii] *defined in Eq.* (3.31) *with the initial condition $\boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0$, $\boldsymbol{\xi}_0 \in \mathbb{R}^{n_\xi}$.*

Then there exists a unique $\widehat{\boldsymbol{\xi}_0} \in \mathcal{I}$ such that $\boldsymbol{\xi}_n(\boldsymbol{\xi}_0) = \boldsymbol{\xi}_n(\widehat{\boldsymbol{\xi}_0})$ for $n \geq (\iota_d - 1)k + 1$, where ι_d is the nil-potency of Eq. (3.30a) *(which is equivalent to the differential index).*

The next step is to compute the actual admissible initial condition $\widehat{\boldsymbol{\xi}_0}$ equivalent to the given initial conditions $\boldsymbol{\xi}_0$. This can be done using the back-tracing function introduced in Definition 3.7 (see Figure 3.1):

[ii] *The theorem is stated under the assumption that the values $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_k\}$ are generated using lower-order Gear formulas, i.e., the order of the method to be used is* $\min\{k, n\}$.

Definition 3.7 (Back-Tracing Function for Linear DAEs)
Let there be a linear constant coefficient DAE Eq. (3.30a) *and a number* $\mu \in \mathbb{N}$. *Given not too bad, but otherwise arbitrary initial values* $\boldsymbol{\xi}_0 \in \mathbb{R}^{n_\xi}$ *apply the backward Euler method integrating forward*

$$(A + hB)\,\boldsymbol{\xi}_n = A\boldsymbol{\xi}_{n-1} - h\boldsymbol{u}(t_n)\,;\ n = 1, \ldots, \mu\,,$$

and obtain $\boldsymbol{\xi}_\nu$, $\nu = 1, \ldots, \mu$. *Then write* $\widetilde{\boldsymbol{\xi}}_\mu := \boldsymbol{\xi}_\mu$ *and apply the forward Euler method integrating backwards generating* $\widetilde{\boldsymbol{\xi}}_\nu$, $\nu = \mu - 1, \ldots, 0$:

$$(A - hB)\,\widetilde{\boldsymbol{\xi}}_{\mu-n} = A\widetilde{\boldsymbol{\xi}}_{\mu-n+1} + h\boldsymbol{u}(t_{\mu-n})\,;\ n = 1, \ldots, \mu\,.$$

Then the back-tracing function $\boldsymbol{\chi}$ *is defined by*

$$\boldsymbol{\chi}(\boldsymbol{\xi}_0, h, \mu) := \widetilde{\boldsymbol{\xi}}_0\,.$$

In words, the back-tracing function $\boldsymbol{\chi}(\boldsymbol{\xi}_0, h, \mu)$ introduced in Definition 3.7 provides the numerical approximation to the desired admissible initial condition $\widehat{\boldsymbol{\xi}_0}$. The last theorem in this paragraph establishes convergence of the back-tracing method for systems of the type Eq. (3.30a) [SEYE 81]:

Theorem 3.7 (Convergence of the Back-Tracing Method)
Suppose the back-tracing function $\boldsymbol{\chi}(\boldsymbol{\xi}_0, h, \mu)$ *is applied to Eq.* (3.30a). *Then for* $\mu \geq \iota_d$, *and as* h *approaches zero,* $\boldsymbol{\chi}(\boldsymbol{\xi}_0, h, \mu)$ *approaches a unique admissible initial condition.*

3.1.11 Solution of a BVP

[AmMa 97], [AmMa 98] propose an algorithm for the computation of consistent initial conditions by the solution of a BVP. In [AmMa 97] the method is applied to nonlinear semi-explicit index-1 DAEs. [AmMa 98] show convergence for index-2 and index-3 DAEs in Hessenberg form.

The approach is based on the discretisation of the continuous DAE problem by *Generalised BDF* (GBDF) [AmMa 94], [BrTr 96], [ISM 98]. Consider the nonlinear DAE initial value problem

$$\boldsymbol{F}(t, \boldsymbol{\xi}(t), \dot{\boldsymbol{\xi}}(t)) = 0\,; \quad t \in [t_0, t_\mathrm{f}] \subset \mathbb{R}, \tag{3.32a}$$

$$\boldsymbol{k}^\mathrm{ini}(\boldsymbol{\xi}(t_0)) = 0\,, \tag{3.32b}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ and $\boldsymbol{F} : \mathbb{R}^{1+n_\xi+n_\xi} \to \mathbb{R}^{n_\xi}$. The initial conditions $\boldsymbol{k}^\mathrm{ini} : \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_\xi}$ given by the user are assumed to be feasible in the sense of guaranteeing existence and uniqueness of a solution of Eqs. (3.32a)–(3.32b). Now let there be a uniform mesh of $s + 1$ points with step-size $h \in \mathbb{R}_+\backslash\{0\}$, i.e., $t_\nu = t_0 + \nu h \in [t_0, t_\mathrm{f}]$,

$\nu = 0, \ldots, s$. Then the DAE evaluated at the mesh points gives the conditions

$$\boldsymbol{F}(t_\nu, \boldsymbol{\xi}(t_\nu), \dot{\boldsymbol{\xi}}(t_\nu)) = 0\,; \quad \nu = 0, \ldots, s\,, \tag{3.33a}$$

$$\boldsymbol{k}^{\mathrm{ini}}(\boldsymbol{\xi}(t_0)) = 0\,. \tag{3.33b}$$

Taken together, Eqs. (3.32a)–(3.32b) and Eqs. (3.33a)–(3.33b) set up an overdetermined multipoint boundary value problem (MPBVP) on $[t_0, t_0 + sh]$. By construction this MPBVP is uniquely solvable if the IVP Eqs. (3.32a)–(3.32b) is uniquely solvable on $[t_0, t_0 + sh]$, and it owns the same solution trajectory as the IVP on this interval.

Let the numerical approximations to the discretised state variable trajectory and its time derivative on the grid be denoted as $\boldsymbol{\xi}_\nu \doteq \boldsymbol{\xi}(t_\nu)$ and $\dot{\boldsymbol{\xi}}_\nu \doteq \dot{\boldsymbol{\xi}}(t_\nu)$, $\nu = 0, \ldots, s$, respectively. Further let $k_{\mathrm{I}} := \lfloor k/2 \rfloor + 1$ and $k_{\mathrm{F}} := k - k_{\mathrm{I}}$. Then the k-step GBDF is set up by the *initial additional methods*

$$\dot{\boldsymbol{\xi}}_\nu = \frac{1}{h} \sum_{\mu=0}^{k} \alpha_\mu^{(\nu)} \boldsymbol{\xi}_\mu\,; \qquad \nu = 1, \ldots, k_{\mathrm{I}} - 1\,,$$

by the *final additional methods*

$$\dot{\boldsymbol{\xi}}_\nu = \frac{1}{h} \sum_{\mu=0}^{k} \alpha_\mu \boldsymbol{\xi}_{\nu - k_{\mathrm{I}} + \mu}\,; \qquad \nu = k_{\mathrm{I}}, \ldots, s - k_{\mathrm{F}}\,,$$

and by the *main method*

$$\dot{\boldsymbol{\xi}}_\nu = \frac{1}{h} \sum_{\mu=0}^{k} \alpha_\mu^{(\nu - s + k)} \boldsymbol{\xi}_{s - k + \mu}\,; \quad \nu = s - k_{\mathrm{F}} + 1, \ldots, s\,.$$

A table of the coefficients α is given in [BrTr 96] for $k = 1, \ldots, 10$ (GBDF possess suitable stability for $k \leq 9$ [AmMa 98]). Using these GBDF the MPBVP Eqs. (3.32a)–(3.32b), Eqs. (3.33a)–(3.33b) on $[t_0, t_0 + sh]$ is discretised as

$$\begin{bmatrix} \gamma \cdot \boldsymbol{k}^{\mathrm{ini}}(\boldsymbol{\xi}_0) + \boldsymbol{F}(t_0, \boldsymbol{\xi}_0, \dot{\boldsymbol{\xi}}_0) \\ \boldsymbol{F}(t_1, \boldsymbol{\xi}_1, \dot{\boldsymbol{\xi}}_1) \\ \vdots \\ \boldsymbol{F}(t_s, \boldsymbol{\xi}_s, \dot{\boldsymbol{\xi}}_s) \end{bmatrix} = 0\,. \tag{3.34}$$

I.e, the fixed distance between the constraint node points of the MPBVP is chosen as the step length of the GBDF method, and the initial condition is enforced by a linear combination with the value of the DAE at the initial point. [AmMa 98] propose a choice of the real-valued constant $\gamma \in \mathbb{R} \setminus \{0\}$ which facilitates the solution of Eq. (3.34). Furthermore, they propose an error-based strategy for the

automatic selection of the step-size h and of the order k of the method. In order to achieve convergence for an index ι_d-DAE the order must be greater then $\iota_d - 2$.

After solution of the nonlinear system of equations Eq. (3.34) $\boldsymbol{\xi}_0$ and $\dot{\boldsymbol{\xi}}_0$ provide numerically consistent initial values.

3.1.12 Other Methods

Finally we mention some other methods which can be used in order to generate consistent initial conditions. As in Section 3.1.10 and Section 3.1.11 the term *consistent initial conditions* may be used in a way which differs from Definition 1.12.

The well-known BDF integrator code DASSL – which is basically designed for index-1 systems – uses a small implicit Euler step in order to arrive at (numerically) consistent initial conditions [BCP 96]. A drawback is that the consistent initial values are not obtained at the initial time t_0 but for a point in time $t_0 + h_{\text{ini}}$ close to it [LiPe 99]. The back-tracing method discussed in Section 3.1.10 overcomes this drawback at an increased computational cost. In [KMG 92], [Krön 02] improvements to the initial Euler step method of DASSL are proposed, including a modified initial step size selection strategy, an adapted corrector iteration scheme, and back-tracing.

In order to obtain improved results [Krön 02] proposes to employ extrapolation of the initial values obtained with the modified DASSL Euler-step. In the *back-extrapolation approach* several modified initial Euler-steps with different steplength are made. The initial conditions are then approximated by a linear combination of the results of these Euler-steps. The coefficients of the linear combination are chosen such that certain step-size dependent error terms are cancelled.

Since version 2.0 DASPK [BHP 94], the successor of DASSL, can generate consistent initial conditions for index-1 DAEs by the solution of the (reduced) consistency equations [BHP 98], [LiPe 99]. The nonlinear system of equations is solved by a modified Newton algorithm with linesearch backtracking. From version 3.0 on DASPK can also compute consistent initial conditions for Hessenberg index-2 DAEs in a way similar to the index-1 case if the original algebraic equations are already satisfied [LiPe 99]. If necessary, a predictor-corrector type method is applied. Still the user has to specify the dynamic degrees of freedom (to these variables initial values prescribed by the user are then assigned) as well as the index-2 constraints.

[MMG 95] develop a method for the computation of consistent initial conditions in the case of index-1 DAEs with step-discontinuities in the *forcing functions* (in our terminology *forcing functions* are equivalent to the control inputs). They propose a continuation method as a robust solution technique for the nonlinear consistency equations. Additionally, they examine the feasibility of continuity conditions in the differential states.

In [Gerd 01] and in the related paper [GeBü 01] a direct multiple shooting

method for the numerical solution of optimal control problems with DAE models is developed. In the direct multiple shooting technique the controls are approximated by parameterised functions. Additionally a mesh (the multiple shooting nodes) is chosen, splitting the integration interval into multiple shooting intervals. On each multiple shooting interval a new DAE-IVP is defined. The initial conditions for the DAE-IVPs at the multiple shooting nodes are then optimisation variables in the resulting NLP, cf. Section 2.4.1. The point is that the initial conditions proposed by the optimiser during the course of the solution of the NLP are in general not consistent. For semi-explicit index-2 DAEs [Gerd 01] develop a projection-type method based on the sequential solution of an NLP by SQP and on the solution of an index-1 system (derived from the DAE) by a Newton-type method in order to obtain consistent initial conditions. This two-step approach differs significantly from the method discussed in Section 3.1.2 where all state variables are determined simultaneously within an SLP. For implicit DAEs of higher index [Gerd 01] combines the projection approach with a (virtual) discretisation of the initial integration step by an appropriate Runge-Kutta method. In this way approximations to consistent initial conditions for the differential variables at the multiple-shooting nodes can be generated. Finally, [Gerd 01] proposes a relaxation approach for the multiple-shooting method if the model DAE is semi-explicit. In this approach bias terms are introduced to the DAEs on the multiple shooting intervals such that arbitrary initial conditions are consistent initial conditions for the relaxed DAE systems. Additionally, equality conditions are added to the multiple-shooting NLP which enforce vanishing bias terms at the solution of the NLP.

[Albe 99], [AFS 00] propose a perturbation analysis approach in order to calculate consistent initial conditions for DAEs originating from mechanical systems (due to their origin from Lagrange's equation of motion [Albe 99] terms such systems *Lagrangian DAEs* (LDAEs)). The semi-symbolic method requires a start value which already lies on the constraint manifold, e.g., a steady-state, and computes the requested set of initial values for the actual initialisation problem interpreting it as a related perturbed problem. If the desired point is too far from the initial point several intermediate steps may be made. The two main ingredients of the algorithm are the replacement of nonlinear functions by truncated Taylor's series expansions in the perturbation parameter, and the expression of the solution of a singular linear system of differential equations

$$A\dot{\boldsymbol{\xi}}(t) + B\boldsymbol{\xi}(t) + \boldsymbol{u}(t) = 0\,,$$

$\boldsymbol{\xi} \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$, $A, B \in \mathbb{R}^{n_{\boldsymbol{\xi}} \times n_{\boldsymbol{\xi}}}$ $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$ with a *regular matrix pencil*[(iii)] $[\lambda A - B]$, $\lambda \in \mathbb{C}$, by means of the *Drazin inverse*[(iv)] of A. [Albe 99] provides an implementation

[(iii)] A matrix pencil $[\lambda A - B]$, $A, B \in \mathbb{R}^{n_{\boldsymbol{\xi}} \times n_{\boldsymbol{\xi}}}$, $\lambda \in \mathbb{C}$, is called regular if $\det([\lambda A - B])$ is not identically zero as a function of λ [BCP 96].

[(iv)] For the definition of the Drazin inverse of a matrix we refer to [Albe 99].

as a MAPLE™ program.

A fundamentally different approach for the integration of DAEs with discontinuous forcing functions is to avoid the difficult computation of consistent initial conditions in the first place by smoothening the discontinuous input. An examination made by [KMG 97] seems to cancel out such an approach. [ViBi 01] assert to having arrived at more promising results. There, a more refined approximation than in [KMG 97] is employed. Additionally the DAE itself is replaced at the discontinuity by a "similar" system which originally is at steady-state. The latter is based on the assumption that the dynamics of a system may be approximated by the response of a related system (originally at steady-state) to a discontinuous perturbation. [Krön 02] recommends smoothening as a viable alternative to rigorous consistent initialisation, especially if the latter approach is too expensive in numerical execution or algorithmic implementation.

3.1.13 Specification of Transition Conditions

In several algorithms discussed above a central problem has not been treated. It is the automatic generation of appropriate transition conditions. For small DAEs a sophisticated user may be in the situation to provide the required transition conditions. In the context of larger systems or models generated by simulation tools the specification of appropriate transition conditions is generally a difficult task. In the case of automatically generated models, e.g., in common already the physical meaning of the various computer-aided derived equations and variables in the system is unclear to the user. Thus depending on the problem setting user interaction may be inefficient, or it may be actually impossible.

3.1.13.a General Work

Currently a general, reliable, and theoretically well-founded method for the determination of transition conditions is still missing [BAFG 98] especially for higher index DAEs although there has been considerable research on this topic. (Partial) solutions have been reported in some special cases.

If at the point of initialisation $t = t_{\text{disc}}$ the transition conditions employed are of the form

$$0 = \boldsymbol{z}(t_{\text{disc}}^{+}) - \boldsymbol{z}_0 \,,$$

where $\boldsymbol{z} \in \mathbb{R}$ is a state variable or the first derivative of a state variable with respect to time, and with a parameter $\boldsymbol{z}_0 \in \mathbb{R}$, e.g., a constant value or – in case of continuity conditions – the value of the variable $\boldsymbol{z}(t_{\text{disc}}^{-})$ immediately before the initialisation, then we call the problem of determining appropriate transition conditions the *assignment problem*. This expression is motivated by the direct equivalence of the choice of transition conditions with the assignment of the dynamic degrees of freedom. The term, however, is not to be mixed with the term *assignment* used in graph-theory, cf. Definition 3.2 in Section 3.1.5.

In chemical engineering physical insight can be employed to narrow down the search for the transition conditions. The general rule is that transition conditions should be used to preserve continuity of differential variables. On the one hand, *conserved* (or *integral*) quantities, e.g., mass or enthalpy, are modelled by differential states. On the other hand, quantities directly related to the conserved quantities are in general represented by differential states [BaPk 97]. We have already encountered a restriction in the choice of the degrees of freedom to the differential states in Section 3.1.4 (cf. Eq. (3.17)) when discussing the results of [GPS 95]. However, there no deeper reasons for the applicability of this restriction are given. A similar strategy is used in the projector based algorithm reviewed in Section 3.1.9 in the context of electrical circuits.

But the assumption of continuity in the differential states may not hold for impulsive changes in, e.g., mass or enthalpy [VSP 94a]. Additionally in the case of higher index DAEs the number of dynamic degrees of freedom is smaller than the number of differential variables. While the first point can be *utilised* for the specification of the transition conditions (see our algorithm in Section 3.2.4), the second point in the general case leads to non-uniqueness of the assignment problem which makes this problem difficult to solve. However there are classes of DAEs for which the assignment problem can be solved uniquely.

In [MMG 95] linear-implicit index-1 DAEs with discontinuous control inputs are considered. Depending on the structure of the DAEs several cases are identified where the assignment problem can be solved (at least partially) uniquely.

[BrPa 92] consider the same class of problems. For discontinuous control inputs they introduce the *genuine initial value* of a DAE which, however, may not exist. On the other hand, they give a sufficient condition for its existence based on the existence of potentials. These potentials also directly specify the transition conditions for the differential states. The drawback of this approach is that the potentials have to be known, i.e., the assignment problem is primarily shifted to the detection of suitable potentials. [BrPa 92] show that if the genuine initial value exists and if no integrating factors are present then these initial values can be obtained by replacing the jump in the controls by an integration along a smooth virtual transition in the controls.

For linear time invariant DAEs of arbitrary index [BaGa 00] show that the consistent initialisation problem, and thus the assignment problem, is always fully and uniquely determined. Based on this result they propose two algorithms for consistent initialisation. The first one is based on the canonical form of the linear DAE, the second one is based on Pantelides' Algorithm.

[BaPk 97] show that in the case of higher index DAE for which the family of equivalent index-1 DAEs obtained by the dummy derivative method exactly contains one member also the assignment problem can be solved uniquely.

In Section 3.1.2 the a priori selection of a *specific* set of transition conditions is avoided by the formulation of the consistent initialisation problem as an NLP [GoBi 99]. Given a set of possible transition conditions the choice of the final set

of transition conditions is left to the NLP solver. For linear implicit DAEs with discontinuous forcing functions [BaGa 00] favour the solution of an overdetermined initialisation problem using the approach of [GoBi 99]. There the (possibly index reduced equivalent) index-1 DAE is solved together with the known state transfer functions and all possible continuity conditions for the states. The analysis of [GoBi 99] regarding discontinuities is restricted to the special case of jump discontinuities in forcing functions. Other scenarios, e.g., changes in the structure of the DAE, are not explicitly considered.

3.1.13.b A Structural Approach

Pantelides' Algorithm, – or, more precisely, subroutine AUGMENTPATH – can be employed to *check* a given set of transition conditions for consistency with a system of reduced derivative array equations. If AUGMENTPATH cannot find an augmenting path for at least one of the E-nodes in the combined system of reduced derivative array equations and transition conditions, then this combined system is structurally singular. In this case the set of transition conditions is not adequate [Pant 88a].

Thus if only a relatively small set of potential transition conditions is considered, e.g., by restriction to continuity in the differential state variables, Pantelides' Algorithm can be employed to set up a procedure for the determination of a suitable set of variables that may be *assigned* arbitrary initial values based on complete enumeration and test of all possible combinations of potential transition conditions. In the notation of Section 3.1.5 such a procedure has to enumerate every combination of n_{ddf} variables out of the $n_{\boldsymbol{x}}$ candidates for continuity at the point of initialisation in the set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_{\boldsymbol{x}}}\}$. The admissibility of a set of transition conditions is then checked by application of AUGMENTPATH to the corresponding system of consistency equations. In case of an admissible set of transition conditions the procedure may terminate if an additional test shows that the system of consistency equations is not only structurally regular, but also numerically solvable. In the worst case all possible combinations of transition conditions have to be checked. However, already for medium size DAEs such a method is computationally infeasible due to the combinatorial complexity of the problem.

Example 5:
For $n_{\boldsymbol{x}} = 200$ and $n_{\mathrm{ddf}} = 195$ in the worst case $\binom{200}{195} \approx 2.5 \cdot 10^9$ combinations of transition conditions have to be considered. ◊

As a possible remedy [Pant 88a] mentions an algorithm originally developed by [Sarg 78] for the analysis of large-scale computing problems with a network structure. Applied to the consistent initialisation problem it can provide a criterion for which of the variables in the reduced system of derivative array equations are *possible candidates* for assignment (note that time derivatives of state variables are independent variables for the consistent initialisation problem, cf., e.g., Definition 1.12). Consider a general underdetermined system of m algebraic equations in n variables, $m < n$. Assuming that the algebraic equations are (functionally)

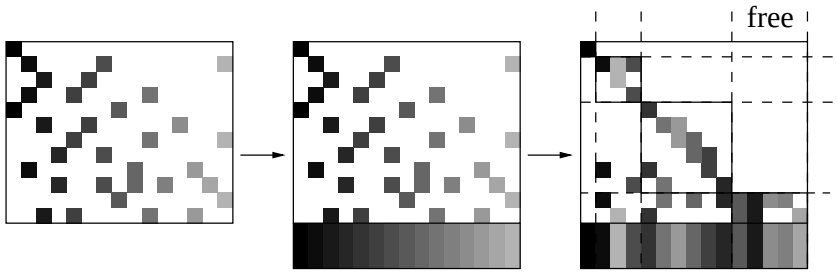

Figure 3.2: Block lower triangularisation (sketch).
The columns are differently shaded in order to indicate the permutation of the matrix (due to the limited graphics the rows are not differently shaded).

independent, there are $n - m$ degrees of freedom. The question is which of the variables may be specified in order to obtain a full rank system. Starting point is the structure of the Jacobian of the algebraic system of equations. [Sarg 78] proposes to add $n-m$ hypothetical equations, each of them containing all variables. From the structural point of view this is equivalent to the addition of $n - m$ full rows (with an entry in each column) to the Jacobian. The main step is a block-triangularisation of the square $n \times n$ system [Tarj 72] leading to a lower left block-triangular matrix.

Remark 3.12:
HSL routine MC23 [AEA 93] provides an implementation of a block lower triangularisation (BLT) algorithm for square, unsymmetric, and possibly structurally singular matrices (or for the matrix patterns, respectively). [PoFa 90] describe a more general algorithm which also computes a block lower triangularisation for rectangular matrices based on the *Dulmage and Mendelsohn decomposition.* ◊

Variables corresponding to the columns of the rightmost lower diagonal block are then candidates for specification as all other variables can be assumed to be determined, see Figure 3.2. In general, this block will contain more variables than there are degrees of freedom. I.e., in general the assignment problem cannot be finally solved by the method of [Sarg 78]. At this point specific knowledge about the problem modelled has to be used for the final choice of the set of transition conditions.

A drawback of the block triangularisation technique is the dependency of the result on the initial *ordering* of the rows and of the columns in the system Jacobian. I.e., the sequence of equations and variables in the DAE can affect the choice of the dynamic degrees of freedom. Additionally the method is sensitive to soft zeros (cf. Section 1.2) which can lead to a choice of dynamic degrees of freedom generating a structurally regular, but numerically non-regular system of reduced consistency equations.

3.2 Algorithm for the Calculation of Consistent Initial Conditions

Mostly Harmless

D. Adams: The Hitchhiker's Guide to the Galaxy

In the sequel we consider the consistent initialisation of semi-explicit index-2 DAEs of the special form

$$0 = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{y}(t)) - \dot{\boldsymbol{x}}(t)\,, \tag{3.35a}$$

$$0 = \boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{y}(t))\,, \tag{3.35b}$$

where $\boldsymbol{f} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$ and $\boldsymbol{g} : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$. The only information available on the system Eqs. (3.35a)–(3.35b) are

- the problem dimensions $n_{\boldsymbol{x}}$, $n_{\boldsymbol{y}}$,
- the value of the right hand side at a given point $[t, \boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}]$,
- the corresponding value of the Jacobians $\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right]$, $\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right]$, $\left[\frac{\partial \boldsymbol{f}}{\partial t}\right]$, and $\left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right]$, $\left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}\right]$, $\left[\frac{\partial \boldsymbol{g}}{\partial t}\right]$, as well as their patterns[(v)],
- problem dependent a priori knowledge from chemical engineering on the applicability of *some* of the differential variables $\{\dot{\boldsymbol{x}}_1, \ldots, \dot{\boldsymbol{x}}_{n_{\boldsymbol{x}}}\}$ as dynamic degrees of freedom (see, e.g., Example 6 on page 131),
- and an in general inconsistent vector $[\theta, \boldsymbol{x}(\theta), \boldsymbol{y}(\theta), \dot{\boldsymbol{x}}(\theta)]$ of variables originating from some other initial value computations (at the initial point of a simulation: $\theta = t_0$) or from a previous dynamical simulation (typically, the value of the discretised solution vector at the left hand side of a switching point: $\theta = t_{\text{disc}}^-$). Without loss of generality we consider $\theta = t_{\text{disc}}$ as the point at which consistent initialisation is required.

The DAEs we are aiming at are highly nonlinear and of very large scale (i.e, $n_{\boldsymbol{x}} + n_{\boldsymbol{y}} \approx \mathcal{O}(10^3), \ldots, \mathcal{O}(10^4)$). In common these DAEs are only defined on a certain subset of $\mathbb{R}^{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}}(\ni [\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}])$ due to physical reasons and modelling assumptions.

[(v)] Analytical expressions for the Jacobians with respect to time have been added by the Linde AG into OPTISIM® as a means to improve the consistent initialisation algorithm; they are *not* required for the plain integration of the DAE process models. As explicit time dependence can be restricted to a small number of unit models the additional effort required is small in relation to the improvements obtained.

3.2.1 Application of the Algorithm of Pantelides

The first step of our algorithm is the analysis of structural properties of Eqs. (3.35a)–(3.35b) by Pantelides' Algorithm (cf. Section 3.1.5). We employ its sparse implementation SPALG [UKM 95].

One of the fundamental pieces of information returned from Pantelides' Algorithm is the differential index ι_{d} of the DAE. More precisely, it returns its structural counterpart ι_{s}. At this point we have to *assume* that $\iota_{\mathrm{s}} = \iota_{\mathrm{d}}$ for the problems considered (cf. the discussion at the end of Section 1.2 and Remark 3.13 below). The generation of the index information renders an a priori index assumption unnecessary. Instead, we distinguish three alternatives depending on ι_{d}:

1. If $\iota_{\mathrm{d}} \leq 1$ we can set the number of dynamic degrees of freedom as $n_{\mathrm{ddf}} = n_{\boldsymbol{x}}$ and continue with the specification of the transition conditions in Section 3.2.4.

2. If $\iota_{\mathrm{d}} > 2$ we have to terminate with an error. This is, however, no restriction to the applicability of our algorithm for the calculation of consistent initial conditions in the context of the simulation environment OPTISIM®.

   In OPTISIM® a BDF algorithm is used for the integration of the dynamic model Eqs. (3.35a)–(3.35b) [EKKS 99], cf. Section 2.4.3.a. It is known that the BDF method is suitable for the direct integration of semi-explicit index-2 DAEs if the local error estimation formula is adapted. In general, the BDF method is not applicable to the integration of semi-explicit DAEs with $\iota_{\mathrm{d}} > 2$ [BCP 96]. Thus if $\iota_{\mathrm{d}} > 2$ is detected it is even *advisable* to terminate.

3. If $\iota_{\mathrm{d}} = 2$ we continue right here.

Remark 3.13:
A problem with SPALG as well as with any other structural algorithm is the influence of soft zeros on the results. We have observed cases in which SPALG returned a structural index lower than the differential index simply because of a special unit generating unnecessary soft zero entries in the sparse Jacobian of the DAE, i.e., entries in the Jacobian that are *identically* zero, in contrast to entries which are only locally zero. See also Example 1 on page 13. ◊

At this point we have assured that the DAE Eqs. (3.35a)–(3.35b) is of index 2, i.e., Pantelides' Algorithm terminates with the result that at least one of the algebraic equations $0 = \boldsymbol{g}_\mu(t, \boldsymbol{x}, \boldsymbol{y})$, $\mu = 1, \ldots, n_{\boldsymbol{y}}$, in Eq. (3.35b) has to be differentiated exactly once totally with respect to time in order to obtain a system of reduced consistency equations for the determination of consistent values for $[\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}]$, introducing time differentials of some of the algebraic variables $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_{\boldsymbol{y}}}\}$. Furthermore, none of the differential equations Eq. (3.35a) is to be differentiated. As a consequence no time derivative with order greater than one of any variable is to be considered.

We now examine the system of reduced derivative array equations in closer detail. Without restriction of generality suppose that the initial *equation association*

list (EAL) corresponding to Eqs. (3.35a)–(3.35b) is assigned as

$$\begin{aligned} \boldsymbol{f}_1(t,\boldsymbol{x},\boldsymbol{y}) - \dot{\boldsymbol{x}}_1 &\leftrightarrow \mathtt{EAL}(1), & \ldots, \quad \boldsymbol{f}_{n_{\boldsymbol{x}}}(t,\boldsymbol{x},\boldsymbol{y}) - \dot{\boldsymbol{x}}_{n_{\boldsymbol{x}}} &\leftrightarrow \mathtt{EAL}(n_{\boldsymbol{x}}), \\ \boldsymbol{g}_1(t,\boldsymbol{x},\boldsymbol{y}) &\leftrightarrow \mathtt{EAL}(1+n_{\boldsymbol{x}}), & \ldots, \quad \boldsymbol{g}_{n_{\boldsymbol{y}}}(t,\boldsymbol{x},\boldsymbol{y}) &\leftrightarrow \mathtt{EAL}(n_{\boldsymbol{y}}+n_{\boldsymbol{x}}) \,. \end{aligned}$$

Similarly, the initial *variable association list* (VAL) is assumed to be assigned as

$$\begin{aligned} \boldsymbol{x}_1 &\leftrightarrow \mathtt{VAL}(1), & \ldots, \quad \boldsymbol{x}_{n_{\boldsymbol{x}}} &\leftrightarrow \mathtt{VAL}(n_{\boldsymbol{x}}), \\ \boldsymbol{y}_1 &\leftrightarrow \mathtt{VAL}(1+n_{\boldsymbol{x}}), & \ldots, \quad \boldsymbol{y}_{n_{\boldsymbol{y}}} &\leftrightarrow \mathtt{VAL}(n_{\boldsymbol{y}}+n_{\boldsymbol{x}}) \,. \end{aligned}$$

Pantelides' Algorithm requests the total differentiation of some of the algebraic equations $\{0 = \boldsymbol{g}_1, \ldots, 0 = \boldsymbol{g}_{n_{\boldsymbol{y}}}\}$ with respect to time, giving rise to new equations and corresponding additional entries in the equation association list

$$\mathtt{EAL}(1+(n_{\boldsymbol{y}}+n_{\boldsymbol{x}})), \ldots, \mathtt{EAL}(m_{\tilde{\boldsymbol{s}}}+(n_{\boldsymbol{y}}+n_{\boldsymbol{x}})) \,,$$

with the number of additional equations given by

$$m_{\tilde{\boldsymbol{s}}} := \operatorname{card}\left\{1 \le \mu \le n_{\boldsymbol{y}} \;\middle|\; \exists \nu \in \{1,\ldots,n_{\boldsymbol{y}}\} : \mathtt{EAL}(\nu+n_{\boldsymbol{x}}) = \mu + (n_{\boldsymbol{x}}+n_{\boldsymbol{y}})\right\} .$$

Furthermore, it adds entries to the variable association list corresponding to the highest derivatives encountered in the reduced derivative arrays, i.e., $\{\dot{\boldsymbol{x}}_1, \ldots, \dot{\boldsymbol{x}}_{n_{\boldsymbol{x}}}\}$ and a subset of $\{\dot{\boldsymbol{y}}_1, \ldots, \dot{\boldsymbol{y}}_{n_{\boldsymbol{y}}}\}$, in

$$\mathtt{VAL}(1+(n_{\boldsymbol{x}}+n_{\boldsymbol{y}})), \ldots, \mathtt{VAL}(n_{\boldsymbol{x}}+n_{\tilde{\boldsymbol{z}}}+(n_{\boldsymbol{x}}+n_{\boldsymbol{y}})) \,,$$

where $n_{\tilde{\boldsymbol{z}}}$ is the number of algebraic variables which time derivatives are created by differentiation of algebraic equations. For simplification we can assume without loss of generality that the derivatives $\{\dot{\boldsymbol{x}}_1, \ldots, \dot{\boldsymbol{x}}_{n_{\boldsymbol{x}}}\}$ are assigned to $\mathtt{VAL}(1+(n_{\boldsymbol{x}}+n_{\boldsymbol{y}}))$, …, $\mathtt{VAL}(n_{\boldsymbol{x}}+(n_{\boldsymbol{x}}+n_{\boldsymbol{y}}))$. Then $n_{\tilde{\boldsymbol{z}}}$ is defined by

$$n_{\tilde{\boldsymbol{z}}} := \operatorname{card}\left\{1 \le \mu \le n_{\boldsymbol{y}} \;\middle|\; \exists \nu \in \{1,\ldots,n_{\boldsymbol{y}}\} : \mathtt{VAL}(\nu+n_{\boldsymbol{x}}) = \mu + (2n_{\boldsymbol{x}}+n_{\boldsymbol{y}})\right\} .$$

For ease of notation we introduce the real-valued injective mappings $\mathcal{P}^{\mathrm{e}} : \{1,\ldots,m_{\tilde{\boldsymbol{s}}}\} \to \{1,\ldots,n_{\boldsymbol{y}}\}$ and $\mathcal{P}^{\mathrm{v}} : \{1,\ldots,n_{\tilde{\boldsymbol{z}}}\} \to \{1,\ldots,n_{\boldsymbol{y}}\}$ by

$$\begin{aligned} \mathcal{P}^{\mathrm{e}}(\mu) &:= \left\{\nu \in \{1,\ldots,n_{\boldsymbol{y}}\} \;\middle|\; \mathtt{EAL}(\nu+n_{\boldsymbol{x}}) = \mu + (n_{\boldsymbol{x}}+n_{\boldsymbol{y}})\right\} , \\ \mathcal{P}^{\mathrm{v}}(\mu) &:= \left\{\nu \in \{1,\ldots,n_{\boldsymbol{y}}\} \;\middle|\; \mathtt{VAL}(\nu+n_{\boldsymbol{x}}) = \mu + (2n_{\boldsymbol{x}}+n_{\boldsymbol{y}})\right\} . \end{aligned}$$

I.e., $\mathcal{P}^{\mathrm{e}}(\cdot)$ points to the algebraic equations which are to be differentiated and $\mathcal{P}^{\mathrm{v}}(\cdot)$ points to the algebraic variables for which first order time derivatives are introduced in the reduced derivative array equations. Further let $\overline{\mathcal{P}^{\mathrm{e}}} : \{1,\ldots,n_{\boldsymbol{y}}-m_{\tilde{\boldsymbol{s}}}\} \to \{1,\ldots,n_{\boldsymbol{y}}\}$ and $\overline{\mathcal{P}^{\mathrm{v}}} : \{1,\ldots,n_{\boldsymbol{y}}-n_{\tilde{\boldsymbol{z}}}\} \to \{1,\ldots,n_{\boldsymbol{y}}\}$ be two real-valued injective mappings such that

$$\begin{aligned} \{\overline{\mathcal{P}^{\mathrm{e}}}(\mu); \mu = 1,\ldots,n_{\boldsymbol{y}}-m_{\tilde{\boldsymbol{s}}}\} &= \{1,\ldots,n_{\boldsymbol{y}}\} \setminus \{\mathcal{P}^{\mathrm{e}}(\mu); \mu = 1,\ldots,n_{\tilde{\boldsymbol{z}}}\} \text{ , and} \\ \{\overline{\mathcal{P}^{\mathrm{v}}}(\mu); \mu = 1,\ldots,n_{\boldsymbol{y}}-n_{\boldsymbol{z}}\} &= \{1,\ldots,n_{\boldsymbol{y}}\} \setminus \{\mathcal{P}^{\mathrm{v}}(\mu); \mu = 1,\ldots,n_{\tilde{\boldsymbol{z}}}\} \,. \end{aligned}$$

After these preparations we introduce the vectors

$$\tilde{\boldsymbol{x}} := \boldsymbol{x}\,, \quad \tilde{\boldsymbol{y}} := \begin{bmatrix} \boldsymbol{y}_{\overline{\mathcal{P}^{\mathrm{v}}}(1)} \\ \vdots \\ \boldsymbol{y}_{\overline{\mathcal{P}^{\mathrm{v}}}(n_{\boldsymbol{y}}-n_{\tilde{\boldsymbol{z}}})} \end{bmatrix}, \quad \text{and} \quad \tilde{\boldsymbol{z}} := \begin{bmatrix} \boldsymbol{y}_{\mathcal{P}^{\mathrm{v}}(1)} \\ \vdots \\ \boldsymbol{y}_{\mathcal{P}^{\mathrm{v}}(n_{\tilde{\boldsymbol{z}}})} \end{bmatrix},$$

$\tilde{\boldsymbol{x}} \in \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}} \equiv \mathbb{R}^{n_{\boldsymbol{x}}}$, $\tilde{\boldsymbol{y}} \in \mathbb{R}^{n_{\tilde{\boldsymbol{y}}}} \equiv \mathbb{R}^{n_{\boldsymbol{y}}-n_{\tilde{\boldsymbol{z}}}}$, $\tilde{\boldsymbol{z}} \in \mathbb{R}^{n_{\tilde{\boldsymbol{z}}}}$. Using these new variables, their relations to the original variables $\{\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}\}$, and the information from Pantelides' Algorithm, we define the functions

$$\tilde{\boldsymbol{f}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}) := \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{y}) - \dot{\boldsymbol{x}}\,,$$

$$\tilde{\boldsymbol{g}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}) := \begin{bmatrix} \boldsymbol{g}_{\overline{\mathcal{P}^{\mathrm{e}}}(1)}(t, \boldsymbol{x}, \boldsymbol{y}) \\ \vdots \\ \boldsymbol{g}_{\overline{\mathcal{P}^{\mathrm{e}}}(n_{\boldsymbol{y}}-m_{\tilde{\boldsymbol{s}}})}(t, \boldsymbol{x}, \boldsymbol{y}) \end{bmatrix}, \text{ and } \tilde{\boldsymbol{s}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) := \begin{bmatrix} \boldsymbol{g}_{\mathcal{P}^{\mathrm{e}}(1)}(t, \boldsymbol{x}, \boldsymbol{y}) \\ \vdots \\ \boldsymbol{g}_{\mathcal{P}^{\mathrm{e}}(m_{\tilde{\boldsymbol{s}}})}(t, \boldsymbol{x}, \boldsymbol{y}) \end{bmatrix},$$

where $\tilde{\boldsymbol{f}} : \mathbb{R}^{1+n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{y}}}+n_{\tilde{\boldsymbol{z}}}+n_{\tilde{\boldsymbol{x}}}} \to \mathbb{R}^{m_{\tilde{\boldsymbol{f}}}} \equiv \mathbb{R}^{n_{\boldsymbol{x}}}$, $\tilde{\boldsymbol{g}} : \mathbb{R}^{1+n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{y}}}+n_{\tilde{\boldsymbol{z}}}} \to \mathbb{R}^{m_{\tilde{\boldsymbol{g}}}} \equiv \mathbb{R}^{n_{\boldsymbol{y}}-m_{\tilde{\boldsymbol{s}}}}$, and $\tilde{\boldsymbol{s}} : \mathbb{R}^{1+n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{z}}}} \to \mathbb{R}^{m_{\tilde{\boldsymbol{s}}}}$.

Remark 3.14:
The function $\tilde{\boldsymbol{s}}$ cannot depend on $\tilde{\boldsymbol{y}}$:

The subset $\tilde{\boldsymbol{z}}$ of $\boldsymbol{y}$ contains all originally algebraic variables for which derivatives $\dot{\tilde{\boldsymbol{z}}}$ are introduced in the reduced derivative array equations. $\tilde{\boldsymbol{y}}$ is the complement of $\tilde{\boldsymbol{z}}$ in $\boldsymbol{y}$. Thus, no element of $\tilde{\boldsymbol{y}}$ can enter $\tilde{\boldsymbol{s}}$ as by the total differentiation of $\tilde{\boldsymbol{s}}$ with respect to time the first order time derivatives of all arguments of $\tilde{\boldsymbol{s}}$ are accessed, cf. Eq. (3.36d) below. ◊

Altogether, the reduced derivative array equations of the semi-explicit index-2 DAE Eqs. (3.35a)–(3.35b) resulting from Pantelides' Algorithm in the variables $[t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}]$ are given by

$$0 = \tilde{\boldsymbol{f}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}})\,, \tag{3.36a}$$

$$0 = \tilde{\boldsymbol{g}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})\,, \tag{3.36b}$$

$$0 = \tilde{\boldsymbol{s}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}})\,, \tag{3.36c}$$

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}\Big(\tilde{\boldsymbol{s}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}})\Big) = \frac{\partial \tilde{\boldsymbol{s}}}{\partial t}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{z}}}\,. \tag{3.36d}$$

The Jacobian of the right hand side of Eqs. (3.36a)–(3.36d) with respect to $[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}]$ is

$$\begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{y}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{z}}}\right] & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \mathbf{0} \\ \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{y}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{z}}}\right] & \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}\right] & \mathbf{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}\right] & \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial}{\partial \tilde{\boldsymbol{x}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right] & \mathbf{0} & \left[\frac{\partial}{\partial \tilde{\boldsymbol{z}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}\right] \end{bmatrix}_{t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}}. \tag{3.37}$$

Here, $\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}}$ is the identity in $\mathbb{R}^{n_{\tilde{\boldsymbol{x}}}}$, and $\mathbf{0}$ are zero matrices of appropriate dimension.

Remark 3.15:
SPALG returns a slightly different characterisation of the reduced derivative array equations derived by Pantelides' Algorithm than the one we have described above. In order to be independent from a specific implementation of Pantelides' Algorithm we stick to the notation of [Pant 88a]. See also Section 3.1.5. ◊

3.2.2 A Regularity Assumption on the Index-1 Subsystem

Consider the reduced system of derivative array equations Eqs. (3.36a)–(3.36d) obtained from Pantelides' Algorithm in the previous Section 3.2.1. Then by dropping Eq. (3.36c) an index-1 DAE can be formulated which is analytically equivalent to the index-2 DAE Eqs. (3.35a)–(3.35b) ([Pant 88a], cf. the discussion in Section 3.1.5 of this treatise):

$$0 = \tilde{\boldsymbol{f}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}})\,, \tag{3.38a}$$

$$0 = \tilde{\boldsymbol{g}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})\,, \tag{3.38b}$$

$$0 = \frac{\partial \tilde{\boldsymbol{s}}}{\partial t}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{z}}}\,. \tag{3.38c}$$

Due to the structural criteria applied in Pantelides' Algorithm the following *structural property* holds for the Jacobian of Eqs. (3.38a)–(3.38c) with respect to the highest order derivatives $\tilde{\boldsymbol{y}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}$ of the variables present in this system:

$$\operatorname{rank}\left(\operatorname{pat}\left(\begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}_n}{\partial \tilde{\boldsymbol{y}}}\right] & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \boldsymbol{0} \\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n}{\partial \tilde{\boldsymbol{y}}}\right] & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}_n}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}_n}{\partial \tilde{\boldsymbol{z}}}\right] \end{bmatrix}_{t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}}\right)\right) = n_{\tilde{\boldsymbol{x}}} + n_{\tilde{\boldsymbol{y}}} + n_{\tilde{\boldsymbol{z}}}\,. \tag{3.39}$$

However, as we intend to develop a numerical algorithm for the consistent initialisation of higher index DAEs based on Pantelides' Algorithm we have to make the *assumption* that this matrix is also *numerically regular* in a sufficiently large neighbourhood of the consistent initial conditions, i.e,

$$\det\left(\begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}_n}{\partial \tilde{\boldsymbol{y}}}\right] & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \boldsymbol{0} \\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n}{\partial \tilde{\boldsymbol{y}}}\right] & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}_n}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}_n}{\partial \tilde{\boldsymbol{z}}}\right] \end{bmatrix}_{t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}}\right) \neq 0\,, \tag{3.40}$$

which is stronger than the guaranteed property Eq. (3.39).

Remark 3.16:
This rank condition is better known for DAEs of the form $0 = \boldsymbol{F}(t, \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{x}}(t))$, $\boldsymbol{F}$:

$\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\dot{\boldsymbol{x}}}} \to \mathbb{R}^{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}$:

$$\operatorname{rank}\left(\left[\left[\tfrac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}\right]\ \left[\tfrac{\partial \boldsymbol{F}}{\partial \dot{\boldsymbol{x}}}\right]\right]_{t,\boldsymbol{x},\boldsymbol{y},\dot{\boldsymbol{x}}}\right) \stackrel{!}{=} n_{\boldsymbol{x}} + n_{\boldsymbol{y}}\,. \tag{3.41}$$

Eq. (3.41) is a sufficient condition for the system either to be an ODE (index-0 DAE) or an index-1 DAE. This condition is commonly used in order to restrict to problems of this class (e.g., [GPS 95], [GFB 99]). ◊

Reordering in Eq. (3.40) gives

$$\det\left(\left[\begin{array}{ccc}\left[\frac{\partial \tilde{\boldsymbol{s}}_n}{\partial \tilde{\boldsymbol{z}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}_n}{\partial \tilde{\boldsymbol{x}}}\right] & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \left[\frac{\partial \tilde{\boldsymbol{f}}_n}{\partial \tilde{\boldsymbol{y}}}\right]\\ \boldsymbol{0} & \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{g}}_n}{\partial \tilde{\boldsymbol{y}}}\right]\end{array}\right]_{t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}}}\right) \neq 0\,. \tag{3.42}$$

From Eq. (3.42) we conclude that $[\partial \tilde{\boldsymbol{s}}_n/\partial \tilde{\boldsymbol{z}}]$ is of full column rank[(vi)]; this reflects that $\dot{\tilde{\boldsymbol{z}}}$ has to be determined by Eq. (3.38c). Especially we have

$$m_{\tilde{\boldsymbol{s}}} \geq n_{\tilde{\boldsymbol{z}}}\,. \tag{3.43}$$

Additionally, Eq. (3.42) necessitates

$$m_{\tilde{\boldsymbol{s}}} \leq n_{\tilde{\boldsymbol{x}}} + n_{\tilde{\boldsymbol{z}}}\ : \tag{3.44}$$

Assume $m_{\tilde{\boldsymbol{s}}} > n_{\tilde{\boldsymbol{x}}} + n_{\tilde{\boldsymbol{z}}}$. Then the first $m_{\tilde{\boldsymbol{s}}}$ rows $[\,[\partial \tilde{\boldsymbol{s}}_n/\partial \tilde{\boldsymbol{z}}]\ [\partial \tilde{\boldsymbol{s}}_n/\partial \tilde{\boldsymbol{x}}]\ \boldsymbol{0}\,]$ of the matrix in Eq. (3.42) are necessarily linearly dependent, in contradiction to Eq. (3.42); this linear dependency further implies that the equations $\tilde{\boldsymbol{s}} \subseteq \boldsymbol{g}$ (Eq. (3.36c)) are functionally dependent, i.e., that already the DAE system Eqs. (3.35a)–(3.35b) is ill-posed.

Remark 3.17:
As will be discussed in Section 3.2.4, $n_{\boldsymbol{x}} - n_{\mathrm{ddf}} = m_{\tilde{\boldsymbol{s}}} - n_{\tilde{\boldsymbol{z}}}$ dynamic degrees of freedom are fixed by the algebraic constraints causing the higher index. The special case $m_{\tilde{\boldsymbol{s}}} = 0$ indicates either an ODE or an index-1 DAE; $m_{\tilde{\boldsymbol{s}}} = n_{\tilde{\boldsymbol{z}}} > 0$ indicates that Pantelides' Algorithm unnecessarily has requested differentiations, attempting to turn an index-1 DAE into an ODE. ◊

3.2.3 Setting Up the Reduced Derivative Array Equations

In the next step we set up the reduced derivative array equations Eqs. (3.36a)–(3.36d) and their Jacobian Eq. (3.37). Assuming that no symbolic manipulations can be performed the reliable and fast generation of both the total time derivative as well as of the Jacobian of Eq. (3.36d) is the main issue.

[(vi)] $A \in \mathbb{R}^{\mu\times\nu}$ has full column rank $\nu \Leftrightarrow \{\zeta \in \mathbb{R}^{\nu} \setminus \{0\} \mid A\zeta = 0\} = \emptyset$

3.2.3.a Full Approximation According to Leimkuhler

At first we consider full numerical approximation of $\mathrm{d}\tilde{\boldsymbol{s}}(t,\tilde{\boldsymbol{x}}(t),\tilde{\boldsymbol{z}}(t))/\mathrm{d}t$. Recall the family of approximations Eq. (3.7)

$$\left[D_h^k\overline{\boldsymbol{F}}\right](\boldsymbol{\Xi}^{(0)},\ldots,\boldsymbol{\Xi}^{(\lambda_\mathrm{t})}) := \frac{k!}{h^k}\left\{\sum_{\nu=0}^{\lambda_\mathrm{s}}\alpha_\nu\overline{\boldsymbol{F}}\left(\sum_{\mu=0}^{\lambda_\mathrm{t}}\frac{(c_\nu h)^\mu}{\mu!}\boldsymbol{\Xi}^{(\mu)}\right)\right\},$$

which we have already discussed in Section 3.1.3. Here, $\boldsymbol{\Xi}^{(\mu)} := d^\mu\boldsymbol{\Xi}(t)/dt^\mu$, $\mu = 1,\ldots,\lambda_\mathrm{t}$, is the value of the state vector and its (higher order) time derivatives at a fixed point in time t, $\lambda_\mathrm{s} \in \mathbb{N}$ is the number of samples taken for linear combination, and $\lambda_\mathrm{t} \in \mathbb{N}$ is the number of terms considered for the Taylor's series expansion of the state vector trajectories at the point of differentiation.

Directly applied to $\tilde{\boldsymbol{s}}$ we obtain an approximation of its first total time derivative as

$$\left[D_h^1\tilde{\boldsymbol{s}}\right](t,\tilde{\boldsymbol{x}}^{(0)},\tilde{\boldsymbol{z}}^{(0)},\ldots,\tilde{\boldsymbol{x}}^{(\lambda_\mathrm{t})},\tilde{\boldsymbol{z}}^{(\lambda_\mathrm{t})}) := \frac{1}{h}\left\{\sum_{\nu=0}^{\lambda_\mathrm{s}}\alpha_\nu\tilde{\boldsymbol{s}}\left(t+c_\nu h,\sum_{\mu=0}^{\lambda_\mathrm{t}}\frac{(c_\nu h)^\mu}{\mu!}\tilde{\boldsymbol{x}}^{(\mu)},\sum_{\mu=0}^{\lambda_\mathrm{t}}\frac{(c_\nu h)^\mu}{\mu!}\tilde{\boldsymbol{z}}^{(\mu)}\right)\right\}.$$

By Theorem 3.1 it is sufficient to use $\lambda_\mathrm{t} = 1$ as we are approximating the 1^{st} total time derivative:

$$\left[D_h^1\tilde{\boldsymbol{s}}\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}}) := \frac{1}{h}\left\{\sum_{\nu=0}^{\lambda_\mathrm{s}}\alpha_\nu\tilde{\boldsymbol{s}}\left(t+c_\nu h,\tilde{\boldsymbol{x}}+(c_\nu h)\dot{\tilde{\boldsymbol{x}}},\tilde{\boldsymbol{z}}+(c_\nu h)\dot{\tilde{\boldsymbol{z}}}\right)\right\}. \quad (3.45)$$

Few algebraic transformations show that the partial Jacobians of $[D_h^1\tilde{\boldsymbol{s}}](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}})$ with respect to the variables $[\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}}]$ are given by

$$\left[\frac{\partial D_h^1\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{x}}}\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}}) = D_h^1\left(\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{x}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right), \quad (3.46a)$$

$$\left[\frac{\partial D_h^1\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{z}}}\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}}) = D_h^1\left(\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{z}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right), \quad (3.46b)$$

$$\left[\frac{\partial D_h^1\tilde{\boldsymbol{s}}}{\partial\dot{\tilde{\boldsymbol{x}}}}\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}}) = \left[\sum_{\nu=0}^{\lambda_\mathrm{s}}\alpha_\nu c_\nu\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{x}}}\left(t+c_\nu h,\tilde{\boldsymbol{x}}+(c_\nu h)\dot{\tilde{\boldsymbol{x}}},\tilde{\boldsymbol{z}}+(c_\nu h)\dot{\tilde{\boldsymbol{z}}}\right)\right], \quad (3.46c)$$

$$\left[\frac{\partial D_h^1\tilde{\boldsymbol{s}}}{\partial\dot{\tilde{\boldsymbol{z}}}}\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}}) = \left[\sum_{\nu=0}^{\lambda_\mathrm{s}}\alpha_\nu c_\nu\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{z}}}\left(t+c_\nu h,\tilde{\boldsymbol{x}}+(c_\nu h)\dot{\tilde{\boldsymbol{x}}},\tilde{\boldsymbol{z}}+(c_\nu h)\dot{\tilde{\boldsymbol{z}}},\right)\right]. \quad (3.46d)$$

Remark 3.18:
As $c_0 := 0$ the contribution for $\nu = 0$ to the sums in Eq. (3.46c) and Eq. (3.46d) vanishes. ◊

Finally, we have to specify the number of samples λ_s. Given a sufficiently smooth DAE Theorem 3.2 shows that the maximum attainable order of approximation for a first total derivative is $\lambda_s - 1 + 1 = \lambda_s$. We use $\lambda_s = 3$ stages. Furthermore, we use equidistant displacements $c_\nu = \nu$, $\nu = 0, \ldots, \lambda_s$, i.e., the coefficients α_ν, $\nu = 0, \ldots, \lambda_s$, are as specified in Table 3.1 on page 85.

3.2.3.b Algorithmic Amendments to Leimkuhler's Approximations

In our numerical tests we have found that the proper selection of the disturbance parameter h – which is not connected to any other quantity used in a DAE integrator – is of major importance for the success of the overall algorithm. However, in the original publications [Leim 88], [LPG 91] no rule has been developed for an appropriate choice of this parameter.

One problem that we have seen originates from Eq. (3.45) only admitting one fixed value h for *all* variables. For demonstration, we use an example function $\eta(\boldsymbol{\zeta})$, $\eta \in \mathcal{C}^1(\mathbb{R}^{n_\zeta}, \mathbb{R}^{m_\eta})$, and a trajectory $\boldsymbol{\zeta} : t \to \boldsymbol{\zeta}(t)$, $\boldsymbol{\zeta} \in \mathcal{C}^1(\mathbb{R}, \mathbb{R}^{n_\zeta})$:

Leimkuhler's approximation to the first total time derivative along $\boldsymbol{\zeta}(t)$ for $\lambda_t = 1$ is

$$\left[D_h^1 \eta\right](\boldsymbol{\zeta}, \dot{\boldsymbol{\zeta}}) = \frac{1}{h} \left\{ \sum_{\nu=0}^{\lambda_s} \alpha_\nu \eta\left(\boldsymbol{\zeta} + (c_\nu h)\dot{\boldsymbol{\zeta}}\right) \right\} . \tag{3.47}$$

From a numerical point of view it is necessary that the disturbance $(c_\nu h)\dot{\boldsymbol{\zeta}}_\mu$, $\mu = 1, \ldots, n_\zeta$, $\nu = 0, \ldots, \lambda_s$, in *each* component of the argument of η is not dropped due to round-off errors as otherwise the result of the finite difference approximation can be without meaning. E.g., in the very simple case of $\eta(\boldsymbol{\zeta}) := \boldsymbol{\zeta} \in \mathbb{R}$ with $\boldsymbol{\zeta}(t_0) := 1.0 \cdot 10^6$, $\dot{\boldsymbol{\zeta}}(t_0) := 1.0 \cdot 10^{-3}$, $c_\nu = \mathcal{O}(1)$, $\nu = 1, \ldots, \lambda_s$, and $h := 1.0 \cdot 10^{-4}$ the result of $D_h^1 \eta\big|_{t_0}$ in single precision arithmetics is 0 instead of $1.0 \cdot 10^{-3} (= \dot{\boldsymbol{\zeta}}(t_0))$. Such a result can severely harm the solution of the approximated consistency equations.

Additionally, for a numerically evaluated function η the disturbance in the argument must be sufficiently large in order to obtain variations in the function values that are not lost due to round-off.

As an ad-hoc measurement we split the vector $\boldsymbol{\zeta}$ in several components and compute the difference approximation in each component separately:

Let $\omega = [\omega_1, \ldots, \omega_{n_\omega}]^T \in \mathbb{R}^{n_\omega}$ contain samples of the estimated range in all components of $\boldsymbol{\zeta}$, including a neutral sample $\omega_\theta = 1$, $\theta \in \{1, \ldots, n_\omega\}$. n_ω should be of moderate size, e.g., $\omega := [1.0 \cdot 10^4, 1.0 \cdot 10^0, 1.0 \cdot 10^{-3}, 1.0 \cdot 10^{-6}, 1.0 \cdot 10^{-9}, 1.0 \cdot 10^{-12}]^T$. Further let $\delta_{\min} \in \mathbb{R}_+ \backslash \{0\}$ be the desired relative disturbance in each component, e.g., $\delta_{\min} := \sqrt[4]{\epsilon_{\text{mach}}}$, where ϵ_{mach} is the machine precision. Then we define the disjoint decomposition $\Omega_1 \oplus \cdots \oplus \Omega_{n_\omega}$ of $\{1, \ldots, n_\zeta\}$ for a given standard

disturbance parameter $h \in \mathbb{R}_+$ according to the following Algorithm 7:

Algorithm 7 (Decomposition of Variables for Numerical Differentiation)

1. *Initialise*
   $\Omega_\mu := \emptyset$, $\mu = 1, \ldots, n_\omega$.

2. *For each* $\nu = 1, \ldots, n_\zeta$:

   (a) *if* $\left|\dot{\zeta}_\nu\right| < \epsilon_{mach}$ *then*
   $\mu := \theta$.

   (b) *else if* $\frac{1}{\omega_{n_\omega}} h \left|\dot{\zeta}_\nu\right| < \delta_{min} \cdot \max(|\zeta_\nu|, 1.0)$ *then*
   $\mu := \theta$.

   (c) *else*
   $\mu := \min\left\{\mu \in \{1, \ldots, n_\omega\} : \frac{1}{\omega_\mu} h \left|\dot{\zeta}_\nu\right| \geq \delta_{min} \cdot \max(|\zeta_\nu|, 1.0)\right\}$.
   end if

   (d) $\Omega_\mu := \Omega_\mu \cup \{\nu\}$.

Step **2a** is employed in order to avoid overscaling in case of very small values of $|\dot{\zeta}|$. Rule **2b** becomes active when $|\dot{\zeta}|$ is extremely small in relation to $|\zeta_\nu|$. In both cases the corresponding variable is added to the neutral set Ω_θ, i.e., in effect it is dropped. Step **2c** contains the main rule. The max-operator in the terms $\max(|\zeta_\nu|, 1.0)$ guarantees that the minimum *absolute* disturbance is of order $\delta_{\min}$ for a small value of a component ζ_ν, $\nu \in \{1, \ldots, n_\zeta\}$.

Remark 3.19:
We ignore the factors c_ν, $\nu = 1, \ldots, \lambda_s$, in Leimkuhler's approximation as they can be chosen to be of order unity. ◊

Remark 3.20:
The decomposition of the variables can be utilised in order to avoid problems due to the violation of a priori known bounds on the disturbed variables. In a critical case the variable is assigned to a subdivision with a smaller disturbance. This is of special importance for models of chemical engineering processes where physical bounds on the variables have to be considered. Otherwise the evaluation of the model equations that are to be differentiated may not be possible at the disturbed points (e.g., the square root of a negative number is not defined in $\mathbb{R}$). ◊

Using the decomposition of the vector $\dot{\zeta}$ according to Algorithm 7 we split the approximation formula Eq. (3.47) introducing the modified approximation $\hat{D}_h^1$ to the approximated pseudo-derivative operator D_h^1

$$\left[\hat{D}_h^1 \eta\right](\zeta, \dot{\zeta}) := \sum_{\mu=1}^{n_\omega} \frac{\omega_\mu}{h} \left\{ \sum_{\nu=0}^{\lambda_s} \alpha_\nu \eta \left(\zeta + c_\nu \frac{h}{\omega_\mu} P^\mu \dot{\zeta} \right) \right\}, \tag{3.48}$$

where $P^\mu \in \mathbb{R}^{n_\zeta \times n_\zeta}$, $\mu = 1, \ldots, n_\omega$, are diagonal matrices

$$P^\mu := \operatorname{diag}(\varpi_1^\mu, \ldots, \varpi_{n_\zeta}^\mu); \quad \varpi_\nu^\mu := \begin{cases} 1; & \nu \in \Omega_\mu \\ 0; & \text{otherwise} \end{cases} ; \ \nu = 1, \ldots, n_\zeta .$$

The partial derivatives required for the construction of the Jacobian of the approximated consistency equations are then given by

$$\left[\frac{\partial}{\partial \boldsymbol{\zeta}} \hat{D}_h^1 \eta\right](\boldsymbol{\zeta}, \dot{\boldsymbol{\zeta}}) = \sum_{\mu=1}^{n_\omega} \frac{\omega_\mu}{h} \left\{ \sum_{\nu=0}^{\lambda_s} \alpha_\nu \frac{\partial \eta}{\partial \boldsymbol{\zeta}} \left(\boldsymbol{\zeta} + c_\nu \frac{h}{\omega_\mu} P^\mu \dot{\boldsymbol{\zeta}} \right) \right\} = \left[\hat{D}_h^1 \frac{\partial \eta}{\partial \boldsymbol{\zeta}}\right](\boldsymbol{\zeta}, \dot{\boldsymbol{\zeta}}) , \tag{3.49a}$$

$$\left[\frac{\partial}{\partial \dot{\boldsymbol{\zeta}}} \hat{D}_h^1 \eta\right](\boldsymbol{\zeta}, \dot{\boldsymbol{\zeta}}) = \sum_{\mu=1}^{n_\omega} \left\{ \sum_{\nu=0}^{\lambda_s} \alpha_\nu c_\nu \frac{\partial \eta}{\partial \boldsymbol{\zeta}} \left(\boldsymbol{\zeta} + c_\nu \frac{h}{\omega_\mu} P^\mu \dot{\boldsymbol{\zeta}} \right) \right\} P^\mu . \tag{3.49b}$$

The drawback of Eq. (3.48) and Eqs. (3.49a)–(3.49b) is the increased number of function and Jacobian evaluations required for an evaluation of the residual of the reduced consistency equations and of its Jacobian, respectively.

According to our experience frequently the computational effort can be reduced by skipping empty sets $\{\Omega_\mu, \mu = 1, \ldots, n_\omega | \Omega_\mu = \emptyset\}$. This amounts to a simple modification of the outer summations in Eq. (3.48) and Eqs. (3.49a)–(3.49b) as

$$\sum_{\mu=1}^{n_\omega} \cdots \quad \curvearrowright \quad \sum_{\substack{\mu=1,\ldots,n_\omega \\ \Omega_\mu \neq \emptyset}} \cdots .$$

By splitting the variables in Eq. (3.48) we can achieve a suitable ratio of the disturbance to the undisturbed function arguments. However, the approximation formulae still lack of an error adapted choice of the disturbance parameter h. This is one of the main deficiencies of Leimkuhler's developments. Therefore, we now develop a basic error control suitable for our needs.

Consider the terms of the outer sum on the right hand side of Eq. (3.48). For $\mu \in \{1, \ldots, n_\omega\}$ and fixed but otherwise arbitrary $\boldsymbol{\zeta}$ and $\dot{\boldsymbol{\zeta}}$ we define the functions

$$\boldsymbol{\Theta}_\mu(h_\mu) := \omega_\mu \left\{ \sum_{\nu=1}^{\lambda_s} \alpha_\nu \eta \left(\boldsymbol{\zeta} + c_\nu \frac{h_\mu}{\omega_\mu} P^\mu \dot{\boldsymbol{\zeta}} \right) \right\} ,$$

$\boldsymbol{\Theta}_\mu : \mathbb{R} \to \mathbb{R}^{m_\eta}$. Note that the summation now starts with 1, instead of 0 before (the 0^{th} term contains the undisturbed function evaluation). Then differentiation with respect to h_μ at $h_\mu = 0$ gives

$$\frac{\mathrm{d}\boldsymbol{\Theta}_\mu}{\mathrm{d}h_\mu}(0) = \left[\omega_\mu \left\{ \sum_{\nu=1}^{\lambda_s} \alpha_\nu \left[\frac{\partial \eta}{\partial \boldsymbol{\zeta}} \left(\boldsymbol{\zeta} + c_\nu \frac{h_\mu}{\omega_\mu} P^\mu \dot{\boldsymbol{\zeta}} \right) \right] \frac{c_\nu}{\omega_\mu} P^\mu \dot{\boldsymbol{\zeta}} \right\} \right]_{h_\mu = 0}$$

$$= \sum_{\nu=1}^{\lambda_s} \alpha_\nu c_\nu \left[\frac{\partial \eta}{\partial \zeta}(\zeta)\right] P^\mu \dot{\zeta} \overset{Eq.\ (3.8)}{=} \left[\frac{\partial \eta}{\partial \zeta}(\zeta)\right] P^\mu \dot{\zeta} \,. \tag{3.50}$$

For the error controlled approximation of the derivatives $\mathrm{d}\Theta_\mu/\mathrm{d}h_\mu(0)$ algorithms are known, e.g., HSL routine TD12 [AEA 93] which implements the results of [CPR 74] and [CuRe 74]. TD12 attempts to select the disturbance used for building the finite differences such that a suitable ratio between round-off error and truncation error is achieved [CuRe 74], cf. Remark 3.21 below. The resulting optimal disturbance is then at the same time a suitable choice for h_μ in Leimkuhler's approximation. Finally,

$$\frac{\mathrm{d}\eta(\zeta)}{\mathrm{d}t} = \left[\frac{\partial \eta}{\partial \zeta}(\zeta)\right] \dot{\zeta} = \sum_{\mu=1}^{n_\omega} \left[\frac{\partial \eta}{\partial \zeta}(\zeta)\right] P^\mu \dot{\zeta} \overset{Eq.\ (3.50)}{=} \sum_{\mu=1}^{n_\omega} \frac{\mathrm{d}\Theta_\mu}{\mathrm{d}h_\mu}(0) \,,$$

so that summation of the terms $\mathrm{d}\Theta_\mu/\mathrm{d}h_\mu(0)$, $\mu = 1, \ldots, n_\omega$, obtained from error controlled finite differences by TD12 provides an error controlled approximation of $\mathrm{d}\eta(\zeta)/\mathrm{d}t$.

Remark 3.21:
It is essential to define suitable upper and lower bounds for the disturbances h, or h_μ, $\mu = 1, \ldots, n_\omega$, respectively. Otherwise the influence of single entries in the residual which can be evaluated with lower precision only adversely affect overall precision and performance of the method. As our version of TD12 did not support user specified lower bounds on the disturbance we extended it with such a feature.

Additionally, we found that in some tests with problems from our area of application the performance of the disturbance selection strategy employed in TD12 was poor. Therefore, we modified the error estimates proposed in [CuRe 74] which are the basis for the disturbance adaptation. Especially we included a parameter describing the precision of the function evaluation as this source of numerical error has not been considered there.

At first we examine the error estimation strategy of TD12 when it is applied to generate $\Theta'_\mu(0) := \mathrm{d}\Theta_\mu/\mathrm{d}h_\mu(0)$; the current value of the disturbance parameter is $h > 0$. In TD12 for each component function $\left[\Theta'_\mu(0)\right]_\nu$, $\nu = 1, \ldots, m_\eta$, of the derivative $\Theta'_\mu(0) \in \mathbb{R}^{m_\eta}$ the truncation error is estimated by the difference between the first and second order difference approximation to the first order derivative

$$\left[\mathrm{err}_{\mathrm{trc}}\right]_\nu := \left| \left[\frac{\Theta_\mu(0) - \Theta_\mu(-h)}{-h} - \frac{\Theta_\mu(h) - \Theta_\mu(-h)}{2h} \right]_\nu \right| . \tag{3.51}$$

The round-off errors are estimated according to

$$\left[\mathrm{err}_{\mathrm{rnd}}\right]_\nu := \frac{\epsilon_{\mathrm{mach}}}{h} \Big\{ 0.5 \cdot \left(\left|\left[\Theta_\mu(h)\right]_\nu\right| + \left|\left[\Theta_\mu(-h)\right]_\nu\right| \right) + \max\left\{ \left| \left[\Theta_\mu^{\mathrm{fwd}}(0)\right]_\nu \right|, \left| \left[\Theta_\mu^{\mathrm{bwd}}(0)\right]_\nu \right| \right\} \cdot |0 + h| \Big\}, \quad \nu = 1, \ldots, m_\eta \,,$$

where

$$\mathbf{\Theta}_\mu^{\mathrm{fwd}}(0) := \frac{\mathbf{\Theta}_\mu(h) - \mathbf{\Theta}_\mu(0)}{h}, \quad \text{and} \quad \mathbf{\Theta}_\mu^{\mathrm{bwd}}(0) := \frac{\mathbf{\Theta}_\mu(0) - \mathbf{\Theta}_\mu(-h)}{-h}$$

denote the first order forward and backward difference quotients. Based on the ratios

$$\kappa_\nu := \begin{cases} [\mathrm{err}_{\mathrm{trc}}]_\nu \,/\, [\mathrm{err}_{\mathrm{rnd}}]_\nu \;; & [\mathrm{err}_{\mathrm{rnd}}]_\nu > 0\,, \\ [\mathrm{err}_{\mathrm{trc}}]_\nu \;; & \text{otherwise}\,, \end{cases} \tag{3.52}$$

TD12 determines the disturbance parameter h. The strategy is to keep the maximum of all κ_ν within a certain range. [CuRe 74] consider

$$10 \leq \max_{\nu=1,\ldots,m_\eta} \{\kappa_\nu\} \leq 1000$$

as acceptable. Due to the choice of the maximum ratio the disturbance parameter is determined by the most nonlinear component function [CuRe 74].

In the context of our application according to our experience the ratios κ_ν do not take into account properly the error in the function evaluations. Taking into account the limited function precision in the formulae used for truncation and round-off error estimation we obtain

$$\tilde{\kappa} = \frac{[\mathrm{err}_{\mathrm{trc}}]_\nu + \frac{3\epsilon_{\mathrm{fct}}}{h}}{\frac{4\epsilon_{\mathrm{fct}}}{h}\left(1 + \frac{h}{\epsilon_{\mathrm{mach}}}[\mathrm{err}_{\mathrm{rnd}}]_\nu\right)}, \tag{3.53}$$

where $\epsilon_{\mathrm{fct}} \geq \epsilon_{\mathrm{mach}}$ is the absolute error in the function evaluations [GMW 95]. As a side effect complications in the special case of a vanishing value of $[\mathrm{err}_{\mathrm{rnd}}]_\nu$ in Eq. (3.52) are avoided. The new estimate Eq. (3.53) can be regarded as a blend between the estimate utilised by TD12 and the estimate proposed by [GMSW 83], [GMW 95] who consider the ratio

$$\frac{1}{\hat{\kappa}_\nu} := \frac{4\epsilon_{\mathrm{fct}}}{h^2 \left|\left[\mathbf{\Theta}_\mu''(0)\right]_\nu\right|}.$$

A simple reformulation of Eq. (3.51) shows that $[\mathrm{err}_{\mathrm{trc}}]_\nu \approx (h/2) \cdot \left|\left[\mathbf{\Theta}_\mu''(0)\right]_\nu\right|$. Therefore, in our notation

$$\hat{\kappa}_\nu = \frac{[\mathrm{err}_{\mathrm{trc}}]_\nu}{\frac{2\epsilon_{\mathrm{fct}}}{h}}.$$

◊

3.2.3.c Partial Application of Leimkuhler's Approximations

In this section we consider a second method for the evaluation of the reduced derivative array equations and of their Jacobian. Short inspection of the beginning of Section 3.2 shows that we have access to all data required for a direct evaluation of

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) = \frac{\partial \tilde{\boldsymbol{s}}}{\partial t}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{z}}}\,. \tag{3.54}$$

The overall cost for setting up the reduced derivative array equations using Eq. (3.54) instead of, e.g., Eq. (3.45), reduces to a single evaluation of the Jacobian and of

the residual of the DAE.

A *partially approximated* Jacobian of $\mathrm{d}\tilde{\boldsymbol{s}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})/\mathrm{d}t$ can then be found in

$$\left[\frac{\partial}{\partial\tilde{\boldsymbol{x}}}\left(\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right)\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}}) \approx \hat{D}_h^1\left(\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{x}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right), \tag{3.55a}$$

$$\left[\frac{\partial}{\partial\tilde{\boldsymbol{z}}}\left(\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right)\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}}) \approx \hat{D}_h^1\left(\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{z}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right), \tag{3.55b}$$

$$\left[\frac{\partial}{\partial\dot{\tilde{\boldsymbol{x}}}}\left(\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right)\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}}) = \frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{x}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}}), \tag{3.55c}$$

$$\left[\frac{\partial}{\partial\dot{\tilde{\boldsymbol{z}}}}\left(\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}})\right)\right](t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}}) = \frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{z}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}}). \tag{3.55d}$$

The approximation of the Jacobians in Eqs. (3.55a)–(3.55b) by Leimkuhler's formulae is based on

$$\frac{\partial}{\partial\tilde{\boldsymbol{x}}}\left(\frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{x}}}\right), \quad \text{and} \quad \frac{\partial}{\partial\tilde{\boldsymbol{z}}}\left(\frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial\tilde{\boldsymbol{s}}}{\partial\tilde{\boldsymbol{z}}}\right),$$

which holds if $\tilde{\boldsymbol{s}}$ is at least of class $\mathcal{C}^2$ with respect to its arguments t, $\tilde{\boldsymbol{x}}$, and $\tilde{\boldsymbol{z}}$.

Remark 3.22:
The solution of the reduced consistency equations based on Eq. (3.54) by a Newton method using the partial Jacobians Eqs. (3.55a)–(3.55d) represents a quasi-Newton method. ◊

3.2.3.d Economic Evaluation of Model Residual and Jacobian

By default the underlying simulator OPTISIM® provides the right hand side $\boldsymbol{f}(t,\boldsymbol{x},\boldsymbol{y})$ and $\boldsymbol{g}(t,\boldsymbol{x},\boldsymbol{y})$, and the Jacobian of the DAE as a whole. I.e., every unit in the flowsheet is evaluated. However, in the context of consistent initialisation nearly all function and Jacobian evaluations are needed for setting up Leimkuhler's approximations to the total time differentials of the subset $\tilde{\boldsymbol{s}}$ of the algebraic equations Eq. (3.45) and their Jacobian Eqs. (3.46a)–(3.46d) if using the full approximation. The same holds for setting up the approximated parts Eqs. (3.55a)–(3.55b) of the Jacobian of Eq. (3.54) by Leimkuhler's approach in the partial approximation mode. The number of residual and Jacobian evaluations required for Leimkuhler's approximations is further increased by our extensions proposed in Section 3.2.3.b, i.e., by the splitting of the variables and by the error control.

In order to avoid unnecessary and time consuming evaluations of the residual or of the Jacobian for units in the flowsheet we analyse flowsheet and DAE, and set up a list containing the units that generate $\tilde{\boldsymbol{s}}$. This list is passed on to the unit evaluation management routine within OPTISIM® which then only evaluates units that are members of the list.

In our experience for problems from application the number of units in this list is small in relation to the entire flowsheet. Thus considerable savings can be achieved by this additional algorithmic effort. The speedup is especially attributed

to the exclusion of units expensive in evaluation, e.g., heat exchangers.

3.2.4 Specification of Transition Conditions

Unus sed leo.
One, but (it is) a lion.
(The lioness to the vixen who boasted about her having many cubs when the lioness only had one.)

Translated from Aesop (Fabulae 194)

In Section 3.2.1 a non-redundant system of reduced derivative array equations Eqs. (3.36a)–(3.36d) has been detected. As discussed in the previous Section 3.2.3 the differentiated equations Eq. (3.36d) can be numerically approximated according to Leimkuhler's approach as denoted in, e.g., Eq. (3.45). If the Jacobian of the algebraic equations Eq. (3.36c) is available their total differentials with respect to time can also be evaluated directly. In order to distinguish between algebraic formulation and numerical evaluation we refer in this case to Eq. (3.54) which actually only restates Eq. (3.36d). Eqs. (3.36a)–(3.36c) together with Eq. (3.45) or Eq. (3.54) set up an (in general) underdetermined system of

$$m_{\tilde{\boldsymbol{f}}} + m_{\tilde{\boldsymbol{g}}} + 2m_{\tilde{\boldsymbol{s}}} = n_{\boldsymbol{x}} + n_{\boldsymbol{y}} + m_{\tilde{\boldsymbol{s}}}$$

equations in

$$2n_{\tilde{\boldsymbol{x}}} + n_{\tilde{\boldsymbol{y}}} + 2n_{\tilde{\boldsymbol{z}}} = 2n_{\boldsymbol{x}} + n_{\boldsymbol{y}} + n_{\tilde{\boldsymbol{z}}}$$

unknowns. In order to arrive at a square system the remaining

$$0 \overset{Eq.\ (3.44)}{\leq} n_{\mathrm{ddf}} = n_{\boldsymbol{x}} + n_{\tilde{\boldsymbol{z}}} - m_{\tilde{\boldsymbol{s}}} \overset{Eq.\ (3.43)}{\leq} n_{\boldsymbol{x}}$$

dynamic degrees of freedom have to be specified by additional functionally independent equations, i.e., problem specific initial or transition conditions $\boldsymbol{k}_{\mu}^{\mathrm{ini}}$, $\mu = 1, \ldots, n_{\mathrm{ddf}}$, have to be added as already discussed in Section 3.1.13.

Even if we restrict to continuity conditions in the differential states – which will be the basis for our method below – the number of

$$\binom{n_{\tilde{\boldsymbol{x}}}}{n_{\boldsymbol{x}} + n_{\tilde{\boldsymbol{z}}} - m_{\tilde{\boldsymbol{s}}}}$$

possible DOF-assignments may be very high already in the case of medium-scale systems, cf. Example 5 on page 114. Thus the main obstacle is in the combinatorial complexity of the problem. Of course, in general neither all possible assignments of the dynamic degrees of freedom are admissible in the sense of generating a regular system of reduced consistency equations, nor will all of the admissible sets be desirable from the application point of view. Therefore our strategy in the

sequel is to cut down the number of potential combinations as far as possible with limited numerical effort, and then make a unique choice in a final computationally more expensive step.

3.2.4.a Choice of the Type of Transition Conditions

According to the discussion in Section 3.1.13.a regarding the simulation of chemical engineering problems described by hybrid DAE models it is in this context reasonable to restrict to transition conditions[vii] enforcing continuity in an appropriately chosen subset of the differential variables of the model DAE. As the subject of our interest are semi-explicit DAEs originating from chemical engineering applications we stick to this approach.

A set of transition conditions of this type can be described by a set of pairwise different indices $K = \{K_1, \ldots, K_{n_{\mathrm{ddf}}}\} \subset \{1, \ldots, n_{\tilde{\boldsymbol{x}}}\}$ and equations

$$\boldsymbol{k}_\mu^{\mathrm{ini}}(\tilde{\boldsymbol{x}}^+_{K_\mu}(t_n), \boldsymbol{x}^-_{K_\mu}(t_n)) := \tilde{\boldsymbol{x}}^+_{K_\mu}(t_n) - \boldsymbol{x}^-_{K_\mu}(t_n) = 0;\ \mu = 1, \ldots, n_{\mathrm{ddf}}\,. \tag{3.56}$$

The conditions formulated in Eq. (3.56) are based on the assumption that each variable $\tilde{\boldsymbol{x}}^+_{K_\mu}$, $\mu = 1, \ldots, n_{\mathrm{ddf}}$, in the dynamical system after the discontinuity exactly corresponds to its counterpart $\boldsymbol{x}^-_{K_\mu}$ in the system before the discontinuity. However, in general a variable $\tilde{\boldsymbol{x}}^+_{K_\mu}$ may be used to express something different from what is described by $\boldsymbol{x}^-_{K_\mu}$. Consider, e.g., the case that some equations in the dynamical system are replaced at a discontinuity. Then the meaning of auxiliary variables in the system after the discontinuity may be entirely different from the meaning of their counterparts in the DAE before the discontinuity. Also, a differential variable $\tilde{\boldsymbol{x}}^+_{K_\mu}$ may correspond to an algebraic variable $\boldsymbol{y}^-_\nu$, $\nu \in \{1, \ldots, n_{\boldsymbol{y}}\}$, in the system before the switching event. Such a situation arises if, e.g., an algebraic condition $0 = \boldsymbol{y}^-_\nu - const_1$ is replaced by an ODE $\dot{\tilde{\boldsymbol{x}}}^+_{K_\mu} = const_2$, $\tilde{\boldsymbol{x}}^+_{K_\mu}(t_n) = \boldsymbol{y}^-_\nu(t_n)$ (an example is the usage of quasi steady-state conditions as initial conditions, cf. Remark 4.12 on page 162). We call the former case a change in the *physical meaning* of a variable, while we call the latter a change in the *mathematical type* of a variable. Under the assumption that the dimension of the state space is constant while the number of differential and algebraic states may vary, i.e.,

$$n_{\boldsymbol{x}^+} + n_{\boldsymbol{y}^+} = n_{\boldsymbol{x}^-} + n_{\boldsymbol{y}^-}\,,$$

there are five cases to be distinguished for each of the entries of the state variable vector $\boldsymbol{\xi}^+ := [[\boldsymbol{x}^+]^T, [\boldsymbol{y}^+]^T]^T$ in the system after the discontinuity:

[vii] In order to avoid the ambiguity in the term *initial conditions* (which is often used for both, the initial value of variables of a DAE and the equations required to define these values) we use the term *transition conditions* also for the specification of the initial conditions at t_0 in the sequel.

i. In the most simple case the entry keeps both its mathematical type and its physical meaning.

ii. An algebraic variable keeps its mathematical type but changes its physical meaning. By construction, this does not influence Eq. (3.56).

iii. A differential variable keeps its mathematical type but changes its physical meaning. Depending on the *specific model* a continuity condition Eq. (3.56) may or may not be appropriate.

iv. A differential variable becomes an algebraic variable. Then Eq. (3.56) does not apply.

v. An algebraic variable becomes a differential variable. In this case a continuity condition Eq. (3.56) is doubtful.

Items **i**, **ii**, and **iv** are clear, and item **iii** can be solved by deeper insight into the background of the model. Only case **v** is ambiguous. Therefore, as long as a differential variable of type **v** need not be assigned as a dynamic degree of freedom in order to obtain a uniquely solvable set of reduced consistency equations it should be excluded from the DOF-assignment.

In summary, transition conditions of the form

$$\boldsymbol{k}_{\mu}^{\mathrm{ini}}(\tilde{\boldsymbol{x}}_{K_\mu}^{+}(t_n), \boldsymbol{\xi}_{K^{\boldsymbol{\xi}}(\mu)}^{-}(t_n)) := \tilde{\boldsymbol{x}}_{K_\mu}^{+}(t_n) - \boldsymbol{\xi}_{K^{\boldsymbol{\xi}}(\mu)}^{-}(t_n) = 0;\ \mu = 1, \ldots, n_{\mathrm{ddf}}, \quad (3.57)$$

are required in order to handle the general case, where $\boldsymbol{\xi}^{-} := [[\boldsymbol{x}^{-}]^T, [\boldsymbol{y}^{-}]^T]^T$, and

$$K^{\boldsymbol{\xi}}(\mu) : \{1, \ldots, n_{\mathrm{ddf}}\} \rightarrow \left\{1, \ldots, n_{\boldsymbol{x}^{-}}, n_{\boldsymbol{x}^{-}} + 1, \ldots, n_{\boldsymbol{x}^{-}} + n_{\boldsymbol{y}^{-}}\right\}$$

is an appropriately defined injective mapping.

3.2.4.b Utilisation of Modelling Level Knowledge

In the previous discussion we have restricted the transition conditions to continuity in a subset of the differential state variables. In the next step we incorporate model specific background knowledge in order to obtain *a priori* statements for a reasonable choice of the transition conditions. This is desirable from both the application point of view as well as from the algorithmic point of view.

However, such information cannot be recovered from the DAE Eqs. (3.35a)–(3.35b) generated by the simulation environment as the numerical representation of a process. At this point we have to utilise detail knowledge on the modelling of the plant itself – or, more precisely – knowledge on the modelling of the single units.

Example 6:
Consider the one-dimensional model of a *heat exchanger* (see Figure 3.3), where a hot stream transfers energy to a second, cold stream. The streams are separated

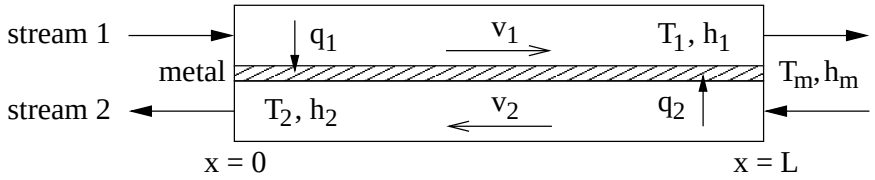

Figure 3.3: Heat exchanger (sketch) [ELBK 97].

through the heat exchanger material. Assuming that the streams exchange their heat with the exchanger material only, that the stream velocities v_1 and v_2 remain constant, and that the heat transport in axial direction is neglectable the heat exchanger can be modelled by the system of hyperbolic PDEs [ELBK 97] [(viii)]

$$\frac{1}{v_1}\frac{\partial h_1}{\partial t} = -\frac{\partial h_1}{\partial x} - \alpha_1\Omega_1(T_1 - T_\mathrm{m})\,, \tag{3.58}$$

$$\frac{1}{v_2}\frac{\partial h_2}{\partial t} = -\frac{\partial h_2}{\partial x} - \alpha_2\Omega_2(T_2 - T_\mathrm{m})\,, \tag{3.59}$$

$$\frac{\partial h_\mathrm{m}}{\partial t} = \alpha_1\Omega_1(T_1 - T_\mathrm{m}) + \alpha_2\Omega_2(T_2 - T_\mathrm{m})\,, \tag{3.60}$$

$$0 = h_\mathrm{stream}(T_1, P_1, \boldsymbol{z}_1) - h_1\,, \tag{3.61}$$

$$0 = h_\mathrm{stream}(T_2, P_2, \boldsymbol{z}_2) - h_2\,, \tag{3.62}$$

$$0 = h_\mathrm{metal}(T_\mathrm{m}) - h_\mathrm{m}\,, \tag{3.63}$$

where $h_{1,2}(x,t)$ are the enthalpy flowrates, $T_{1,2}(x,t)$ are the temperatures, $P_{1,2}(x,t)$ are the pressures, and (the vectors) $\boldsymbol{z}_{1,2}(x,t)$ are the compositions of the two streams. $h_\mathrm{m}(x,t)$ is the enthalpy per unit length, and $T_\mathrm{m}(x,t)$ is the temperature of the exchanger material. The heat transfer coefficients of the streams with the exchanger wall are given by $\alpha_\nu(T_\nu, P_\nu, \boldsymbol{z}_\nu)$, $\nu = 1, 2$, and $\Omega_{1,2}$ describes the respective heat transfer per unit length.

Leaving the numerical details to, e.g., [ELBK 97], the PDE system is approximated using the method of lines, i.e., the heat exchanger is discretised by a spatial grid. The result is a system of coupled ODEs and algebraic constraints, i.e., a DAE. In the context of this section it is of interest that the metal enthalpies at the grid points within the heat exchanger are differential variables. By (simple) physical reasoning it is clear that these enthalpies have to remain constant across discontinuities unless there is a possibility to transfer heat directly to the heat exchanger material. ◊

Example 6 demonstrates the knowledge required and the type of statements that can be made:

[(viii)] In order to keep consistency with the chemical engineering literature we *temporarily* overload symbols which may already be used in a different context.

1. The underlying models have to be known in detail.
2. Physical reasoning has to be applied.
3. The statements are restricted to differential states which are necessarily continuous across a discontinuity (i.e, statements of the form $\nu \in K$), independent from the type of discontinuity.

Items **1** and **2** imply that statements on the specification of the transition conditions require very basic knowledge. In application this information has to be provided in extension to the pure model DAE. These preparations require considerable effort. However, we want to emphasise that according to our experience otherwise a fully automated determination of a reasonable set of dynamic degrees of freedom is unacceptably time consuming in the case of high dimensional models. Additionally note that the assignment of the DOF has to be performed every time a consistent initialisation problem is to be solved, while the a priori classification of the possible dynamic degrees of freedom need to be done only once. Item **3** expresses that we do not see a means for the *exclusion* of a differential variable a priori from the set of dynamic degrees of freedom (i.e, statements of the type $\nu \notin K$).

For the sake of efficiency in our code the set of transition conditions specified by the considerations above is taken as inalterable. This means that these transition conditions have to be selected carefully in order to avoid a failure of the overall algorithm.

A basically similar, but complementary way of thought is reported by [BBR 01]. There, in the context of modelling languages for hybrid dynamical systems the additional *unknown declarator* for variables is proposed. This declarator gives the modeller the possibility to specify variables as *dependent* variables in the a general system of equations. In the context of the reduced system of consistency equations such variables are then excluded from the set of dynamic degrees of freedom. Similar to our method [BBR 01] restrict the transition conditions to continuity conditions in a subset of the differential variables. However, in contrast to our point of view (see item **3** above) [BBR 01] employ the unknown declarator in order to specify differential variables as dependent variables, explicitly *excluding* them from the set of dynamic degrees of freedom. The choice of the (appropriate) differential states is left to the modeller. Also we directly incorporate modelling level knowledge in order to bound the choice in the transition conditions. However, at the same time we consider in the higher index case at least some freedom in the choice of the DOF according to the actual problem as necessary. The reason is that in the context of our automatically generated higher index DAE the structure of the reduced derivative array equations and thus the final set of DOF is beyond control by the user.

3.2.4.c Analysis of the Structure of the Initialisation Problem

Now consider a fixed, but arbitrary selection of transition conditions Eq. (3.57) specified by a set K (and by a corresponding mapping $K^{\xi}(\mu)$). For brevity, we

define $\boldsymbol{h}(\tilde{\boldsymbol{x}}, \boldsymbol{\xi}^-) := [\boldsymbol{k}_1^{\mathrm{ini}}, \ldots, \boldsymbol{k}_{n_{\mathrm{ddf}}}^{\mathrm{ini}}]^T(\tilde{\boldsymbol{x}}, \boldsymbol{\xi}^-)$. Together with the reduced derivative array equations Eqs. (3.36a)–(3.36d) we obtain the reduced system of consistency equations

$$0 = \tilde{\boldsymbol{f}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}) \,, \tag{3.64a}$$

$$0 = \tilde{\boldsymbol{g}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}) \,, \tag{3.64b}$$

$$0 = \tilde{\boldsymbol{s}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \,, \tag{3.64c}$$

$$0 = \frac{\partial \tilde{\boldsymbol{s}}}{\partial t}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) \cdot \dot{\tilde{\boldsymbol{z}}} \,, \tag{3.64d}$$

$$0 = \boldsymbol{h}(\tilde{\boldsymbol{x}}, \boldsymbol{\xi}^-) \,, \tag{3.64e}$$

where the vector $\boldsymbol{\xi}^-$ is to be interpreted as a parameter. The Jacobian of Eqs. (3.64a)–(3.64e) with respect to $[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}]$ is

$$J(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}) := \begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{y}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{z}}}\right] & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \boldsymbol{0} \\ \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{y}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{z}}}\right] & \boldsymbol{0} & \boldsymbol{0} \\ \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}\right] & \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}\right] & \boldsymbol{0} & \boldsymbol{0} \\ \left[\frac{\partial}{\partial \tilde{\boldsymbol{x}}} \frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right] & \boldsymbol{0} & \left[\frac{\partial}{\partial \tilde{\boldsymbol{z}}} \frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}\right] \\ \left[\frac{\partial \boldsymbol{h}}{\partial \tilde{\boldsymbol{x}}}\right] & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}_{t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}} .$$

Using a transformation of block row and colum interchange the Jacobian J can be brought into block triangular form:

$$\det\left(J(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}})\right) = \det\left(\left[\begin{array}{ccc|cc} \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{y}}}\right] & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{z}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}}{\partial \tilde{\boldsymbol{x}}}\right] \\ \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{y}}}\right] & \boldsymbol{0} & \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{z}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}}{\partial \tilde{\boldsymbol{x}}}\right] \\ \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}\right] & \left[\frac{\partial}{\partial \tilde{\boldsymbol{z}}} \frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right] & \left[\frac{\partial}{\partial \tilde{\boldsymbol{x}}} \frac{\mathrm{d}\tilde{\boldsymbol{s}}}{\mathrm{d}t}\right] \\ \hline \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}\right] \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \left[\frac{\partial \boldsymbol{h}}{\partial \tilde{\boldsymbol{x}}}\right] \end{array}\right]_{t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}}\right) . \tag{3.65}$$

Because of the regularity of the upper left block of the transformed matrix (cf. Eq. (3.40)) the reduced system of consistency equations Eqs. (3.64a)–(3.64e) has a regular Jacobian J if and only if

$$\det\left(\begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{z}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}}{\partial \tilde{\boldsymbol{x}}}\right] \\ \boldsymbol{0} & \left[\frac{\partial \boldsymbol{h}}{\partial \tilde{\boldsymbol{x}}}\right] \end{bmatrix}_{t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \dot{\tilde{\boldsymbol{z}}}}\right) \neq 0 \,, \tag{3.66}$$

which is the same result as already encountered in Eq. (3.16) and Eq. (3.18) in Section 3.1.4. The dimension of this submatrix $m_{\tilde{\boldsymbol{s}}} + n_{\mathrm{ddf}} = n_{\boldsymbol{x}} + n_{\tilde{\boldsymbol{z}}}$ is by $n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$ smaller than the dimension $2n_{\boldsymbol{x}} + n_{\boldsymbol{y}} + n_{\tilde{\boldsymbol{z}}}$ of the Jacobian J. Thus we have arrived at a more concise criterion for the assignment of the dynamic degrees of freedom, especially in the case of a DAE Eqs. (3.35a)–(3.35b) with more algebraic than differential equations ($n_{\boldsymbol{y}} \gg n_{\boldsymbol{x}}$), and only few algebraic equations that have to be differentiated in order to arrive at an index-1 system ($n_{\boldsymbol{y}} \gg n_{\tilde{\boldsymbol{z}}}$).

Remark 3.23:
The previous argument is still valid if the transition conditions involve $\tilde{\boldsymbol{z}}$. In this case the entry $[\partial \boldsymbol{h}/\partial \tilde{\boldsymbol{z}}]$ is no longer a zero-matrix. However, in case of transition conditions in the algebraic variables $\tilde{\boldsymbol{y}}$ one has $[\partial \boldsymbol{h}/\partial \tilde{\boldsymbol{y}}] \not\equiv \boldsymbol{0}$, and the transformation of the Jacobian of the reduced system of consistency equations into block triangular form Eq. (3.65) is no longer possible. ◊

Eq. (3.66) shows several interesting properties of the DOF assignment problem in view:

1. As $[\partial \tilde{\boldsymbol{s}}/\partial \tilde{\boldsymbol{z}}]$ is of full column rank (which follows from Eq. (3.42)) the exclusion of $\tilde{\boldsymbol{z}}$ from the set of candidates for assignment as dynamic degrees of freedom is admissible.

2. Differential variables which cannot be assigned as dynamic degrees of freedom are fixed by those algebraic equations generating the higher index.

3. Differential variables which are not present in $\tilde{\boldsymbol{s}}$ *must* be assigned as dynamic degrees of freedom: Let K be an admissible DOF-assignment. Then

$$\forall \nu \in \{1, \ldots, n_{\tilde{\boldsymbol{x}}}\} : \{\mu \in \{1, \ldots, m_{\tilde{\boldsymbol{s}}}\} \mid \partial \tilde{\boldsymbol{s}}_\mu / \partial \tilde{\boldsymbol{x}}_\nu \neq 0\} = \emptyset \Rightarrow \nu \in K\,.$$

4. Let there be an assignment K of dynamic degrees of freedom. A first test for the admissibility of K can be performed by a *structural* check of the existence of a full maximum transversal ([Duff 81]: MC21 [AEA 93]) in Eq. (3.66). This is a numerically cheap method to filter sets K representing ineligible combinations of candidates for assignment as dynamic degrees of freedom.

3.2.4.d Algorithm

In summary the following steps are performed in order to determine a set of suitable candidates K for specification as dynamic degrees of freedom:

Algorithm 8 (Specification of Dynamic Degrees of Freedom)

1. *Based on considerations concerning the modelling of the units and their physical nature a subset of the differential variables* $\{\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_{n_{\tilde{\boldsymbol{x}}}}\}$ *is specified that* must *be continuous at the point of consistent initialisation* $t = t_{disc}$. *Let* $K^{unit} \subseteq K^{all} := \{1, \ldots, n_{\tilde{\boldsymbol{x}}}\}$ *contain the indices of these differential variables.*

This information is by necessity generated within the unit models.

2. *According to the analysis of Eq.* (3.66) *all differential variables that are not contained in* $\tilde{\boldsymbol{s}}$ *have to be specified as dynamic degrees of freedom, unless they are already considered in* K^{unit}. *The corresponding indices are collected in the set* K^{strct}, $K^{strct} \subseteq K^{all} \setminus K^{unit}$.

3. *If* $\mathrm{card}(K^{unit} \uplus K^{strct}) < n_{ddf}$ *the block lower triangularisation method discussed in Section 3.1.13 is applied to the system composed of the reduced derivative array equations together with the transition conditions enforcing continuity in the variables collected in* $K^{unit} \uplus K^{strct}$. *The probe is restricted to* $\tilde{\boldsymbol{x}}_\nu$, $\nu \in K^{all} \setminus \left\{K^{unit} \uplus K^{strct}\right\}$.

   In this way, differential variables are cancelled out *that have not been proposed for specification up to now, and specification of which will lead to a structurally singular system of reduced consistency equations. Let their indices be collected in* K^{sing}.

4. *From the above tests there is (in general) a remainder of indifferent differential variables* $\tilde{\boldsymbol{x}}_\nu$, $\nu \in K^{indiff} := K^{all} \setminus \left\{K^{unit} \uplus K^{strct} \uplus K^{sing}\right\}$ *that cannot be classified by problem specific knowledge and structural considerations.*

   Now the following possibilities exist:

   (a) *If* $\mathrm{card}(K^{unit} \uplus K^{strct}) > n_{ddf}$, *or if* $\mathrm{card}(K^{unit} \uplus K^{strct}) = n_{ddf}$ *and if the corresponding reduced system of consistency equations is* singular, *the algorithm terminates with an error as either too many variables or inappropriate variables have been specified as dynamic degrees of freedom.*

   (b) *If* $\mathrm{card}(K^{unit} \uplus K^{strct}) = n_{ddf}$ *and if the set of corresponding transition conditions generates* a regular *system of reduced consistency equations a suitable set of variables has been assigned as dynamic degrees of freedom, and the algorithm terminates.*

   (c) *Otherwise, the remaining* $n_{ddf} - \mathrm{card}(K^{unit} \uplus K^{strct})$ *dynamic degrees of freedom have to be selected from the set of indifferent candidates:*

      i. *If* $\mathrm{card}(K^{indiff}) < n_{ddf} - \mathrm{card}(K^{unit} \uplus K^{strct})$ *the algorithm terminates with an error as there is no choice for building a regular set of reduced consistency equations.*

      ii. *Otherwise a heuristic rule is used:*
      Consider all $\binom{\mathrm{card}(K^{indiff})}{n_{ddf} - \mathrm{card}(K^{unit} \uplus K^{strct})}$ *possible systems of reduced consistency equations that can be constructed by selecting the missing dynamic degrees of freedom from the set of indifferent candidates. Drop all systems that already* structurally *violate Eq.* (3.66) *(see the discussion there).*

From the remainder select that set of transition conditions which generates the Jacobian of the reduced consistency equations with the smallest condition number estimate at the initially given start values for the consistent initial values.
If even this criterion fails to deliver a unique result take that set of dynamic degrees of freedom which has been enumerated first.

Apart from step **4(c)ii** all steps in Algorithm 8 have either been already treated above, or their motivation is obvious. Step **4(c)ii** is discussed in detail within the subsequent Section 3.2.4.e.

Remark 3.24:
Problem dependent knowledge is applied in step **1** of Algorithm 8. In the overall algorithm there is no other step which enforces physically reasonable consistent initial conditions directly. ◊

3.2.4.e Condition Estimation

The numerically most difficult step in Algorithm 8 is the condition-based heuristic in step **4(c)ii**:

1. The number of possible sets of transition conditions that has to be examined may be very large: there are $\binom{\mathrm{card}(K^{\mathrm{indiff}})}{n_{\mathrm{ddf}}-\mathrm{card}(K^{\mathrm{unit}} \uplus K^{\mathrm{strct}})}$ such sets.

   Remark 3.25:
   For bookkeeping purposes we found it easier to look at the complementary combinations of $\mathrm{card}(K^{\mathrm{unit}} \uplus K^{\mathrm{strct}} \uplus K^{\mathrm{indiff}}) - n_{\mathrm{ddf}}$ out of $\mathrm{card}(K^{\mathrm{indiff}})$ indifferent candidates that can be *dropped*, as in our experiments $\mathrm{card}(K^{\mathrm{unit}} \uplus K^{\mathrm{strct}} \uplus K^{\mathrm{indiff}}) - n_{\mathrm{ddf}} \ll n_{\mathrm{ddf}} - \mathrm{card}(K^{\mathrm{unit}} \uplus K^{\mathrm{strct}})$ ◊

2. The estimation of the condition number for large, sparse, and unstructured matrices is computationally expensive.

3. The ranking of the Jacobians is to be done *before* the consistent initial conditions are available. Therefore the condition numbers may not be meaningful if the start estimate for the initial values is too far from the result.

We use the matrix condition as a means to introduce a half-ordering $(\mathcal{K}, \leq)$ on the set $\mathcal{K}$ of possible sets of candidates specifiable as dynamic degrees of freedom remaining after steps **1** to **3** in Algorithm 8. The choice of the smallest set in this ordering is motivated by the assumption that this set results in the system of reduced consistency equations which can be successfully solved with the highest probability. The weakness of our heuristic noted in item **3** has to be seen in relation to the fact that the final numerical solution of the consistent initialisation problem – which is a high dimensional, nonlinear root-finding problem – fails if the initial guess is too bad, anyway.

The numerical effort required for the enumerative check of all remaining possible DOF-configurations in step **4(c)ii** of our algorithm represents the main obstacle for its application. This problem is addressed in two ways:

1. A structural method (maximum transversal [Duff 81]: MC21 [AEA 93]) is used in order to filter structurally singular Jacobians, cf. the discussion of the structural properties of the initialisation problem in Section 3.2.4.c. An important point is that according to Eq. (3.66) this criterion can be restricted to a relatively *small* submatrix of the Jacobian of the consistency equations. Even more important, this structural method is very fast in comparison to numerical computations.

2. In case of a feasible set of dynamic degrees of freedom the condition number of the corresponding Jacobian J has to be computed. This computation has to be done highly efficiently. Here we utilise that only some columns of the Jacobian are affected by the differences in the possible assignments of dynamic degrees of freedom (see below).

As item **1** is clear we restrict our further discussion to item **2**. Recall that the condition number $\kappa(J)$ of a regular matrix J is defined as

$$\kappa(J) := \|J\| \left\|J^{-1}\right\| , \tag{3.67}$$

where $\|\cdot\|$ is a matrix norm. It is a measure for the maximum relative error to be expected in the solution of a linear system of the type $J\beta = \alpha$, given a disturbance in the system matrix J or in the right hand side α [Stoe 94].

Obviously, direct application of Eq. (3.67) is prohibitive. Therefore several techniques have been developed in order to obtain an iterative estimate for the condition number, or at least of its order. Two popular methods are the "LINPACK estimator" and "Hager's estimator" [Björ 96]. In both cases iterative schemes are employed in order to obtain estimates for the required matrix norms (especially of $\left\|J^{-1}\right\|$). The LINPACK estimator ([CMSW 79], improved by [OLea 80]) is equivalent to an inverse iteration scheme, while Hager's estimator for estimating $\|\cdot\|_1$ or $\|\cdot\|_\infty$ ([Hage 84], improved and implemented in [High 88], with a generalisation in [High 92], [HiTi 00]) is based on convex optimisation.

We have chosen Hager's estimator which requires the solution of systems of the type $J^{-1}\alpha = \beta$ and $J^{-T}\alpha = \beta$. An implementation of this condition estimator algorithm can be found in the HSL routine MC45 [AEA 93]. MC45 utilises MC41 [AEA 93] which implements Hager's estimator with Higham's improvements.

In our implementation we have replaced the routine for the solution of general sparse linear systems MA28 [AEA 93] – which is used in the version of MC45 available to us – by the state-of-the-art HSL routine MA48 [AEA 93]. Apart from an increased performance the new solver offers an additional feature allowing to specify *late columns* [DuRe 96]. If it is known that only a subset of the columns in a matrix changes between several calls of MA48 this subset can be declared as late columns. These late columns are then handled in a special way such that previously computed factorisation information can be used, which otherwise must be dropped and re-computed from scratch.

As in our problem setting the Jacobians J of the possible systems of consistency equations differ only in the columns representing the partial derivatives with respect to $\tilde{\boldsymbol{x}}_\nu$, $\nu \in K^{\text{indiff}}$, this feature can be applied advantageously.

Although we have successfully sped up an existing efficient code for the estimation of condition numbers in our setting we emphasise that the condition number based criterion in step **4(c)ii** of Algorithm 8 is intended as our "final line of defence". We consider the thorough analysis of the physically necessary transition conditions to be of utmost importance in order to provide as much data as possible for step **1** of Algorithm 8.

3.2.5 Solution of the Consistency Equations

The numerical solution of the reduced consistency equations Eqs. (3.64a)–(3.64e) completes our algorithm for the computation of consistent initial conditions. As in our problem setting the reduced consistency equations are large, sparse, and nonlinear efficient iterative multidimensional root finding methods have to be employed. Additionally, robustness is of special interest as our algorithm is implemented into the OPTISIM® simulation environment which is used in practical engineering work.

For a general review and discussion of methods for the solution of systems of nonlinear equations we refer to, e.g., [DeSc 96].

3.2.5.a Initial Guess for the Consistent Initial Conditions

In the case of the OPTISIM® environment there are three possible scenarios in which the calculation of consistent initial conditions may have to be performed:

1. When a completely new dynamic simulation is started from a quasi steady-state,
2. when a discontinuity during the integration of a DAE model has to be handled, or
3. when a dynamic simulation is restarted after it has been interrupted by the user.

In each of these cases the value of the state variable vector before consistent initialisation is part of a solution of a related simulation problem. Thus, on the one hand it can be assumed to be physically reasonable. On the other hand, in general only a part of the dynamical system is directly affected by the change between $t = t^-_{\text{disc}}$ and $t = t^+_{\text{disc}}$. Therefore the vectors $[\boldsymbol{x}^-, \boldsymbol{y}^-, \dot{\boldsymbol{x}}^-, \dot{\boldsymbol{y}}^-]$ in general provide a suitable initial guess.

In the context of chemical engineering applications the initial guess does not only have to be chosen appropriately due to purely numerical problems, e.g., limited convergence areas of iterative root finding methods, or convergence to undesired solutions, but also due to difficulties with the evaluation of the model DAE

and its Jacobian at intermediate iterates in the course of the iterative solution of the root-finding problem. If such an intermediate point is too far away from a physically reasonable point underlying routines for the calculation of physical properties may be forced to use extreme extrapolation giving even more unphysical results. In the worst case routines based on subordinate iterative processes may fail completely. Our setting thus provides us at least with a good chance for finding consistent initial conditions.

Remark 3.26:
In case of simultaneous equation-oriented simulation tools the provision of a sufficiently reasonable initial guess for a *steady-state solution* of an entire flowsheet is known as a major problem (cf. Section 2.2.2). Therefore it would be unreasonable to demand higher robustness – i.e., the ability to converge to the solution starting from scratch – from an algorithm designed for the treatment of the even more challenging consistent initialisation problem. ◊

3.2.5.b Steps Towards a Solution

As discussed above the start estimate for the consistent initial conditions in common possesses at least some physical meaning and can be assumed sufficiently close to the solution of the consistent initialisation problem. Additionally, we try to increase the robustness of the method by taking several steps towards the solution (some of these steps are optional):

1. In the first optional step a back-tracing algorithm as described in Section 3.1.10 may be employed. In our algorithm back-tracing is applied to the original index-2 DAE Eqs. (3.35a)–(3.35b) *before* the reduced consistency equations are formed.

   This back-tracing step can be regarded as a homotopy of a system related to Eqs. (3.64a)–(3.64e). However, during back-tracing neither the transition conditions Eq. (3.64e) specified in Section 3.2.4 are enforced, nor are the derived equations Eq. (3.64c). Additionally, the back-tracing method is not guaranteed to terminate successfully, see the numerical examples in Section 6.2.

2. In the second optional step numerical solution methods with special global convergence properties can be employed in order to get closer to the solution.

   (a) A Levenberg-Marquardt algorithm (NS13, [AEA 93]) may be used, or

   (b) the *feasibility phase* option of the SQP solver SNOPT [GMS 97a], [GMS 97b] may be employed. During the feasibility phase SNOPT minimises an l_1-criterion on the infeasibilities of the nonlinear constraints. In our case the nonlinear constraints are given by the residuals of the reduced consistency equations.

   The advantage of SNOPT is its ability to restart and continue automatically if at some point during the solution a residual or Jacobian

evaluation fails. Furthermore, it is able to return an accurate solution in case of convergence.

The disadvantage of these residual minimisation methods is that they are typically inefficient close to the solution. Thus we employ them only as pre-solvers in case of difficulties with the final solvers of step **3** below.

3. Finally, exact approximations to the consistent initial values are obtained

   (a) by a dog-leg method (NS02, [AEA 93]), and/or

   (b) by the affine-invariant Newton method NLEQ1S [NoWe 91].

According to our experience it is usually sufficient to use NLEQ1S only, which at the same time returns the most reliable results. The precision obtained by NS02 is inferior to the precision of the results of NLEQ1S. In our larger examples application of SNOPT as pre-solver in general has shown prohibitive due to high computational costs. Similarly, we can recommend NS13 for extremely difficult problems only. Results of our numerical tests are provided in Section 6.2.

Remark 3.27:
The Levenberg-Marquardt and the dog-leg solver routines from the HSL provide a reverse-communication interface. I.e., most of the major computations such as matrix-vector products, or the solution of linear systems are in the responsibility of the routine invoking the respective solver. This is advantageous especially in our case of large and sparse matrices, or in general for problems with special structure.

A brief discussion of the Levenberg-Marquardt and of the dog-leg method can be found in [DNR 87], [ChSt 81]. ◊

Remark 3.28:
For our purposes we have made two major modifications to NLEQ1S. In the first place, we apply the matrix equilibration strategy for the solution of linear systems described in Section 3.2.5.d below. Secondly, we have implemented an interface to the sophisticated sparse linear solver MA48 [AEA 93] which we use instead of the default solver MA28 [AEA 93]. ◊

3.2.5.c Scaling of the Nonlinear System

According to [AEA 93] both, variables and residual functions of the nonlinear system to be solved should be reasonably well scaled in the case of NS02. In [ChSt 81] the effects of scaling in the case of the dog-leg method is discussed in detail.

We apply linear scaling to the consistency equations and to the initial values in such a way that both the magnitude of the scaled variables and of the scaled residuals are kept within given bounds. As in the course of the solution both the residuals in the consistency equations and the iterates of the initial values may vary by orders in magnitude scaling is adapted if it appears to deteriorate. In such a case we restart NS13 and NS02 from the current iterate.

Scaling is fixed during a solution by SNOPT as restarts severely diminish its performance. A pathological case is rescaling during a line-search (cf. Section 2.4.2.b).

During the solution with NLEQ1S our scaling algorithm for the nonlinear system is turned off as this routine already uses tailored scaling mechanisms.

3.2.5.d Solution of Linear Systems

The computation of the Newton direction required by the dog-leg and by the affine-invariant Newton method is a basic problem. Especially, in our larger examples the system matrix given by the Jacobian of the reduced consistency equations turned out to be ill-conditioned. This is a matter of concern as the Jacobians are at least in parts computed by numerical approximations and thus are subject to perturbation. E.g., for the example of an air separation plant discussed in Section 6.2.3 the estimated condition is of $\mathcal{O}(1 \cdot 10^{12})$.

Remark 3.29:
In the beginning of our implementation the Jacobians of the reduced consistency equations turned out to be extremely ill-conditioned for several examples from application. E.g., for the example of an air separation plant discussed in Section 6.3.5 the estimated condition was of $\mathcal{O}(1 \cdot 10^{40})$. After some analysis this phenomenon could be assigned to the modelling of a specific unit. The – in itself correct – modelling approach used there leads to a hidden linear dependency in the Jacobian. Due to an accumulation of numerical errors this singularity appears as ill-conditioning. However, the most notable point is that this disadvantageous model by far does not affect standard steady-state and dynamic simulations as severely as the solution of the reduced consistency equations. ◊

In order to improve the performance of the linear solver we apply row and column equilibration using MC29 [AEA 93]. In our experiments a restriction on the size of the scaling factors showed to be advantageous if the restricted scaling was combined with an additional equilibration of the matrix rows. All scaling actions are performed using factors that are powers of two, thus avoiding round-off errors on digital computers using standard IEEE floating point arithmetics.

Chapter 4

Transfer of Sensitivity Functions at Discontinuities

Raffiniert ist der Herrgott,
aber boshaft ist er nicht.
(God may be subtle,
but He isn't plain mean.)

Albert Einstein
(FORTUNE cookie)

4.1 Review of Previous Work

4.1.1 Sensitivity Transfer in Systems of Ordinary Differential Equations with Discontinuities

Already [Roze 67] has derived concise sensitivity equations for discontinuous systems of ODEs of the form

$$\dot{\boldsymbol{\xi}}(t;\boldsymbol{p}) = \boldsymbol{F}_n(t,\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})\,; \qquad \begin{cases} t_{n-1} < t < t_n\,, \\ n = 1,2,\dots\,, \end{cases} \tag{4.1a}$$

$$\boldsymbol{\xi}(t_0;\boldsymbol{p}) = \boldsymbol{\xi}_0(t_0,\boldsymbol{p})\,, \tag{4.1b}$$

$$t_0 = t_0(\boldsymbol{p})\,, \tag{4.1c}$$

with dependent variables $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$, parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$, the independent variable $t \in \mathbb{R}$, and functions $\boldsymbol{F}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+n_\xi+n_p}, \mathbb{R}^{n_\xi}\right)$, $n = 1,2,\dots$. It is assumed that the switching instants t_n are determined as the isolated roots of the real-valued switching functions

$$\boldsymbol{q}_n(t_n,\boldsymbol{\xi}(t_n;\boldsymbol{p}),\boldsymbol{p}) = 0\,,$$

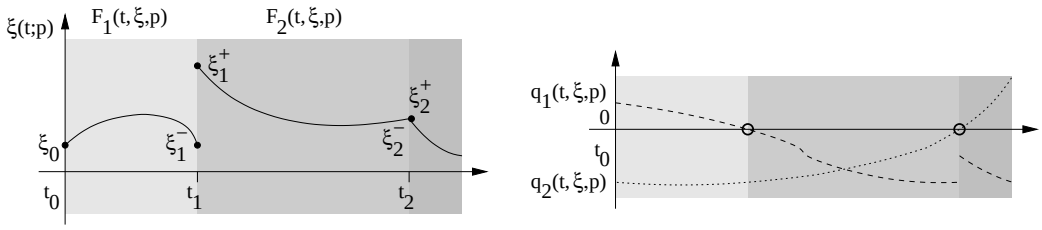

Figure 4.1: A discontinuous dynamical system $\dot{\boldsymbol{\xi}}(t;\boldsymbol{p}) = \boldsymbol{F}_n(t,\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})$, $n = 1,2,\ldots$, and its switching functions $\boldsymbol{q}_n(t,\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})$.

$\boldsymbol{q}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}},\mathbb{R}\right)$, $n = 1,2,\ldots$, and that the values of the dependent variables $\boldsymbol{\xi}_n^- := \lim_{t\nearrow t_n}\boldsymbol{\xi}(t;\boldsymbol{p})$, and $\boldsymbol{\xi}_n^+ := \lim_{t\searrow t_n}\boldsymbol{\xi}(t;\boldsymbol{p})$ at the discontinuities are related by jump functions

$$\boldsymbol{\xi}_n^+ = \boldsymbol{h}_n(t_n,\boldsymbol{\xi}_n^-,\boldsymbol{p})\,,$$

$\boldsymbol{h}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}},\mathbb{R}^{n_{\boldsymbol{\xi}}}\right)$, $n = 1,2,\ldots$. Figure 4.1 gives a sketch of such a system.

Within each interval $(t_{n-1};t_n)$, $n = 1,2,\ldots$, the sensitivities

$$\boldsymbol{\omega}(t,\boldsymbol{p}) := \left[\frac{\partial\boldsymbol{\xi}(t;\boldsymbol{p})}{\partial\boldsymbol{p}}\right] \in \mathbb{R}^{n_{\boldsymbol{\xi}}\times n_{\boldsymbol{p}}}$$

of the solution $\boldsymbol{\xi}(t;\boldsymbol{p})$ of Eq. (4.1a) satisfy the sensitivity equations

$$\dot{\boldsymbol{\omega}}(t,\boldsymbol{p}) = \left[\frac{\partial\boldsymbol{F}_n}{\partial\boldsymbol{\xi}}\right]_{\substack{t,\boldsymbol{p}\\ \boldsymbol{\xi}(t;\boldsymbol{p})}}\cdot\boldsymbol{\omega}(t,\boldsymbol{p}) + \left[\frac{\partial\boldsymbol{F}_n}{\partial\boldsymbol{p}}\right]_{\substack{t,\boldsymbol{p}\\ \boldsymbol{\xi}(t;\boldsymbol{p})}} \quad ; \quad t_{n-1} < t < t_n\,,$$

where the initial values of the sensitivities corresponding to Eqs. (4.1b)–(4.1c) are given by

$$\boldsymbol{\omega}(t_0,\boldsymbol{p}) = -\boldsymbol{F}_1(t_0,\boldsymbol{\xi}_0^+,\boldsymbol{p})\cdot\left[\frac{\partial t_0}{\partial\boldsymbol{p}}\right] + \left[\frac{\mathrm{d}\boldsymbol{\xi}_0}{\mathrm{d}\boldsymbol{p}}\right]\,. \tag{4.2}$$

Remark 4.1:
Please note the total derivative in Eq. (4.2). Resolving this derivative we obtain

$$\boldsymbol{\omega}(t_0,\boldsymbol{p}) = \left[-\boldsymbol{F}_1(t_0,\boldsymbol{\xi}_0^+,\boldsymbol{p}) + \frac{\partial\boldsymbol{\xi}_0}{\partial t_0}\right]\cdot\left[\frac{\partial t_0}{\partial\boldsymbol{p}}\right] + \left[\frac{\partial\boldsymbol{\xi}_0}{\partial\boldsymbol{p}}\right]\,.$$

◊

The problem in question are the *transition conditions* for the sensitivities. Using the abbreviations

$$\begin{aligned}
\Delta\boldsymbol{\omega}_n &= \boldsymbol{\omega}_n^+ - \boldsymbol{\omega}_n^- := \lim_{t\searrow t_n}\boldsymbol{\omega}(t,\boldsymbol{p}) - \lim_{t\nearrow t_n}\boldsymbol{\omega}(t,\boldsymbol{p})\,,\\
\Delta\boldsymbol{F}_n &= \boldsymbol{F}_n^+ - \boldsymbol{F}_n^- := \lim_{t\searrow t_n}\boldsymbol{F}_{n+1}(t,\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p}) - \lim_{t\nearrow t_n}\boldsymbol{F}_n(t,\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})\,,
\end{aligned}$$

$$\boldsymbol{h}_n^- := \lim_{t \nearrow t_n} \boldsymbol{h}_n(t, \boldsymbol{\xi}(t;\boldsymbol{p}), \boldsymbol{p}) \,,$$
$$\boldsymbol{q}_n^- := \lim_{t \nearrow t_n} \boldsymbol{q}_n(t, \boldsymbol{\xi}(t;\boldsymbol{p}), \boldsymbol{p}) \,,$$

the expressions derived by [Roze 67] can be written as

$$\Delta\boldsymbol{\omega}_n = \left\{ -\Delta\boldsymbol{F}_n + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} - \boldsymbol{Id}\right] \cdot \boldsymbol{F}_n^- + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial t}\right] \right\} \cdot \left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right] + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} - \boldsymbol{Id}\right] \cdot \boldsymbol{\omega}_n^- + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{p}}\right] , \quad (4.3)$$

where the sensitivities of the switching times are determined by

$$\left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right] = \left[\frac{\partial t_n}{\partial \boldsymbol{p}_1}, \ldots, \frac{\partial t_n}{\partial \boldsymbol{p}_{n_{\boldsymbol{p}}}}\right] = -\frac{\left[\frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{\xi}} \boldsymbol{\omega}_n^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{p}}\right]}{\frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{\xi}} \boldsymbol{F}_n^- + \frac{\partial \boldsymbol{q}_n^-}{\partial t}} . \quad (4.4)$$

Eq. (4.4) is valid as long as the denominator on the right hand side does not vanish. This condition is equivalent to the *transversality condition* $d\boldsymbol{q}_n^-/dt \neq 0$.

Later on the following reformulation of Eq. (4.3) is advantageous:

$$\begin{aligned} \boldsymbol{\omega}_n^+ - \boldsymbol{\omega}_n^- &= \left\{ -\left[\boldsymbol{F}_n^+ - \boldsymbol{F}_n^-\right] + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} - \boldsymbol{Id}\right] \cdot \boldsymbol{F}_n^- + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial t}\right] \right\} \cdot \left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right] \\ &\quad + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} - \boldsymbol{Id}\right] \cdot \boldsymbol{\omega}_n^- + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{p}}\right] \\ &\Leftrightarrow \\ \boldsymbol{\omega}_n^+ &= \left\{ -\boldsymbol{F}_n^+ + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial t} + \frac{\partial \boldsymbol{h}_n^-}{\partial t}\right] \right\} \cdot \left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right] + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{p}}\right] \\ &\overset{Eq.\ (4.4)}{=} \left\{ \boldsymbol{F}_n^+ - \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial t} + \frac{\partial \boldsymbol{h}_n^-}{\partial t}\right] \right\} \cdot \frac{\frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{\xi}} \boldsymbol{\omega}_n^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{p}}}{\frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{\xi}} \boldsymbol{F}_n^- + \frac{\partial \boldsymbol{q}_n^-}{\partial t}} + \left[\frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{h}_n^-}{\partial \boldsymbol{p}}\right] . \end{aligned} \quad (4.5)$$

4.1.2 Sensitivity Transfer of Differential States in Linear Implicit Index-1 DAEs

[SWS 95] require sensitivity information for discontinuous index-1 DAEs in order to perform parameter identification in mechanical multibody systems by a direct multiple shooting algorithm (cf. Section 2.4.1). Their investigations regarding sensitivities after discontinuities are based on [Kräm 85], [Bock 87] who consider the ODE case.

4.1.2.a Formulae for Sensitivities after Discontinuities

In general, modelling of multibody systems in redundant coordinates (according to the Euler-Lagrange formalism) results in linear implicit index-3 DAEs, cf., e.g., [GLG 85], [ABES 93]. [SWS 95] consider corresponding index-1 DAE systems which are obtained by replacing the algebraic constraints by their second total derivatives with respect to time (MBSSIM [ScWi 94]).

Remark 4.2:
In order to avoid drift effects during the numerical integration of the index-1 system the simulator MBSSIM uses *sequential projection* onto the first total derivative with respect to time of the algebraic constraints and onto the undifferentiated algebraic constraints [ScWi 94]. This is equivalent to the enforcement of the reduced system of derivative array equations. As an IND algorithm for the computation of the sensitivities is used this projection has to be considered for the sensitivities, too [SWS 95]. More detailed information can be found in [Schw 99]. ◊

Removing all of the special structure inherent in the class of problems treated, the index-1 DAE IVP can be stated in the notation of this treatise as

$$0 = \begin{cases} \boldsymbol{F}_1(t, \boldsymbol{x}(t;\boldsymbol{x}_0,\boldsymbol{p}), \boldsymbol{y}(t;\boldsymbol{x}_0,\boldsymbol{p}), \dot{\boldsymbol{x}}(t;\boldsymbol{x}_0,\boldsymbol{p}), \boldsymbol{p}); & t_0 \le t < t_{\text{disc}}\,, \\ \boldsymbol{F}_2(t, \boldsymbol{x}(t;\boldsymbol{x}_0,\boldsymbol{p}), \boldsymbol{y}(t;\boldsymbol{x}_0,\boldsymbol{p}), \dot{\boldsymbol{x}}(t;\boldsymbol{x}_0,\boldsymbol{p}), \boldsymbol{p}); & t_{\text{disc}} < t \le t_{\text{f}}\,, \end{cases} \tag{4.6}$$

$$0 = \boldsymbol{h}_0(\boldsymbol{x}(t_0), \boldsymbol{x}_0) = \boldsymbol{x}(t_0) - \boldsymbol{x}_0\,, \tag{4.7}$$

under the index-1 condition

$$\text{rank}\left(\left[\frac{\partial \boldsymbol{F}_n}{\partial \dot{\boldsymbol{x}}} \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{y}}\right]\right) = n_{\boldsymbol{x}} + n_{\boldsymbol{y}} \quad \text{in a neighbourhood of the solution}; \quad n = 1, 2\,,$$

with the independent variable $t \in [t_0, t_{\text{f}}]$ ($t_0, t_{\text{f}} \in \mathbb{R}$ fixed), differential and algebraic states $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$ and $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$, parameters $\boldsymbol{p} \in n_{\boldsymbol{p}}$, two sets of model equations $\boldsymbol{F}_1, \boldsymbol{F}_2 \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\boldsymbol{x}}+n_{\boldsymbol{y}}}\right)$, initial conditions $\boldsymbol{h}_0 \in \mathcal{C}^1\left(\mathbb{R}^{n_{\boldsymbol{x}}+n_{\boldsymbol{x}}}, \mathbb{R}^{n_{\boldsymbol{x}}}\right)$, and user given parameters $\boldsymbol{x}_0 \in \mathbb{R}^{n_{\boldsymbol{x}}}$ characterising the actual IVP to be solved. Due to their structure, MBSSIM can handle $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ as semi-explicit index-1 DAEs which eases the determination of consistent initial conditions for the derivatives of the differential variables $\dot{\boldsymbol{x}}$ as well as for the algebraic variables $\boldsymbol{y}$ [ScWi 94].

In order to simplify notation we only consider a single discontinuity in Eqs. (4.6)–(4.7). The time of this discontinuity $t_{\text{disc}} \in]t_0, t_{\text{f}}[$ is assumed to be indicated by the root of a real-valued switching function

$$0 = \boldsymbol{q}(t_{\text{disc}}, \boldsymbol{x}(t_{\text{disc}};\boldsymbol{p}), \boldsymbol{p})\,,$$

$\boldsymbol{q} \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{p}}}, \mathbb{R}\right)$. Note that $\boldsymbol{q}$ does not depend on the algebraic states $\boldsymbol{y}$. Further, the jump function $\hat{\boldsymbol{h}} \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\boldsymbol{x}}}\right)$ is defined as the increment

$$\boldsymbol{x}^+(t_0, \boldsymbol{x}_0, \boldsymbol{p}) = \boldsymbol{x}^-(t_0, \boldsymbol{x}_0, \boldsymbol{p}) + \hat{\boldsymbol{h}}(t_{\text{disc}}, \boldsymbol{x}^-(t_0, \boldsymbol{x}_0, \boldsymbol{p}), \boldsymbol{p})\,, \tag{4.8}$$

$\boldsymbol{x}^-(t_0, \boldsymbol{x}_0, \boldsymbol{p}) := \lim_{t \nearrow t_{\text{disc}}} \boldsymbol{x}(t; t_0, \boldsymbol{x}_0, \boldsymbol{p})$, $\boldsymbol{x}^+(t_0, \boldsymbol{x}_0, \boldsymbol{p}) := \lim_{t \searrow t_{\text{disc}}} \boldsymbol{x}(t; t_0, \boldsymbol{x}_0, \boldsymbol{p})$.

I.e., the differential variables $\boldsymbol{x}$ are selected as the $n_{\boldsymbol{x}}$ dynamic degrees of freedom. The initial values $\dot{\boldsymbol{x}}^+$ and $\boldsymbol{y}^+$ are determined consistently after the discontinuity.

The *Wronskians*[(i)] of the *differential* states $\boldsymbol{x}$ at time t with respect to the parameters $\boldsymbol{p}$ and with respect to the initial values (or *defining parameters*) $\boldsymbol{x}_*$, starting from t_*, are defined as

$$W_{\mathrm{ini}}(t,t_*) := \frac{\partial}{\partial \boldsymbol{x}_*}\boldsymbol{x}(t;t_*,\boldsymbol{x}_*,\boldsymbol{p}) \quad \text{and} \quad W_{\boldsymbol{p}}(t,t_*) := \frac{\partial}{\partial \boldsymbol{p}}\boldsymbol{x}(t;t_*,\boldsymbol{x}_*,\boldsymbol{p})\,, \tag{4.9}$$

$W_{\mathrm{ini}} \in \mathbb{R}^{n_{\boldsymbol{x}}\times n_{\boldsymbol{x}}}$, $W_{\boldsymbol{p}} \in \mathbb{R}^{n_{\boldsymbol{x}}\times n_{\boldsymbol{p}}}$. t_* denotes t_0 or t^+_{disc}. Accordingly, $\boldsymbol{x}_*$ is one of $\boldsymbol{x}_0$ or $\boldsymbol{x}^+$. The sensitivities of the algebraic states are not addressed. In their implementation [SWS 95] apply an *internal numerical differentiation* (IND) scheme based on finite differences (see, e.g., [Bock 87], [SBS 98], [Kieh 99]).

Remark 4.3:
[SWS 95] have to consider the sensitivities with respect to the initial values $\boldsymbol{x}_0$ as they use a direct multiple shooting approach for parameter identification. Further information on direct multiple shooting for parameter identification can be found in, e.g., [Bock 87], [Schl 88] who consider the ODE case, and [Heim 92], [HeSt 96] where index-1 DAEs arising from mechanical multibody systems without state dependent discontinuities are treated. ◊

The formulae given for the Wronskians in the interval after the discontinuity $t \in [t^+_{\mathrm{disc}}, t_{\mathrm{f}}]$ are

$$W_{\mathrm{ini}}(t,t_0) = W_{\mathrm{ini}}(t,t^+_{\mathrm{disc}})U_{\mathrm{ini}}W_{\mathrm{ini}}(t^-_{\mathrm{disc}},t_0)\,, \tag{4.10a}$$

$$W_{\boldsymbol{p}}(t,t_0) = W_{\mathrm{ini}}(t,t^+_{\mathrm{disc}})\left(U_{\mathrm{ini}}W_{\boldsymbol{p}}(t^-_{\mathrm{disc}},t_0) + U_{\boldsymbol{p}}\right) + W_{\boldsymbol{p}}(t,t^+_{\mathrm{disc}})\,, \tag{4.10b}$$

where the update matrices $U_{\mathrm{ini}} \in \mathbb{R}^{n_{\boldsymbol{x}}\times n_{\boldsymbol{x}}}$ and $U_{\boldsymbol{p}} \in \mathbb{R}^{n_{\boldsymbol{x}}\times n_{\boldsymbol{p}}}$ are defined by

$$U_{\mathrm{ini}} := \left[\dot{\boldsymbol{x}}^+ - \dot{\boldsymbol{x}}^- - \frac{\partial \hat{\boldsymbol{h}}}{\partial t} - \frac{\partial \hat{\boldsymbol{h}}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^-\right]\frac{\left[\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}\right]}{\dot{\boldsymbol{q}}} + \boldsymbol{Id} + \left[\frac{\partial \hat{\boldsymbol{h}}}{\partial \boldsymbol{x}}\right]\,, \tag{4.11a}$$

$$U_{\boldsymbol{p}} := \left[\dot{\boldsymbol{x}}^+ - \dot{\boldsymbol{x}}^- - \frac{\partial \hat{\boldsymbol{h}}}{\partial t} - \frac{\partial \hat{\boldsymbol{h}}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^-\right]\frac{\left[\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}}\right]}{\dot{\boldsymbol{q}}} + \left[\frac{\partial \hat{\boldsymbol{h}}}{\partial \boldsymbol{p}}\right]\,, \tag{4.11b}$$

using $\dot{\boldsymbol{q}} := \mathrm{d}\boldsymbol{q}(t,\boldsymbol{x}(t,\boldsymbol{p}),\boldsymbol{p})/\mathrm{d}t$.

4.1.2.b Equivalence with the Results Derived by Rozenvasser

In accordance with the notation in Section 4.1.1 we define a new jump function

$$\boldsymbol{h}(t,\boldsymbol{x},\boldsymbol{p}) := \hat{\boldsymbol{h}}(t,\boldsymbol{x},\boldsymbol{p}) + \boldsymbol{x}\,,$$

[(i)] In this section we stick to the nomenclature used in [SWS 95], although in our case a *Wronskian* does not differ from a sensitivity matrix.

$\boldsymbol{h} \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\boldsymbol{x}}}\right)$, which (trivially) owns the partial derivatives

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \frac{\partial \hat{\boldsymbol{h}}}{\partial \boldsymbol{x}} + \boldsymbol{Id}\,, \quad \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{p}} = \frac{\partial \hat{\boldsymbol{h}}}{\partial \boldsymbol{p}}\,, \text{ and } \quad \frac{\partial \boldsymbol{h}}{\partial t} = \frac{\partial \hat{\boldsymbol{h}}}{\partial t}\,.$$

Therefore the leading terms in Eqs. (4.11a)–(4.11b) transform as

$$\left[\dot{\boldsymbol{x}}^+ - \dot{\boldsymbol{x}}^- - \frac{\partial \hat{\boldsymbol{h}}}{\partial t} - \frac{\partial \hat{\boldsymbol{h}}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^-\right] = \left\{\frac{\partial \boldsymbol{x}^+}{\partial t} - \left[\frac{\partial \boldsymbol{h}}{\partial t} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial t}\right]\right\}.$$

By Eq. (4.9) and by $\boldsymbol{x}(t^+_{\text{disc}}) = \boldsymbol{x}(t^+_{\text{disc}}; t^+_{\text{disc}}, \boldsymbol{x}^+, \boldsymbol{p}) = \boldsymbol{x}^+$ (where $\boldsymbol{x}^+$ is to be interpreted as the *fixed* initial value at the discontinuity defined in Eq. (4.8)) the Wronskians satisfy

$$\begin{aligned}
W_{\text{ini}}(t^-_{\text{disc}}, t_0) &= \frac{\partial \boldsymbol{x}(t^-_{\text{disc}}; t_0, \boldsymbol{x}_0, \boldsymbol{p})}{\partial \boldsymbol{x}_0} = \frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{x}_0}\,, \\
W_{\boldsymbol{p}}(t^-_{\text{disc}}, t_0) &= \frac{\partial \boldsymbol{x}(t^-_{\text{disc}}; t_0, \boldsymbol{x}_0, \boldsymbol{p})}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{p}}\,, \\
W_{\text{ini}}(t^+_{\text{disc}}, t^+_{\text{disc}}) &= \frac{\partial}{\partial \boldsymbol{x}^+}\boldsymbol{x}(t^+_{\text{disc}}; t^+_{\text{disc}}, \boldsymbol{x}^+, \boldsymbol{p}) = \boldsymbol{Id}\,, \text{ and} \\
W_{\boldsymbol{p}}(t^+_{\text{disc}}, t^+_{\text{disc}}) &= \frac{\partial}{\partial \boldsymbol{p}}\boldsymbol{x}(t^+_{\text{disc}}; t^+_{\text{disc}}, \boldsymbol{x}^+, \boldsymbol{p}) = \boldsymbol{0}\,.
\end{aligned}$$

Thus Eqs. (4.10a)–(4.10b) evaluated at t^+_{disc} for the Wronskians $W_{\text{ini}}(t^+_{\text{disc}}, t_0)$ and $W_{\boldsymbol{p}}(t^+_{\text{disc}}, t_0)$ of the discontinuous IVP Eqs. (4.6)–(4.7) read as

$$\begin{aligned}
W_{\text{ini}}(t^+_{\text{disc}}, t_0) &= W_{\text{ini}}(t^+_{\text{disc}}, t^+_{\text{disc}}) U_{\text{ini}} W_{\text{ini}}(t^-_{\text{disc}}, t_0) \\
&= \left\{\frac{\partial \boldsymbol{x}^+}{\partial t} - \left[\frac{\partial \boldsymbol{h}}{\partial t} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial t}\right]\right\} \frac{\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{x}_0}}{\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial t} + \frac{\partial \boldsymbol{q}}{\partial t}} + \left[\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{x}_0}\right],
\end{aligned}$$

and

$$\begin{aligned}
W_{\boldsymbol{p}}(t^+_{\text{disc}}, t_0) &= W_{\text{ini}}(t^+_{\text{disc}}, t^+_{\text{disc}}) \left(U_{\text{ini}} W_{\boldsymbol{p}}(t^-_{\text{disc}}, t_0) + U_{\boldsymbol{p}}\right) + W_{\boldsymbol{p}}(t^+_{\text{disc}}, t^+_{\text{disc}}) \\
&= \left\{\frac{\partial \boldsymbol{x}^+}{\partial t} - \left[\frac{\partial \boldsymbol{h}}{\partial t} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial t}\right]\right\} \frac{\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}}}{\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial t} + \frac{\partial \boldsymbol{q}}{\partial t}} + \left[\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{p}}\right].
\end{aligned}$$

The expression for $W_{\boldsymbol{p}}(t^+_{\text{disc}}, t_0)$ is equivalent to Eq. (4.5) given in Section 4.1.1. This means that in case of discontinuous index-1 DAEs the sensitivities for the differential states transform as in the ODE case, given that the differential states are chosen as the dynamic degrees of freedom and that the transition conditions for the DAE as well as the switching functions do not depend on the algebraic states.

Remark 4.4:
The difference in the expressions obtained for $W_{\mathrm{ini}}(t^+_{\mathrm{disc}}, t_0)$ and $W_{\boldsymbol{p}}(t^+_{\mathrm{disc}}, t_0)$ by the reformulation above mirrors that the defining parameters $\boldsymbol{x}_0$ do neither enter the switching function nor the jump function directly, in contrast to the model parameters $\boldsymbol{p}$. ◊

Remark 4.5:
Also [GFB 99] report the equivalence of the results of [SWS 95] with the results of [Roze 67]. However, [GFB 99] do not provide an explicit derivation of this equivalence. ◊

4.1.3 Sensitivity Transfer in DAEs: Numerical Differentiation of the Jump-Function

Similar to [SWS 95] discussed in Section 4.1.2 above [ErAr 98] aim at the calculation of sensitivities for discontinuous DAEs arising from dynamic mechanical multibody system simulation. For the continuous part of the task they derive sensitivity equations for index-3 DAEs of mechanical multibody systems with holonomic constraints. In order to treat the discrete part of the task they resort to the simplified problem of sensitivities for discontinuous systems of ODEs. The argument is that for DAEs describing mechanical multibody systems it is always possible to find an ODE representation at least *locally* (see Definition 1.9 (differential index), Definition 1.10 (corresponding extended system of a DAE)). Under this assumption the following lemma is shown by [ErAr 98]:

Lemma 4.1 (Sensitivities of a Discontinuous Solution)
Let there be a nonempty interval $[t_0, t_f] \subset \mathbb{R}$, *a vector of parameters* $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$, *and a switching function* $\boldsymbol{q} : \mathbb{R}^{n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}} \to \mathbb{R}$. *Suppose that there exists a unique* $t_{disc} = t_{disc}(\boldsymbol{p})$, $t_0 < t_{disc} < t_f$, *such that*

$$\begin{aligned} \boldsymbol{q}(\boldsymbol{\xi}(t_0;\boldsymbol{p}),\boldsymbol{p}) &< 0\,, \\ \lim_{t \nearrow t_{disc}} \boldsymbol{q}(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p}) &= 0\,, \end{aligned}$$

where $\boldsymbol{\xi}(t;\boldsymbol{p}) : [t_0, t_f] \times \mathbb{R}^{n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$ *is the solution of the system of IVPs*

$$\begin{aligned} \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}) &= \boldsymbol{F}_1(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})\,; \quad t \in [t_0, t_{disc}[\,, \\ \boldsymbol{\xi}(t_0;\boldsymbol{p}) &= \boldsymbol{\xi}_0(\boldsymbol{p})\,, \quad \textit{and} \\ \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}) &= \boldsymbol{F}_2(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})\,; \quad t \in [t_{disc}, t_f]\,, \\ \boldsymbol{\xi}(t_{disc};\boldsymbol{p}) &= \boldsymbol{h}(\boldsymbol{\xi}^-(\boldsymbol{p}),\boldsymbol{p})\,, \end{aligned}$$

with

$$\boldsymbol{q}(\boldsymbol{h}(\boldsymbol{\xi}^-(\boldsymbol{p}),\boldsymbol{p}),\boldsymbol{p}) \geq 0\,,$$

where $\boldsymbol{F}_1, \boldsymbol{F}_2 : \mathbb{R}^{n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$, $\boldsymbol{\xi}^-(\boldsymbol{p}) := \lim_{t \nearrow t_{disc}} \boldsymbol{\xi}(t;\boldsymbol{p})$, *and* $\boldsymbol{h} : \mathbb{R}^{n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{\xi}}}$ *is the jump function.*

Assume that $\boldsymbol{F}_1(\boldsymbol{\xi},\boldsymbol{p})$, $\boldsymbol{F}_2(\boldsymbol{\xi},\boldsymbol{p})$, $\boldsymbol{q}(\boldsymbol{\xi},\boldsymbol{p})$, *and* $\boldsymbol{h}(\boldsymbol{\xi},\boldsymbol{p})$ *are continuously differentiable with respect to* $\boldsymbol{\xi}$ *and* $\boldsymbol{p}$, *and that* $\boldsymbol{\xi}_0(\boldsymbol{p})$ *is continuously differentiable with*

respect to $\boldsymbol{p}$. *Further assume that for all* $t \in [t_0, t_{disc}]$ *the following relations hold:*

$$\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p}) \cdot \boldsymbol{F}_1(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p}) > 0\,,$$
$$\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}^-(\boldsymbol{p}),\boldsymbol{p}) \cdot \boldsymbol{F}_1(\boldsymbol{\xi}^-(\boldsymbol{p}),\boldsymbol{p}) > 0\,,$$

and in case of $\boldsymbol{q}(\boldsymbol{h}(\boldsymbol{\xi}^-(\boldsymbol{p}),\boldsymbol{p}),\boldsymbol{p}) = 0$

$$\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}(t_{disc};\boldsymbol{p}),\boldsymbol{p}) \cdot \boldsymbol{F}_1(\boldsymbol{\xi}(t_{disc};\boldsymbol{p}),\boldsymbol{p}) > 0\,.$$

Then for all $t \in]t_0, t_{disc}[\cup]t_{disc}, t_f[$ *the sensitivities* $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t,\boldsymbol{p})$ *exist. Further, for all* $t \in]t_0, t_{disc}[$ *the sensitivities are given by*

$$\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t,\boldsymbol{p}) = \frac{\partial \boldsymbol{\xi}_0}{\partial \boldsymbol{p}}(\boldsymbol{p}) + \int_{t_0}^{t} \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{p}}(\boldsymbol{\xi}(\tau;\boldsymbol{p}),\boldsymbol{p})d\tau\,,$$

with the limit from below

$$\frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}}(\boldsymbol{p}) := \lim_{t \nearrow t_{disc}} \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t,\boldsymbol{p}) = \frac{\partial \boldsymbol{\xi}_0}{\partial \boldsymbol{p}}(\boldsymbol{p}) + \int_{t_0}^{t_{disc}} \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{p}}(\boldsymbol{\xi}(\tau;\boldsymbol{p}),\boldsymbol{p})d\tau\,.$$

Furthermore, with $\boldsymbol{F}^- := \lim_{t \nearrow t_{disc}} \boldsymbol{F}_1(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})$, $\boldsymbol{F}^+ := \lim_{t \searrow t_{disc}} \boldsymbol{F}_2(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})$, $\boldsymbol{q}^- := \lim_{t \nearrow t_{disc}} \boldsymbol{q}(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})$, *and* $\boldsymbol{h}^- := \lim_{t \nearrow t_{disc}} \boldsymbol{h}(\boldsymbol{\xi}(t;\boldsymbol{p}),\boldsymbol{p})$ *the sensitivities for all* $t \in]t_{disc}, t_f[$ *are given by*

$$\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(\boldsymbol{p},t) = \left[\frac{\boldsymbol{F}^+ - \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-}{\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-} \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}}\right] \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \left[\frac{\boldsymbol{F}^+ - \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-}{\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-} \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{p}}\right] + \int_{t_{disc}}^{t} \frac{\partial \boldsymbol{F}_2}{\partial \boldsymbol{p}}(\boldsymbol{\xi}(\tau;\boldsymbol{p}),\boldsymbol{p})d\tau\,. \quad (4.15)$$

4.1.3.a Algorithm for the Calculation of Sensitivities after a Discontinuity

For $t^+_{\text{disc}} := t \searrow t_{\text{disc}}$ Eq. (4.15) gives the value of the sensitivities shortly after the discontinuity as

$$\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(t^+_{\text{disc}},\boldsymbol{p}) = \frac{\boldsymbol{F}^+ - \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-}{\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-} \left[\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{p}}\right] + \left[\frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{p}}\right]. \quad (4.16)$$

In the application considered by [ErAr 98] the right hand side of the model equations at the discontinuity, i.e., $\boldsymbol{F}^+$, is directly accessible, but derivative information for both the switching function $\boldsymbol{q}$ and for the jump function $\boldsymbol{h}$ is missing. Therefore [ErAr 98] propose to approximate the unknown derivatives of the switching function and of the jump function present in Eq. (4.16) by one-sided first order finite differences. From the integrator the time of discontinuity t_{disc}, the numerical, left-sided approximation $\boldsymbol{\xi}^-$ of the states at t_{disc}, as well as their derivatives $\dot{\boldsymbol{\xi}}^-(=\boldsymbol{F}^-)$ and their sensitivities $\boldsymbol{\omega}^-$, and the actual set of parameters $\boldsymbol{p}$ have to be passed to the algorithm. Additionally, the norms $\|\boldsymbol{\xi}^-\|$, $\|\boldsymbol{\omega}^-\|$, and $\|\boldsymbol{p}\|$ are needed. Other input parameters of the algorithm are the relative tolerance ϵ_{rel}, the absolute tolerance ϵ_{abs}, and the machine precision ϵ_{mach}. Altogether, the algorithm for the sensitivity transfer after discontinuities developed by [ErAr 98] reads as:

Algorithm 9 (Sensitivities after Discontinuities)

1. *Set disturbance for finite difference approximation of time derivatives*

$$\epsilon_{\boldsymbol{\xi}} := \sqrt{\epsilon_{mach}} \cdot \max(\|\boldsymbol{\xi}^-\|, \epsilon_{rel} \cdot \|\boldsymbol{\xi}^-\| + \epsilon_{abs}) \,.$$

2. *Approximate total derivatives with respect to t*

$$\frac{d}{dt}\boldsymbol{q}^- = \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^- \doteq \frac{\boldsymbol{q}(\boldsymbol{\xi}^- + \epsilon_{\boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^-, \boldsymbol{p}) - \boldsymbol{q}(\boldsymbol{\xi}^-, \boldsymbol{p})}{\epsilon_{\boldsymbol{\xi}}} \,,$$
$$\frac{d}{dt}\boldsymbol{h}^- = \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^- \doteq \frac{\boldsymbol{h}(\boldsymbol{\xi}^- + \epsilon_{\boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^-, \boldsymbol{p}) - \boldsymbol{h}(\boldsymbol{\xi}^-, \boldsymbol{p})}{\epsilon_{\boldsymbol{\xi}}} \,.$$

3. *Introduce auxiliary variable $\boldsymbol{v}$*

$$\boldsymbol{v} := \frac{\boldsymbol{F}^+ - \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^-}{\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^-} \,.$$

4. *Fix disturbance for finite difference approximation of parametric derivatives*

$$\epsilon_{\boldsymbol{p}} := \sqrt{\epsilon_{mach}} \cdot \max\left(\|\boldsymbol{p}\|, \|\boldsymbol{\omega}^-\|, \epsilon_{rel} \cdot \|\boldsymbol{\omega}^-\| + \epsilon_{abs}\right) \,.$$

5. *Approximate the total derivatives with respect to $\boldsymbol{p}_\nu$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$*

$$\frac{d}{d\boldsymbol{p}_\nu}\boldsymbol{q}^- = \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}_\nu} + \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{p}_\nu} \doteq \frac{\boldsymbol{q}\left(\boldsymbol{\xi}^- + \epsilon_{\boldsymbol{p}} \cdot [\boldsymbol{\omega}^-]_\nu, \boldsymbol{p} + \epsilon_{\boldsymbol{p}} \boldsymbol{e}_\nu\right) - \boldsymbol{q}\left(\boldsymbol{\xi}^-, \boldsymbol{p}\right)}{\epsilon_{\boldsymbol{p}}} \,,$$
$$\frac{d}{d\boldsymbol{p}_\nu}\boldsymbol{h}^- = \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}_\nu} + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{p}_\nu} \doteq \frac{\boldsymbol{h}\left(\boldsymbol{\xi}^- + \epsilon_{\boldsymbol{p}} \cdot [\boldsymbol{\omega}^-]_\nu, \boldsymbol{p} + \epsilon_{\boldsymbol{p}} \boldsymbol{e}_\nu\right) - \boldsymbol{h}\left(\boldsymbol{\xi}^-, \boldsymbol{p}\right)}{\epsilon_{\boldsymbol{p}}} \,,$$

where $\boldsymbol{e}_\nu$ *is the* ν^{th} *unit vector in* $\mathbb{R}^{n_\xi}$ *and* $[\boldsymbol{\omega}]_\nu$ *is the* ν^{th} *column of the sensitivity matrix* $\boldsymbol{\omega} \in \mathbb{R}^{n_\xi \times n_p}$.

6. *Calculate new sensitivities*

$$[\boldsymbol{\omega}^+]_\nu \doteq \boldsymbol{v} \cdot \left[\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}_\nu} + \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{p}_\nu}\right] + \left[\frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}_\nu} + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{p}_\nu}\right] .$$

The point to be noted in Algorithm 9 is that by utilising directional derivatives in steps **2** and **5** costly approximation of the Jacobians of the switching function and of the jump function can be avoided.

4.1.3.b Equivalence with the Results Derived by Rozenvasser

Slightly reordering Eq. (4.16) we find

$$\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}(\boldsymbol{p}, t^+_{\text{disc}}) = \left[\boldsymbol{F}^+ - \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-\right] \frac{\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{p}}}{\frac{\partial \boldsymbol{q}^-}{\partial \boldsymbol{\xi}} \cdot \boldsymbol{F}^-} + \left[\frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{p}}\right] . \quad (4.17)$$

The difference between Eq. (4.17) and Eq. (4.5) in Section 4.1.1 is due to [Roze 67] taking into account explicit time dependence in the switching function $\boldsymbol{q}$ and in the jump function $\boldsymbol{h}$, which is not the case with [ErAr 98]. As [ErAr 98] restrict to the underlying ODE of the original DAE system this equivalence has to be expected. In a strict sense due to the restriction to the ODE case the above results apply to the UODE only. However, as shown in [SWS 95] under some assumptions on the jump and switching functions these formulae also apply directly to the differential states of index-1 DAEs obtained by index reduction of mechanical multibody DAE models, cf. Section 4.1.2 above.

4.1.4 Sensitivity Transfer in Systems of Index-1 Differential-Algebraic Equations with Discontinuities

In [Feeh 98] and in the related papers [GaBa 98], [GFB 99] optimal control of batch processes (which are a common type of process in chemical engineering) and related problems of sensitivity transfer at discontinuities are addressed. In this section we restrict to a summary of their results regarding the computation of sensitivity functions for discontinuous DAE models.

The process model is given by the DAEs [(ii)]

$$\boldsymbol{F}_n(t, \boldsymbol{x}(t;\boldsymbol{p}), \boldsymbol{y}(t;\boldsymbol{p}), \dot{\boldsymbol{x}}(t;\boldsymbol{p}), \boldsymbol{p}) = 0\,; \quad t_{n-1} < t < t_n,\ n = 1, 2, \dots , \quad (4.18)$$

[(ii)] The notation used in [Feeh 98], [GaBa 98], [GFB 99] has been adapted and simplified. Especially, control variables $\boldsymbol{u}$ have been suppressed and the dimension of the variables has been fixed for all times. Furthermore, the sequence of transitions is assumed to be fixed.

where $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$ is the vector of differential variables, $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$ is the vector of algebraic variables, $\boldsymbol{p} \in n_{\boldsymbol{p}}$ is the vector of parameters, $t \in \mathbb{R}$ is the independent variable, and $\boldsymbol{F}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\dot{\boldsymbol{x}}}+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\dot{\boldsymbol{x}}}+n_{\boldsymbol{y}}}\right)$, $n = 1, 2, \ldots$ are the consecutive models. By the rank condition

$$\operatorname{rank}\left(\left[\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{F}_n}{\partial \dot{\boldsymbol{x}}}\right]\right) = n_{\dot{\boldsymbol{x}}} + n_{\boldsymbol{y}} \quad \text{(along the solution)}\,, \tag{4.19}$$

the range of problems covered is restricted to ODEs and most index-1 DAEs. Higher index DAEs are treated by index reduction according to the method of dummy derivatives (see Section 3.1.6). Further we have the a priori adequately defined sets of initial conditions and transition conditions

$$\begin{aligned} \boldsymbol{h}_0(t_0, \boldsymbol{x}_0, \boldsymbol{y}_0, \dot{\boldsymbol{x}}_0, \boldsymbol{p}) &= 0\,, \\ \boldsymbol{h}_n(t_n, \boldsymbol{x}_n^+, \boldsymbol{y}_n^+, \dot{\boldsymbol{x}}_n^+, \boldsymbol{x}_n^-, \boldsymbol{y}_n^-, \dot{\boldsymbol{x}}_n^-, \boldsymbol{p}) &= 0\,; \quad n = 1, 2, \ldots\,, \end{aligned}$$

$\boldsymbol{h}_0 \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\dot{\boldsymbol{x}}}+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\dot{\boldsymbol{x}}}}\right)$, $\boldsymbol{h}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+2(n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\dot{\boldsymbol{x}}})+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\dot{\boldsymbol{x}}}}\right)$, where with $\boldsymbol{z} \in \{\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}\}$, $\boldsymbol{z}_0 = \boldsymbol{z}_0(t_0; \boldsymbol{p}) := \boldsymbol{z}(t; \boldsymbol{p})|_{t=t_0}$, $\boldsymbol{z}_n^- := \boldsymbol{z}_n^-(t_n; \boldsymbol{p}) := \lim_{t \nearrow t_n} \boldsymbol{z}(t; \boldsymbol{p})$, and $\boldsymbol{z}_n^+ := \boldsymbol{z}_n^+(t_n; \boldsymbol{p}) := \lim_{t \searrow t_n} \boldsymbol{z}(t; \boldsymbol{p})$, $n = 1, 2, \ldots$.

The times of the discontinuity events are given as the isolated zeros of the real-valued switching functions

$$\boldsymbol{q}_n(t_n, \boldsymbol{x}(t_n; \boldsymbol{p}), \boldsymbol{y}(t_n; \boldsymbol{p}), \dot{\boldsymbol{x}}(t_n; \boldsymbol{p}), \boldsymbol{p}) = 0\,,$$

$\boldsymbol{q}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\dot{\boldsymbol{x}}}+n_{\boldsymbol{p}}}, \mathbb{R}\right)$, $n = 1, 2, \ldots$. In general, the roots of different switching functions must not coincide (see also the discussion in Section 4.2).

Later on the higher order time derivatives $\ddot{\boldsymbol{x}}$ and $\dot{\boldsymbol{y}}$ will be required. Under the rank condition Eq. (4.19) these derivatives can be obtained from the definition of index-1 DAEs according to Definition 1.7 as at a *consistent* point $[t, \boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}, \boldsymbol{p}]$, $t \in [t_{n-1}, t_n[$, the following statement holds

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F}_n(t, \boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}, \boldsymbol{p}) = 0 \overset{Eq.\,(4.19)}{\Rightarrow} \left[\frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{F}_n}{\partial \dot{\boldsymbol{x}}}\right] \begin{bmatrix} \dot{\boldsymbol{y}} \\ \ddot{\boldsymbol{x}} \end{bmatrix} = -\left[\frac{\partial \boldsymbol{F}_n}{\partial t} + \frac{\partial \boldsymbol{F}_n}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}}\right]\,. \tag{4.20}$$

4.1.4.a Sensitivities of the Initial Values

In the sequel, we consider the initial time t_0 as an additional parameter. Then the sensitivities

$$\left[\frac{\partial \boldsymbol{x}_0(t_0;\boldsymbol{p})}{\partial \boldsymbol{p}}\right], \left[\frac{\partial \boldsymbol{y}_0(t_0;\boldsymbol{p})}{\partial \boldsymbol{p}}\right], \left[\frac{\partial \dot{\boldsymbol{x}}_0(t_0;\boldsymbol{p})}{\partial \boldsymbol{p}}\right], \text{ and } \left[\frac{\partial \boldsymbol{x}_0(t_0;\boldsymbol{p})}{\partial t_0}\right], \left[\frac{\partial \boldsymbol{y}_0(t_0;\boldsymbol{p})}{\partial t_0}\right], \left[\frac{\partial \dot{\boldsymbol{x}}_0(t_0;\boldsymbol{p})}{\partial t_0}\right],$$

of the solution of the *initial consistent initialisation problem*

$$\boldsymbol{F}_1(t, \boldsymbol{x}_0, \boldsymbol{y}_0, \dot{\boldsymbol{x}}_0, \boldsymbol{p}) = 0\,, \tag{4.21a}$$
$$\boldsymbol{h}_0(t, \boldsymbol{x}_0, \boldsymbol{y}_0, \dot{\boldsymbol{x}}_0, \boldsymbol{p}) = 0\,, \tag{4.21b}$$
$$t - t_0 = 0\,, \tag{4.21c}$$

are obtained by differentiation of Eqs. (4.21a)–(4.21c) with respect to the parameters $\boldsymbol{p}$ and with respect to the initial time t_0, respectively, and solution of the resulting linear system of equations

$$\begin{bmatrix} \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_1}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_0}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{h}_0}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{h}_0}{\partial \dot{\boldsymbol{x}}} \end{bmatrix} \begin{bmatrix} \frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{p}} & \frac{\partial \boldsymbol{x}_0}{\partial t_0} \\ \frac{\partial \boldsymbol{y}_0}{\partial \boldsymbol{p}} & \frac{\partial \boldsymbol{y}_0}{\partial t_0} \\ \frac{\partial \dot{\boldsymbol{x}}_0}{\partial \boldsymbol{p}} & \frac{\partial \dot{\boldsymbol{x}}_0}{\partial t_0} \end{bmatrix} = - \begin{bmatrix} \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{p}} & \frac{\partial \boldsymbol{F}_1}{\partial t} \\ \frac{\partial \boldsymbol{h}_0}{\partial \boldsymbol{p}} & \frac{\partial \boldsymbol{h}_0}{\partial t} \end{bmatrix}. \tag{4.22}$$

The trivial relations $\partial t/\partial \boldsymbol{p} = 0$ and $\partial t/\partial t_0 = 1$ are already considered in Eq. (4.22). Eq. (4.22) is (uniquely) solvable if the system matrix is of full rank $2n_{\boldsymbol{x}} + n_{\boldsymbol{y}}$, i.e., the initial conditions $\boldsymbol{h}_0$ have to be suitably chosen.

Actually, we are interested in the sensitivities of the *dynamic initial consistent initialisation problem*. On the one hand, the sensitivities of the dependent variables with respect to the parameters $\boldsymbol{p}$ need not to be modified, i.e.,

$$\boldsymbol{\rho}(t_0, \boldsymbol{p}) = \left[\frac{\partial \boldsymbol{x}_0(t_0;\boldsymbol{p})}{\partial \boldsymbol{p}}\right],\ \boldsymbol{\sigma}(t_0, \boldsymbol{p}) = \left[\frac{\partial \boldsymbol{y}_0(t_0;\boldsymbol{p})}{\partial \boldsymbol{p}}\right],\ \dot{\boldsymbol{\rho}}(t_0, \boldsymbol{p}) = \left[\frac{\partial \dot{\boldsymbol{x}}_0(t_0;\boldsymbol{p})}{\partial \boldsymbol{p}}\right],$$

where

$$\boldsymbol{\rho}(t, \boldsymbol{p}) := \left[\frac{\partial \boldsymbol{x}(t;\boldsymbol{p})}{\partial \boldsymbol{p}}\right]_{t,\boldsymbol{p}} \in \mathbb{R}^{n_{\boldsymbol{x}} \times n_{\boldsymbol{p}}} \text{ and } \boldsymbol{\sigma}(t, \boldsymbol{p}) := \left[\frac{\partial \boldsymbol{y}(t;\boldsymbol{p})}{\partial \boldsymbol{p}}\right]_{t,\boldsymbol{p}} \in \mathbb{R}^{n_{\boldsymbol{y}} \times n_{\boldsymbol{p}}}$$

denote the parametric sensitivity functions for the solution trajectory of the DAE Eq. (4.18). But on the other hand, as we identify $\boldsymbol{z}(t;\boldsymbol{p})$ with $\boldsymbol{z}(t;t_0,\boldsymbol{p})$, $\boldsymbol{z} \in \{\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}\}$, due to

$$\begin{aligned} \left[\frac{\partial \boldsymbol{z}_0(t_0;\boldsymbol{p})}{\partial t_0}\right] = \left[\frac{\partial \boldsymbol{z}(t_0;t_0,\boldsymbol{p})}{\partial t_0}\right] &= \left[\frac{\partial \boldsymbol{z}(t;t_0,\boldsymbol{p})}{\partial t}\right]_{t_0,\boldsymbol{p}} \cdot \left[\frac{\partial t}{\partial t_0}\right]_{t_0,\boldsymbol{p}} + \left[\frac{\partial \boldsymbol{z}(t;t_0,\boldsymbol{p})}{\partial t_0}\right]_{t_0,\boldsymbol{p}} \\ &= \dot{\boldsymbol{z}}(t_0;\boldsymbol{p}) \cdot 1 + \left[\frac{\partial \boldsymbol{z}(t;t_0,\boldsymbol{p})}{\partial t_0}\right]_{t_0,\boldsymbol{p}} = \dot{\boldsymbol{z}}(t_0;\boldsymbol{p}) + \frac{\partial \boldsymbol{z}}{\partial t_0}(t_0, \boldsymbol{p})\,, \end{aligned}$$

the dynamic sensitivities with respect to t_0 become

$$\left[\frac{\partial \boldsymbol{x}(t_0,\boldsymbol{p})}{\partial t_0}\right] = \left[\frac{\partial \boldsymbol{x}_0(t_0,\boldsymbol{p})}{\partial t_0}\right] - \dot{\boldsymbol{x}}(t_0;\boldsymbol{p})\,,\ \left[\frac{\partial \boldsymbol{y}(t_0,\boldsymbol{p})}{\partial t_0}\right] = \left[\frac{\partial \boldsymbol{y}_0(t_0,\boldsymbol{p})}{\partial t_0}\right] - \dot{\boldsymbol{y}}(t_0;\boldsymbol{p})\,, \text{ and}$$
$$\left[\frac{\partial \dot{\boldsymbol{x}}(t_0,\boldsymbol{p})}{\partial t_0}\right] = \left[\frac{\partial \dot{\boldsymbol{x}}_0(t_0,\boldsymbol{p})}{\partial t_0}\right] - \ddot{\boldsymbol{x}}(t_0;\boldsymbol{p})\,.$$

The value $\dot{\boldsymbol{x}}(t_0;\boldsymbol{p})$ is available after consistent initialisation. The missing derivatives $\ddot{\boldsymbol{x}}(t_0;\boldsymbol{p})$ and $\dot{\boldsymbol{y}}(t_0;\boldsymbol{p})$ can be obtained from Eq. (4.20).

4.1.4.b Sensitivity Transfer Across Discontinuities

In addition to the notation introduced at the beginning of this section we now use $\boldsymbol{z}_n^- = \boldsymbol{z}_n^-(t_n;\boldsymbol{p}) := \lim_{t\nearrow t_n} \boldsymbol{z}(t;\boldsymbol{p})$ and $\boldsymbol{z}_n^+ = \boldsymbol{z}_n^+(t_n;\boldsymbol{p}) := \lim_{t\searrow t_n} \boldsymbol{z}(t;\boldsymbol{p})$ for $\boldsymbol{z} \in \{\boldsymbol{\rho}, \boldsymbol{\sigma}, \dot{\boldsymbol{\rho}}\}$, $\boldsymbol{F}_n^+ := \lim_{t\searrow t_n} \boldsymbol{F}_{n+1} = \boldsymbol{F}_{n+1}(t_n, \boldsymbol{x}_n^+, \boldsymbol{y}_n^+, \dot{\boldsymbol{x}}_n^+, \boldsymbol{p})$, and $\boldsymbol{q}_n^- := \lim_{t\nearrow t_n} \boldsymbol{q}_n = \boldsymbol{q}_n(t_n, \boldsymbol{x}_n^-, \boldsymbol{y}_n^-, \dot{\boldsymbol{x}}_n^-, \boldsymbol{p})$.

By definition, at a discontinuity $t = t_n$, $n = 1, 2, \ldots$, the system of equations

$$\boldsymbol{F}_{n+1}(t_n, \boldsymbol{x}_n^+, \boldsymbol{y}_n^+, \dot{\boldsymbol{x}}_n^+, \boldsymbol{p}) = 0\,, \tag{4.23a}$$

$$\boldsymbol{h}_n(t_n, \boldsymbol{x}_n^+, \boldsymbol{y}_n^+, \dot{\boldsymbol{x}}_n^+, \boldsymbol{x}_n^-, \boldsymbol{y}_n^-, \dot{\boldsymbol{x}}_n^-, \boldsymbol{p}) = 0\,, \tag{4.23b}$$

$$\boldsymbol{q}_n(t_n, \boldsymbol{x}_n^-, \boldsymbol{y}_n^-, \dot{\boldsymbol{x}}_n^-, \boldsymbol{p}) = 0\,, \tag{4.23c}$$

holds. The sensitivity of the switching time $t_n = t_n(\boldsymbol{p})$ with respect to the parameters $\boldsymbol{p}$ is obtained by total differentiation [(iii)] of the switching condition Eq. (4.23c)

$$\left[\frac{\partial \boldsymbol{q}_n^-}{\partial t} + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}}^-\right]\left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right] = - \left[\frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{x}}\boldsymbol{\rho}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{y}}\boldsymbol{\sigma}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \dot{\boldsymbol{x}}}\dot{\boldsymbol{\rho}}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{p}}\right],$$

under the transversality condition

$$\frac{\mathrm{d}\boldsymbol{q}_n^-}{\mathrm{d}t} = \left[\frac{\partial \boldsymbol{q}_n^-}{\partial t} + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \dot{\boldsymbol{x}}}\ddot{\boldsymbol{x}}^-\right] \neq 0\,.$$

Accordingly, total differentiation of Eqs. (4.23a)–(4.23b) with respect to the parameters $\boldsymbol{p}$ under consideration of the parametric dependency of the switching time gives the sensitivities of the *dynamic consistent initialisation problem*

$$\begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^+} \end{bmatrix} \begin{bmatrix} \boldsymbol{\rho}^+ \\ \boldsymbol{\sigma}^+ \\ \dot{\boldsymbol{\rho}}^+ \end{bmatrix} = - \begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^+} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}^+ \\ \dot{\boldsymbol{y}}^+ \\ \ddot{\boldsymbol{x}}^+ \end{bmatrix} \cdot \left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right]$$

$$- \begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial t} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{h}_n}{\partial t} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{p}} \end{bmatrix} \begin{bmatrix} \frac{\partial t_n}{\partial \boldsymbol{p}} \\ \boldsymbol{\rho}^- + \dot{\boldsymbol{x}}^- \cdot \frac{\partial t_n}{\partial \boldsymbol{p}} \\ \boldsymbol{\sigma}^- + \dot{\boldsymbol{y}}^- \cdot \frac{\partial t_n}{\partial \boldsymbol{p}} \\ \dot{\boldsymbol{\rho}}^- + \ddot{\boldsymbol{x}}^- \cdot \frac{\partial t_n}{\partial \boldsymbol{p}} \\ \boldsymbol{Id}_{n_{\boldsymbol{p}}} \end{bmatrix}. \tag{4.24}$$

This linear system of equations is solvable if

$$\operatorname{rank}\left(\begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^+} \end{bmatrix}\right) = n_{\boldsymbol{x}} + n_{\boldsymbol{y}} + n_{\boldsymbol{x}}\,,$$

[(iii)] E.g., we have $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{p}} = \frac{\mathrm{d}\boldsymbol{x}(t_n;\boldsymbol{p})}{\mathrm{d}\boldsymbol{p}} = \frac{\mathrm{d}\boldsymbol{x}(t_n(\boldsymbol{p});\boldsymbol{p})}{\mathrm{d}\boldsymbol{p}} = \frac{\partial \boldsymbol{x}(t,\boldsymbol{p})}{\partial t} \cdot \frac{\partial t_n(\boldsymbol{p})}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{x}(t,\boldsymbol{p})}{\partial \boldsymbol{p}}$.

i.e., if the consistent initialisation problem is well posed in the sense that the transition conditions are compatible with the DAE.

Remark 4.6:
In case of ODEs with autonomous and explicit transition conditions [Feeh 98] shows the coincidence of Eq. (4.24) with the expressions derived in [Roze 67] (see Section 4.1.1, Eq. (4.3)). ◊

Similar to the sensitivities of the dynamic initial consistent initialisation problem at the initial time $t = t_0$ treated in Section 4.1.4.a above, the sensitivities of the *dynamic consistent initialisation problem* at discontinuities can be obtained in two steps. In the first step, solely the sensitivities of the consistent initialisation problem contained in Eqs. (4.23a)–(4.23b) are considered. Differentiation of Eqs. (4.23a)–(4.23b) with respect to the parameters neglecting the parametric dependency of the switching time gives

$$\begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^+} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\rho}} \\ \tilde{\boldsymbol{\sigma}} \\ \tilde{\dot{\boldsymbol{\rho}}} \end{bmatrix} = - \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{p}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\rho}^- \\ \boldsymbol{\sigma}^- \\ \dot{\boldsymbol{\rho}}^- \\ \boldsymbol{Id}_{n_{\boldsymbol{p}}} \end{bmatrix},$$

where $\tilde{\boldsymbol{\rho}}$ and $\tilde{\boldsymbol{\sigma}}$ denote the sensitivities of the consistent initialisation problem. In the second step, the sensitivities of the dynamic consistent initialisation problem are obtained from

$$\begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^+} \end{bmatrix} \begin{bmatrix} \boldsymbol{\rho}^+ \\ \boldsymbol{\sigma}^+ \\ \dot{\boldsymbol{\rho}}^+ \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^+} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\rho}} \\ \tilde{\boldsymbol{\sigma}} \\ \tilde{\dot{\boldsymbol{\rho}}} \end{bmatrix}$$
$$- \left\{ \begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \boldsymbol{y}} & \frac{\partial \boldsymbol{F}_n^+}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^+} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^+} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}^+ \\ \dot{\boldsymbol{y}}^+ \\ \ddot{\boldsymbol{x}}^+ \end{bmatrix} + \begin{bmatrix} \frac{\partial \boldsymbol{F}_n^+}{\partial t} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \boldsymbol{h}_n}{\partial t} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^-} & \frac{\partial \boldsymbol{h}_n}{\partial \dot{\boldsymbol{x}}^-} \end{bmatrix} \begin{bmatrix} 1 \\ \dot{\boldsymbol{x}}^- \\ \dot{\boldsymbol{y}}^- \\ \ddot{\boldsymbol{x}}^- \end{bmatrix} \right\} \cdot \begin{bmatrix} \frac{\partial t_n}{\partial \boldsymbol{p}} \end{bmatrix}. \tag{4.25}$$

Remark 4.7:
If the switching time t_n does not depend on the parameters the sensitivities of the consistent initialisation problem and of the dynamic initialisation problem are identical, as can be seen from Eq. (4.25). ◊

4.2 Multi-Dimensional Switching Functions

In most literature a single, real-valued switching function is considered. However, in practice it is often desirable to treat a vector of real-valued switching functions, say, $\boldsymbol{q} = \boldsymbol{q}(t, \boldsymbol{\xi}, \boldsymbol{p}) = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{m_{\boldsymbol{q}}}]^T(t, \boldsymbol{\xi}, \boldsymbol{p})$, $\boldsymbol{q} \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}}, \mathbb{R}^{m_{\boldsymbol{q}}}\right)$, $\boldsymbol{q}_\nu \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}}, \mathbb{R}\right)$, $\nu = 1, \ldots, m_{\boldsymbol{q}}$. In such cases usually several independent

real-valued switching functions are implemented, each of them characterising a potential modification of a subset of the model equations.

Remark 4.8:
Please note that this type of switching functions differs from the switching functions introduced in Definition 1.22 (description of hybrid dynamical systems) which mark the transition to a new dynamical system in the sense that the entire model is replaced by a new model. ◊

The determination of the discontinuity must be modified in such a way that all switching functions $\boldsymbol{q}_\nu$, $\nu = 1, \ldots, m_{\boldsymbol{q}}$, are traced simultaneously during integration. Without loss of generality assume that in the integration step from $t^{\mu-1}$ to t^μ, $t_0 \leq t^{\mu-1} < t^\mu \leq t_\mathrm{f}$, for the first time some of the switching functions have a root. The discontinuity is then uniquely determined as the location of the earliest root

$$t_\mathrm{disc} := \min \left\{ t \in [t^{\mu-1}, t^\mu] \,\middle|\, \exists \nu \in \{1, \ldots, m_{\boldsymbol{q}}\} : \boldsymbol{q}_\nu(t, \boldsymbol{\xi}(t; \boldsymbol{p}), \boldsymbol{p}) = 0 \right\} , \qquad (4.26)$$

where $\boldsymbol{\xi}(t; \boldsymbol{p})$ is the solution trajectory of the underlying dynamical system.

In the context of optimal control, problems arise at points where several of the switching functions $\boldsymbol{q}_\nu$ vanish simultaneously, cf., e.g., [Feeh 98]. Such *critical points* indicate a potential change in the *sequence* of discontinuities, i.e., they mark the boundary between different realisations of a hybrid dynamical system. Thus at critical points in the parameter space the parametric sensitivities do not exist in the general case (see Section 4.2.1 below).

A similar problem can be observed already for a single real-valued switching function ($m_{\boldsymbol{q}} = 1$) if the trajectory meets the switching manifold at a touching point. As above the discontinuity sequence can change for a small variation in the parameters. But in contrast to the multi-dimensional setting where critical points are somewhat natural, in the scalar setting such critical points are commonly regarded as special degenerate cases. Therefore, touching points are in general excluded by the transversality conditions

$$\boldsymbol{q}_\nu(t, \boldsymbol{\xi}(t; \boldsymbol{p}), \boldsymbol{p}) = 0 \quad \Rightarrow \quad \left. \frac{\mathrm{d}\boldsymbol{q}_\nu}{\mathrm{d}t} \right|_{t, \boldsymbol{p}} \neq 0 , \ \nu = 1, \ldots, m_{\boldsymbol{q}} . \qquad (4.27)$$

Remark 4.9:
The transversality condition Eq. (4.27) simplifies the procedure of locating discontinuities. This condition in general allows to detect the presence of discontinuities numerically advantageously by tracing the signs of the switching functions (for limitations of this method see the discussion at the end of Section 1.3.2). ◊

4.2.1 Nonexistence of the Parametric Sensitivities in the General Case

Here we introduce a counter-example to the existence of parametric sensitivities in the general case of a multi-dimensional switching function. To this purpose

consider the ODE-IVP

$$\begin{aligned} 0 &= \dot{\boldsymbol{\xi}}(t;\boldsymbol{p}) - 1\,; \quad t \in [t_0, t_\mathrm{f}] :\equiv [-1, 1]\,, \\ 0 &= \boldsymbol{\xi}(t_0;\boldsymbol{p}) + 1\,, \end{aligned} \tag{4.28}$$

with the solution $\boldsymbol{\xi}(t;\boldsymbol{p}) = t$. Additionally, we introduce the two-dimensional switching function

$$\boldsymbol{q}(t,\boldsymbol{\xi},\boldsymbol{p}) := \begin{bmatrix} \boldsymbol{q}_1(t,\boldsymbol{\xi},\boldsymbol{p}) \\ \boldsymbol{q}_2(t,\boldsymbol{\xi},\boldsymbol{p}) \end{bmatrix} := \begin{bmatrix} \boldsymbol{\xi} + \boldsymbol{p} \\ 2\boldsymbol{\xi} - \boldsymbol{p} \end{bmatrix}.$$

The roots of $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ along the solution $\boldsymbol{\xi}(t;\boldsymbol{p})$ of Eq. (4.28) are given by

$$\hat{t}_1(\boldsymbol{p}) = -\boldsymbol{p}\,, \text{ and } \hat{t}_2(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}\,,$$

with the parametric sensitivities

$$\frac{\partial \hat{t}_1}{\partial \boldsymbol{p}}(\boldsymbol{p}) \;=\; -1\,, \text{ and } \frac{\partial \hat{t}_2}{\partial \boldsymbol{p}}(\boldsymbol{p}) \;=\; \frac{1}{2}\,.$$

According to rule Eq. (4.26) the switching point is determined as the earliest root, cf. Eq. (4.26). In the example case the first switching point is characterised by the $\mathcal{C}^0$ function

$$t_\mathrm{disc}(\boldsymbol{p}) := \min\left\{\hat{t}_1(\boldsymbol{p}), \hat{t}_2(\boldsymbol{p})\right\} = \begin{cases} \frac{1}{2}\boldsymbol{p}\,; & \boldsymbol{p} < 0\,, \\ 0\,; & \boldsymbol{p} = 0\,, \\ -\boldsymbol{p}\,; & \boldsymbol{p} > 0\,. \end{cases} \tag{4.29}$$

Consider the point in the parameter space $\hat{\boldsymbol{p}} = 0$. The switching point is then $t_\mathrm{disc}(\hat{\boldsymbol{p}}) = 0$. Although both switching functions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ fulfil the transversality condition Eq. (4.27) at $t = t_\mathrm{disc}(\hat{\boldsymbol{p}})$, the parametric sensitivity $\partial t_\mathrm{disc}(\boldsymbol{p})/\partial \boldsymbol{p}$ of the switching time determined by Eq. (4.29) is not defined at $\boldsymbol{p} = \hat{\boldsymbol{p}}$.

Remark 4.10:
The example does not fully represent realistic problems as here two different switching functions govern the same (and only) part of the model equations. However, the sole intention of this example is to provide a simple case of non-differentiability with respect to the parameters. ◊

Remark 4.11:
Consider the simulation tool OPTISIM® which provides the basis of our work. In the case of purely time dependent switching functions we have often observed that several switching functions are activated simultaneously during a dynamical simulation. In such cases the models contain, e.g., several ramp functions with the same fixed duration of the ramp. On the other hand, we have rarely observed a coinciding change of sign for state dependent switching functions, cf. Remark 4.13 on page 167. ◊

4.2.2 Existence of the Parametric Sensitivities in a Special Case

In spite of the counter-example given in Section 4.2.1 a simultaneous change of sign in multiple switching functions does not necessarily imply non-existence of the sensitivity matrices at the discontinuity. In special cases existence of the parametric sensitivity of the point of discontinuity can be proven, as shown in the following Theorem 4.1.

Theorem 4.1 (Parametric Sensitivities for Multiple Switching Functions) *Consider the multi-dimensional switching function* $\boldsymbol{q}(t, \boldsymbol{\xi}, \boldsymbol{p}) \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}}, \mathbb{R}^{m_{\boldsymbol{q}}}\right)$ *composed of* $m_{\boldsymbol{q}}$ *real-valued switching functions* $\boldsymbol{q}_\nu(t, \boldsymbol{\xi}, \boldsymbol{p}) \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{\xi}}+n_{\boldsymbol{p}}}, \mathbb{R}\right)$, $\nu = 1, \ldots, m_{\boldsymbol{q}}$. *Further let there be a sufficiently smooth trajectory* $\boldsymbol{\xi} : [t, \boldsymbol{p}] \mapsto \boldsymbol{\xi}(t; \boldsymbol{p}) \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$.

Assume that for a fixed parameter value $\hat{\boldsymbol{p}}$ *at time* $\hat{t}(\hat{\boldsymbol{p}})$ *all switching functions vanish*

$$\boldsymbol{q}_\nu(\hat{t}(\hat{\boldsymbol{p}}), \boldsymbol{\xi}(\hat{t}(\hat{\boldsymbol{p}}); \hat{\boldsymbol{p}}), \hat{\boldsymbol{p}}) = 0\,;\ \nu = 1, \ldots, m_{\boldsymbol{q}}\,,$$

and that each one fulfils the sign and transversality conditions $\boldsymbol{q}_\nu^- < 0$, $\boldsymbol{q}_\nu^+ \geq 0$, *and* $d\boldsymbol{q}_\nu^-/dt > 0$ *at this point. By the implicit function theorem the location of the root for each real-valued switching function* $\boldsymbol{q}_\nu$, $\nu = 1, \ldots, m_{\boldsymbol{q}}$, *can be described at least locally around* $(\hat{t}(\hat{\boldsymbol{p}}), \boldsymbol{\xi}(\hat{t}(\hat{\boldsymbol{p}}); \hat{\boldsymbol{p}}), \hat{\boldsymbol{p}})$ *by a continuously differentiable function* $\hat{t}_\nu(\boldsymbol{p}) : \mathbb{R}^{n_{\boldsymbol{p}}} \to \mathbb{R}$.

Suppose that the following condition holds:

$$\left.\frac{\partial \hat{t}_1(\boldsymbol{p})}{\partial \boldsymbol{p}}\right|_{\hat{\boldsymbol{p}}} = \left.\frac{\partial \hat{t}_2(\boldsymbol{p})}{\partial \boldsymbol{p}}\right|_{\hat{\boldsymbol{p}}} = \ldots = \left.\frac{\partial \hat{t}_{m_{\boldsymbol{q}}}(\boldsymbol{p})}{\partial \boldsymbol{p}}\right|_{\hat{\boldsymbol{p}}}. \tag{4.30}$$

Then the multi-dimensional switching function $\boldsymbol{q}$ *can be substituted in a neighbourhood of* $(\hat{t}(\hat{\boldsymbol{p}}), \boldsymbol{\xi}(\hat{t}(\hat{\boldsymbol{p}}); \hat{\boldsymbol{p}}), \hat{\boldsymbol{p}})$ *by the continuously differentiable one-dimensional function*

$$\hat{\boldsymbol{q}}(t, \boldsymbol{\xi}, \boldsymbol{p}) := \sum_{\nu=1}^{m_{\boldsymbol{q}}} \boldsymbol{q}_\nu(t, \boldsymbol{\xi}, \boldsymbol{p})\,.$$

Here, the trajectory $\boldsymbol{\xi}(t; \boldsymbol{p})$ *is to be interpreted as extended smoothly to some sufficiently long interval* $[t_0, \hat{t}(\hat{\boldsymbol{p}}) + \delta t[$ without *execution of the discontinuity.*

Proof:
By construction, $\hat{\boldsymbol{q}}$ inherits continuous differentiability from the switching functions $\boldsymbol{q}_\nu$, $\nu = 1, \ldots, m_{\boldsymbol{q}}$. The function $\hat{\boldsymbol{q}}$ satisfies the transversality condition $\mathrm{d}\hat{\boldsymbol{q}}_\nu^-/\mathrm{d}t > 0$ by linearity. Also, the sign conditions $\hat{\boldsymbol{q}}^- < 0, \hat{\boldsymbol{q}}^+ \geq 0$ hold for $\hat{\boldsymbol{q}}$, as otherwise at least one of the switching functions $\boldsymbol{q}_\nu$ must have a change of sign at a different time than others. In a neighbourhood of $\hat{\boldsymbol{p}}$ this is not possible due to Eq. (4.30). Especially, the switching point $t_{\mathrm{disc}}(\boldsymbol{p})$, defined as the root of $\hat{\boldsymbol{q}}$ at $(\hat{t}(\hat{\boldsymbol{p}}), \boldsymbol{\xi}(\hat{t}(\hat{\boldsymbol{p}}); \hat{\boldsymbol{p}}), \hat{\boldsymbol{p}},)$ is unique. □

Condition Eq. (4.30) is equivalent to

$$\exists \rho \in \mathbb{R}^{1\times n_{\boldsymbol{p}}} : \forall \nu \in \{1,\dots m_{\boldsymbol{q}}\} : \quad \frac{\partial \hat{t}_\nu(\boldsymbol{p})}{\partial \boldsymbol{p}}\Big|_{\hat{\boldsymbol{p}}} = -\left[\begin{matrix}\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}+\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{p}}\\ \frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}\end{matrix}\right]_{\hat{t}(\hat{\boldsymbol{p}}),\hat{\boldsymbol{p}}} = \rho\,.$$

Therefore,

$$\left[\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}+\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{p}}\right]_{\hat{t}(\hat{\boldsymbol{p}}),\hat{\boldsymbol{p}}} = -\left[\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}\right]_{\hat{t}(\hat{\boldsymbol{p}}),\hat{\boldsymbol{p}}}\cdot\rho;\ \nu=1,\dots,m_{\boldsymbol{q}}\,,$$

and

$$\begin{aligned}\frac{\partial t_{\mathrm{disc}}(\boldsymbol{p})}{\partial \boldsymbol{p}}\Big|_{\hat{\boldsymbol{p}}} &= -\left[\begin{matrix}\frac{\partial \hat{\boldsymbol{q}}}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}+\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{p}}\\ \frac{\partial \hat{\boldsymbol{q}}}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}\end{matrix}\right]_{\hat{t}(\hat{\boldsymbol{p}}),\hat{\boldsymbol{p}}} = -\left[\begin{matrix}\sum_{\nu=1}^{m_{\boldsymbol{q}}}\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}+\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{p}}\\ \sum_{\nu=1}^{m_{\boldsymbol{q}}}\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}\end{matrix}\right]_{\hat{t}(\hat{\boldsymbol{p}}),\hat{\boldsymbol{p}}}\\ &= -\left[\frac{\sum_{\nu=1}^{m_{\boldsymbol{q}}}\left(-\left[\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}\right]\cdot\rho\right)}{\sum_{\nu=1}^{m_{\boldsymbol{q}}}\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}}\right]_{\hat{t}(\hat{\boldsymbol{p}}),\hat{\boldsymbol{p}}} = \left[\frac{\left(\sum_{\nu=1}^{m_{\boldsymbol{q}}}\left[\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}\right]\right)\cdot\rho}{\sum_{\nu=1}^{m_{\boldsymbol{q}}}\frac{\partial \boldsymbol{q}_\nu}{\partial \boldsymbol{\xi}}\frac{\partial \dot{\boldsymbol{\xi}}}{\partial t}+\frac{\partial \boldsymbol{q}_\nu}{\partial t}}\right]_{\hat{t}(\hat{\boldsymbol{p}}),\hat{\boldsymbol{p}}}\\ &= \rho\,.\end{aligned}$$

This means that the real-valued substitute $\hat{\boldsymbol{q}}$ for the multi-dimensional switching function $\boldsymbol{q}$ preserves the parametric sensitivity of the switching point.

4.2.3 Problems with Discontinuous Sensitivities in Direct Optimal Control Algorithms

In Section 2.4.2 we have discussed a direct shooting-type optimal control algorithm. Just to recall, the idea is to transform the variational problem into a nonlinear programming problem (NLP) by parameterisation of the controls. For every parameter value of interest the model is integrated numerically. The resulting NLP can be solved efficiently by SQP methods which are based on the availability of *continuously differentiable* first order derivatives of the objective function and of the point constraints with respect to the parameters. These derivatives are obtained via parametric sensitivities.

One advantage of direct methods is that *in principle* no *a priori* knowledge of switching events is required. In contrast, the user of an indirect optimal control algorithm (e.g., transformation of the variational problem into a multi-point boundary value problem (*MPBVP*) based on the first order necessary conditions for optimality and solution of the MPBVP with a multiple shooting code (*indirect multiple shooting*, cf. Section 2.4.1)) has to provide the switching structure of the problem, cf., e.g, [BMP 91a], [BMP 91b], [Pesc 94].

However, the discussion in Section 4.2.1 and Section 4.2.2 above shows that also for direct methods the switching structure can be of importance. This is the case when a modification of the optimisation parameters (within the admissible parameter domain) leads to a *different* switching structure. When the switching

structure changes, smoothness properties (SQP: local $\mathcal{C}^2$ differentiability of the objective function and of the constraints with respect to the optimisation parameters) may not hold. As a consequence a conventional SQP optimisation algorithm may have difficulties in finding a solution of the optimisation problem.

Due to its inherent *combinatorial nature* the adequate form of the optimisation problem in this setting may be a mixed-integer nonlinear programming problem (*MINLP*) [Feeh 98]. This is the natural extension of the NLP obtained by the control parameterisation approach when control parameterisation is applied to hybrid dynamical systems described in Definition 1.22. However, mixed-integer dynamic optimisation (*MIDO*) introduces an additional order of complexity which makes its direct application to our class of optimal control problems difficult.

As a possible approach in the context of batch process optimisation [Allg 97] proposes simplified *screening models* in order to obtain numerically cheap initial bounds for the various scenarios. Another method employed by [Stry 00], [StGl 01] is to introduce a continuous relaxation of the MIDO.

4.3 Two Tailored Algorithms for the Sensitivity Transfer Across Discontinuities

4.3.1 Problem Setting

In the sequel two algorithms for the sensitivity transfer in parameter dependent, discontinuous, and large-scale semi-explicit index-2 DAEs of the form

$$0 = \boldsymbol{f}_n(t, \boldsymbol{x}(t;\boldsymbol{p}), \boldsymbol{y}(t;\boldsymbol{p}), \boldsymbol{p}) - \dot{\boldsymbol{x}}(t;\boldsymbol{p})\,,\ t \in [t_{n-1}, t_n]\,,\ n = 1, 2, \dots\,, \tag{4.31a}$$

$$0 = \boldsymbol{g}_n(t, \boldsymbol{x}(t;\boldsymbol{p}), \boldsymbol{y}(t;\boldsymbol{p}), \boldsymbol{p})\,, \tag{4.31b}$$

with $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$, $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$, $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$, $\boldsymbol{f}_n : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{x}}}$, and $\boldsymbol{g}_n : \mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{n_{\boldsymbol{y}}}$ are developed. These systems are similar to the DAEs already discussed in Section 3.2. Each switching point t_n, i.e., each transition from model n to model $n+1$, $n = 1, 2, \dots$, is indicated by the root of a switching function

$$0 = \boldsymbol{q}_n(t_n, \boldsymbol{x}^-(t_n;\boldsymbol{p}), \boldsymbol{y}^-(t_n;\boldsymbol{p}), \boldsymbol{p})\,, \tag{4.32}$$

$\boldsymbol{q}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{p}}}, \mathbb{R}\right)$, where $\boldsymbol{x}^-(t_n;\boldsymbol{p}) = \lim_{t \nearrow t_n} \boldsymbol{x}(t;\boldsymbol{p})$ and $\boldsymbol{y}^-(t_n;\boldsymbol{p}) = \lim_{t \nearrow t_n} \boldsymbol{y}(t;\boldsymbol{p})$. The initial time t_0 is considered to be fixed. At $t = t_0$ the initial conditions are given by

$$0 = \boldsymbol{h}_0(t_0, \boldsymbol{x}(t_0;\boldsymbol{p}), \boldsymbol{y}(t_0;\boldsymbol{p}), \dot{\boldsymbol{x}}(t_0;\boldsymbol{p}), \boldsymbol{p})\,, \tag{4.33}$$

$\boldsymbol{h}_0 \in \mathcal{C}^1\left(\mathbb{R}^{1+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}}+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\mathrm{ddf}}(0)}\right)$. $n_{\mathrm{ddf}}(0)$ is the number of dynamic degrees of freedom of the model DAE at $t = t_0^+$. At $t = t_n$, $n = 1, 2, \dots$, the models are

connected by transition conditions

$$0 = \boldsymbol{h}_n(t_n, \boldsymbol{x}^+(t_n;\boldsymbol{p}), \boldsymbol{y}^+(t_n;\boldsymbol{p}), \dot{\boldsymbol{x}}^+(t_n;\boldsymbol{p}), \boldsymbol{x}^-(t_n;\boldsymbol{p}), \boldsymbol{y}^-(t_n;\boldsymbol{p}), \dot{\boldsymbol{x}}^-(t_n;\boldsymbol{p}), \boldsymbol{p}) \,, \tag{4.34}$$

$\boldsymbol{h}_n \in \mathcal{C}^1\left(\mathbb{R}^{1+2(n_{\boldsymbol{x}}+n_{\boldsymbol{y}}+n_{\boldsymbol{x}})+n_{\boldsymbol{p}}}, \mathbb{R}^{n_{\mathrm{ddf}}(n)}\right)$, where for each variable $\boldsymbol{z} \in \{\boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{x}}\}$ the values at the discontinuity from the left hand side and from right hand side are (again) defined by the limits $\boldsymbol{z}^-(t_n;\boldsymbol{p}) := \lim_{t \nearrow t_n} \boldsymbol{z}(t;\boldsymbol{p})$, and $\boldsymbol{z}^+(t_n;\boldsymbol{p}) := \lim_{t \searrow t_n} \boldsymbol{z}(t;\boldsymbol{p})$. $n_{\mathrm{ddf}}(n)$, $n = 1, 2, \ldots$, denotes the number of dynamic degrees of freedom at $t = t_n^+$.

In OPTISIM® the transition conditions Eq. (4.34) are not an explicitly formulated part of the simulation problem. Instead, the user relies on the BDF integrator to "determine" the transition conditions, without any means for external interference (cf. Section 3.1.10). In our first algorithm for the computation of sensitivities in discontinuous models presented in Section 4.3.2 we keep this paradigm. The algorithm extends the numerical differentiation approach of [ErAr 98] discussed in Section 4.1.3 by employing back-tracing as a means to evaluate the jump function.

In our second algorithm developed in Section 4.3.3 we compute the sensitivities of the dynamic consistent initialisation problem adapting the results of [Feeh 98], [GaBa 98], and [GFB 99] discussed in Section 4.1.4. In order to set up the consistent initialisation problem we employ our hybrid consistent initialisation algorithm introduced in Section 3.2. This algorithm automatically generates an adequate set of explicit transition conditions.

Remark 4.12:
In general, OPTISIM® uses initial conditions of the type $\dot{\boldsymbol{x}}(t_0) = 0$ in order to start an integration. However, for a typical dynamical model Eqs. (4.31a)–(4.31b) the system

$$\begin{aligned} 0 &= \boldsymbol{f}_1(t_0, \boldsymbol{x}(t_0;\boldsymbol{p}), \boldsymbol{y}(t_0;\boldsymbol{p}), \boldsymbol{p}) - 0 \,, \\ 0 &= \boldsymbol{g}_1(t_0, \boldsymbol{x}(t_0;\boldsymbol{p}), \boldsymbol{y}(t_0;\boldsymbol{p}), \boldsymbol{p}) \,, \end{aligned}$$

does not possess a well-defined solution due to the presence of *pure integrators*, e.g., the hold-up in a flash drum, cf. Example 3 on page 30. Therefore the initial values are determined by a modified steady-state model.

As a simple example consider the ODE $0 = \dot{\boldsymbol{x}}(t;\boldsymbol{p}) - \boldsymbol{p}$, $\boldsymbol{x} \in \mathbb{R}$, $\boldsymbol{p} \in \mathbb{R}$, which in the steady-state model may be replaced by the algebraic equation $0 = \boldsymbol{x}(t_0;\boldsymbol{p}) - \boldsymbol{x}_0(t_0;\boldsymbol{p})$. On the one hand this setting provides the required initial value, on the other hand it also allows for a start with a nonzero initial slope when $\boldsymbol{p} \neq 0$. In order to obtain correct results this means that we have to perform consistent initialisation at $t = t_0$ with explicit initial conditions Eq. (4.33) where the subset of the differential variables which is chosen as the dynamic degrees of freedom are assigned their values from the steady-state calculation. ◊

4.3.2 Sensitivity Transfer Using Back-Tracing

In Section 3.1.10 we have briefly discussed back-tracing as a method for the generation of *numerically consistent* initial conditions

$$\boldsymbol{\xi}^+(t_n;\boldsymbol{p}) := \begin{bmatrix} \boldsymbol{x}^+(t_n;\boldsymbol{p}) \\ \boldsymbol{y}^+(t_n;\boldsymbol{p}) \end{bmatrix} \,, \quad \text{and} \quad \dot{\boldsymbol{\xi}}^+(t_n;\boldsymbol{p}) := \begin{bmatrix} \dot{\boldsymbol{x}}^+(t_n;\boldsymbol{p}) \\ \dot{\boldsymbol{y}}^+(t_n;\boldsymbol{p}) \end{bmatrix} \,,$$

starting from

$$\boldsymbol{\xi}^-(t_n;\boldsymbol{p}) := \begin{bmatrix}\boldsymbol{x}^-(t_n;\boldsymbol{p})\\ \boldsymbol{y}^-(t_n;\boldsymbol{p})\end{bmatrix}, \quad \text{and} \quad \dot{\boldsymbol{\xi}}^-(t_n;\boldsymbol{p}) := \begin{bmatrix}\dot{\boldsymbol{x}}^-(t_n;\boldsymbol{p})\\ \dot{\boldsymbol{y}}^-(t_n;\boldsymbol{p})\end{bmatrix}.$$

The back-tracing method has been introduced in the context of linear DAEs. As far as we know currently no theoretical result is available establishing convergence of the back-tracing method applied to nonlinear semi-explicit index-2 DAEs. However, at the current state of our research we can provide some numerical results.

Back-tracing provides an algorithm for the approximation of numerically consistent initial conditions. The explicit form of the jump functions is not revealed. Therefore, we assume the functional dependencies

$$\boldsymbol{\xi}^+(t_n;\boldsymbol{p}) := \boldsymbol{\chi}_0(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}},\boldsymbol{p})\Big|_{t_n^-,\boldsymbol{p}} = \boldsymbol{\chi}_0(t_n,\boldsymbol{\xi}^-(t_n;\boldsymbol{p}),\dot{\boldsymbol{\xi}}^-(t_n;\boldsymbol{p}),\boldsymbol{p}), \tag{4.35a}$$

$$\dot{\boldsymbol{\xi}}^+(t_n;\boldsymbol{p}) := \boldsymbol{\chi}_1(t,\boldsymbol{\xi},\dot{\boldsymbol{\xi}},\boldsymbol{p})\Big|_{t_n^-,\boldsymbol{p}} = \boldsymbol{\chi}_1(t_n,\boldsymbol{\xi}^-(t_n;\boldsymbol{p}),\dot{\boldsymbol{\xi}}^-(t_n;\boldsymbol{p}),\boldsymbol{p}), \tag{4.35b}$$

for the jump functions, where $\boldsymbol{\chi}_0(\cdot)$ and $\boldsymbol{\chi}_1(\cdot)$ are defined as the results of the back-tracing method for the state variables, and for the first order derivatives of the state variables with respect to time, respectively, cf. Section 3.1.10. The relations Eqs. (4.35a)–(4.35b) have been derived from the properties of the numerical integration algorithm employed. The point is that after each discontinuity the variable-order BDF method is restarted with lowest order one (i.e., implicit Euler) in order to drop inaccurate higher order derivative information.

Following Section 4.1.4.b we split the sensitivity computation for the consistent initialisation problem into a pure initialisation part (neglecting $\partial t_n/\partial \boldsymbol{p}$), and a dynamic initialisation part (taking into account $\partial t_n/\partial \boldsymbol{p}$). Then on the one hand starting from Eqs. (4.35a)–(4.35b) the sensitivities for the solution of the initialisation problem based on back-tracing are given by

$$\widetilde{\boldsymbol{\omega}} = \frac{\partial \boldsymbol{\chi}_0}{\partial \boldsymbol{\xi}^-}\frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\chi}_0}{\partial \dot{\boldsymbol{\xi}}^-}\frac{\partial \dot{\boldsymbol{\xi}}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\chi}_0}{\partial \boldsymbol{p}}, \quad \text{and} \quad \widetilde{\dot{\boldsymbol{\omega}}} = \frac{\partial \boldsymbol{\chi}_1}{\partial \boldsymbol{\xi}^-}\frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\chi}_1}{\partial \dot{\boldsymbol{\xi}}^-}\frac{\partial \dot{\boldsymbol{\xi}}^-}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\chi}_1}{\partial \boldsymbol{p}}, \tag{4.36}$$

while on the other hand differentiation of Eqs. (4.35a)–(4.35b) with respect to the parameters under consideration of the parametric dependency of the switching time t_n gives the sensitivities of the dynamic initialisation problem

$$\frac{\partial \boldsymbol{\xi}^+}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\xi}^+}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{\chi}_0}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\chi}_0}{\partial \boldsymbol{\xi}^-}\left[\frac{\partial \boldsymbol{\xi}^-}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}}\right] + \frac{\partial \boldsymbol{\chi}_0}{\partial \dot{\boldsymbol{\xi}}^-}\left[\frac{\partial \dot{\boldsymbol{\xi}}^-}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \dot{\boldsymbol{\xi}}^-}{\partial \boldsymbol{p}}\right] + \frac{\partial \boldsymbol{\chi}_0}{\partial \boldsymbol{p}},$$

$$\frac{\partial \dot{\boldsymbol{\xi}}^+}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \dot{\boldsymbol{\xi}}^+}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{\chi}_1}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\chi}_1}{\partial \boldsymbol{\xi}^-}\left[\frac{\partial \boldsymbol{\xi}^-}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}}\right] + \frac{\partial \boldsymbol{\chi}_1}{\partial \dot{\boldsymbol{\xi}}^-}\left[\frac{\partial \dot{\boldsymbol{\xi}}^-}{\partial t}\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \dot{\boldsymbol{\xi}}^-}{\partial \boldsymbol{p}}\right] + \frac{\partial \boldsymbol{\chi}_1}{\partial \boldsymbol{p}}.$$

After reordering of the latter equations and application of Eq. (4.36) we obtain

$$\begin{aligned}\boldsymbol{\omega}^+ = \frac{\partial \boldsymbol{\xi}^+}{\partial \boldsymbol{p}} &= -\left[\dot{\boldsymbol{\xi}}^+ - \frac{\partial \boldsymbol{\chi}_0}{\partial t} - \frac{\partial \boldsymbol{\chi}_0}{\partial \boldsymbol{\xi}^-}\dot{\boldsymbol{\xi}}^- - \frac{\partial \boldsymbol{\chi}_0}{\partial \dot{\boldsymbol{\xi}}^-}\ddot{\boldsymbol{\xi}}^-\right]\frac{\partial t_n}{\partial \boldsymbol{p}} + \widetilde{\boldsymbol{\omega}} \\ &= -\left[\dot{\boldsymbol{\xi}}^+ - \frac{\mathrm{d}\boldsymbol{\chi}_0}{\mathrm{d}t}\right]\frac{\partial t_n}{\partial \boldsymbol{p}} + \widetilde{\boldsymbol{\omega}}, \qquad (4.37a)\\ \dot{\boldsymbol{\omega}}^+ = \frac{\partial \dot{\boldsymbol{\xi}}^+}{\partial \boldsymbol{p}} &= -\left[\ddot{\boldsymbol{\xi}}^+ - \frac{\partial \boldsymbol{\chi}_1}{\partial t} - \frac{\partial \boldsymbol{\chi}_1}{\partial \boldsymbol{\xi}^-}\dot{\boldsymbol{\xi}}^- - \frac{\partial \boldsymbol{\chi}_1}{\partial \dot{\boldsymbol{\xi}}^-}\ddot{\boldsymbol{\xi}}^-\right]\frac{\partial t_n}{\partial \boldsymbol{p}} + \widetilde{\dot{\boldsymbol{\omega}}} \\ &= -\left[\ddot{\boldsymbol{\xi}}^+ - \frac{\mathrm{d}\boldsymbol{\chi}_1}{\mathrm{d}t}\right]\frac{\partial t_n}{\partial \boldsymbol{p}} + \widetilde{\dot{\boldsymbol{\omega}}}, \qquad (4.37b)\end{aligned}$$

as expressions for the sensitivity transfer at discontinuities using back-tracing.

4.3.2.a Back-Tracing and Numerical Differentiation of the Jump Function

As basis for our algorithm the technique presented in Section 4.1.3.a is employed. The main extension is in the utilisation of back-tracing in order to evaluate the jump function, together with formulae Eqs. (4.37a)–(4.37b) for the sensitivity transfer. Additionally, in the handling of a multi-dimensional switching function the special case discussed in Section 4.2.2 is considered. The result is given in the subsequent Algorithm 10.

Note that in Algorithm 10 the time of the discontinuity is denoted as t_{disc} instead of t_n as a vector of real-valued switching functions is to be treated instead of a single specific switching function $\boldsymbol{q}_n$. Additionally, we identify $\boldsymbol{q}(t,\boldsymbol{\xi},\boldsymbol{p}) := [\boldsymbol{q}_1,\ldots,\boldsymbol{q}_{m_{\boldsymbol{q}}}]^T(t,\boldsymbol{\xi},\boldsymbol{p})$ with $\boldsymbol{q}(t,\boldsymbol{x},\boldsymbol{y},\boldsymbol{p}) := [\boldsymbol{q}_1,\ldots,\boldsymbol{q}_{m_{\boldsymbol{q}}}]^T(t,\boldsymbol{x},\boldsymbol{y},\boldsymbol{p})$, where $m_{\boldsymbol{q}}$ is the number of switching functions in the model exhibiting a root at $t = t_{\mathrm{disc}}$. I.e., in the notation of Algorithm 10 $\boldsymbol{q}$ only contains *active* switching functions. In general, $m_{\boldsymbol{q}}$ and the subset of all switching functions in the model which change their sign usually differ from one switching point to another.

Algorithm 10 (Sensitivities after Discontinuities)

(i) Approximate the derivatives of the switching functions $\boldsymbol{q}_\mu$, $\mu = 1,\ldots,m_{\boldsymbol{q}}$, with respect to time t by forward difference approximation:

(a) Determine the disturbances for the finite difference approximation

$$\epsilon_t := \Delta t_{min} \quad \textit{(minimum integrator step size)},$$

$$\epsilon_{\boldsymbol{\xi}} := \begin{cases} \sqrt{\epsilon_{mach}} \cdot \left(1 + \left\|\boldsymbol{\xi}^-\right\|\right) / \left\|\dot{\boldsymbol{\xi}}^-\right\|; & \left\|\dot{\boldsymbol{\xi}}^-\right\| > 0\,, \\ 0; & \textit{otherwise}. \end{cases}$$

(b) Apply the finite difference approximations

$$\left[\frac{\partial \boldsymbol{q}}{\partial t}\right] \doteq \left[\frac{\boldsymbol{q}(t_{disc} + \epsilon_t, \boldsymbol{\xi}^-, \boldsymbol{p}) - \boldsymbol{q}(t_{disc}, \boldsymbol{\xi}^-, \boldsymbol{p})}{\epsilon_t}\right],$$
$$\left[\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}^-}{\partial t}\right] \doteq \left[\frac{\boldsymbol{q}(t_{disc}, \boldsymbol{\xi}^- + \epsilon_{\boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^-, \boldsymbol{p}) - \boldsymbol{q}(t_{disc}, \boldsymbol{\xi}^-, \boldsymbol{p})}{\epsilon_{\boldsymbol{\xi}}}\right].$$

(ii) For each *parameter* $\boldsymbol{p}_\nu$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$, *execute:*

(1) Approximate the derivatives of the switching functions $\boldsymbol{q}_\mu^-$, $\mu = 1, \ldots, m_{\boldsymbol{q}}$, *with respect to parameter* $\boldsymbol{p}_\nu$ *by forward difference approximation.*

(a) Determine the disturbance for the finite difference approximation

$$\epsilon_{\boldsymbol{p}_\nu} := \sqrt{\epsilon_{mach}} \cdot \left(1 + \left\|\left[\boldsymbol{\xi}^-, \boldsymbol{p}\right]\right\|\right) / \left\|\left[[\boldsymbol{\omega}^-]_\nu, 1\right]\right\|.$$

(b) Apply the finite difference approximation

$$\left[\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}_\nu} + \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}_\nu}\right] \doteq \left[\frac{\boldsymbol{q}\left(t_{disc}, \boldsymbol{\xi}^- + \epsilon_{\boldsymbol{p}_\nu} \cdot [\boldsymbol{\omega}^-]_\nu, \boldsymbol{p} + \epsilon_{\boldsymbol{p}_\nu}\boldsymbol{e}_\nu\right) - \boldsymbol{q}\left(t_{disc}, \boldsymbol{\xi}^-, \boldsymbol{p}\right)}{\epsilon_{\boldsymbol{p}_\nu}}\right].$$

(2) Approximate the derivative of the switching time t_{disc} *with respect to parameter* $\boldsymbol{p}_\nu$ *and check for the special case.*

(a) Approximate the (possibly different) derivatives of t_{disc} *with respect to parameter* $\boldsymbol{p}_\nu$ *as determined by* each *of the switching functions* $\boldsymbol{q}_\mu$, $\mu = 1, \ldots, m_{\boldsymbol{q}}$

$$[\widetilde{\rho}_{\nu,\mu}] := -\left[\frac{\frac{\partial \boldsymbol{q}_\mu}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}_\nu} + \frac{\partial \boldsymbol{q}_\mu}{\partial \boldsymbol{p}_\nu}}{\frac{\partial \boldsymbol{q}_\mu}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}^-}{\partial t} + \frac{\partial \boldsymbol{q}_\mu}{\partial t}}\right]_{\mu=1,\ldots,m_{\boldsymbol{q}}}.$$

(A vanishing denominator indicates that a switching function violates the transversality condition. In this case exit with an error.)

(b) If $m_{\boldsymbol{q}} = 1$ *then goto iii.*

(c) Compute the arithmetic mean over all derivatives obtained in step ***ii(2)a***

$$\rho_\nu := \frac{1}{m_{\boldsymbol{q}}} \sum_{\mu=1}^{m_{\boldsymbol{q}}} \widetilde{\rho}_{\nu,\mu}\,.$$

(d) *If all derivatives* $\widetilde{\rho}_{\nu,\mu}$, $\mu = 1, \ldots, m_{\boldsymbol{q}}$, *are close enough to* ρ_ν *then assume that the special case is detected. Otherwise, terminate with an error:*
If $(\exists \mu \in \{1, \ldots, m_{\boldsymbol{q}}\} : (|\widetilde{\rho}_{\nu,\mu} - \rho_\nu| > RTOL \cdot |\rho_\nu| + ATOL))$ *then exit(*cannot compute sensitivities!*).*

(iii) *Determine the second order time derivative* $\ddot{\boldsymbol{\xi}}^- := [[\ddot{\boldsymbol{x}}^-]^T, [\ddot{\boldsymbol{y}}^-]^T]^T$ *by*

$$\left[\frac{\partial \boldsymbol{f}^-}{\partial \dot{\boldsymbol{x}}}\right] \ddot{\boldsymbol{x}}^- = -\left[\frac{\partial \boldsymbol{f}^-}{\partial t} + \frac{\partial \boldsymbol{f}^-}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^- + \frac{\partial \boldsymbol{f}^-}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}}^-\right],$$
$$\ddot{\boldsymbol{y}}^- := 0 \quad \textit{(see the discussion in Section 4.3.2.b below).}$$

(iv) *Switch the model equations, i.e., perform the transition from* t_{disc}^- *to* t_{disc}^+.

(v) *Approximate the total time derivative of the jump function (respectively, of the back-tracing functions). Additionally, compute numerically consistent initial values by back-tracing as well as the corresponding sensitivities of the initialisation problem.*

(1) *Determine the disturbances for the finite difference approximation*

$$\epsilon_t := \Delta t_{min},$$
$$\epsilon_{\boldsymbol{\xi}} := \begin{cases} \sqrt{\epsilon_{mach}} \cdot \left(1 + \left\|\boldsymbol{\xi}^-\right\|\right) / \left\|\dot{\boldsymbol{\xi}}^-\right\|; & \left\|\dot{\boldsymbol{\xi}}^-\right\| > 0, \\ 0; & \textit{otherwise}, \end{cases}$$
$$\epsilon_{\dot{\boldsymbol{\xi}}} := \begin{cases} \sqrt{\epsilon_{mach}} \cdot \left(1 + \left\|\dot{\boldsymbol{\xi}}^-\right\|\right) / \left\|\ddot{\boldsymbol{\xi}}^-\right\|; & \left\|\ddot{\boldsymbol{\xi}}^-\right\| > 0, \\ 0; & \textit{otherwise}. \end{cases}$$

(2) *Compute the back-tracing function* $\boldsymbol{\chi} := [{\boldsymbol{\chi}_0}^T, {\boldsymbol{\chi}_1}^T]^T$ *starting from undisturbed as well as from disturbed initial values*

(a) *undisturbed:* $\boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p})$,
(b) *disturbed in t:* $\boldsymbol{\chi}(t_{disc} + \epsilon_t, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p})$,
(c) *disturbed in* $\boldsymbol{\xi}^-$: $\boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^- + \epsilon_{\boldsymbol{\xi}} \cdot \dot{\boldsymbol{\xi}}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p})$,
(d) *disturbed in* $\dot{\boldsymbol{\xi}}^-$: $\boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^- + \epsilon_{\dot{\boldsymbol{\xi}}} \cdot \ddot{\boldsymbol{\xi}}^-, \boldsymbol{p})$.

In order to obtain the sensitivities of the initialisation problem activate the integration of the sensitivity equations during step ***v(2)a*** *and set*

$$\begin{bmatrix} \widetilde{\boldsymbol{\omega}} \\ \widetilde{\dot{\boldsymbol{\omega}}} \end{bmatrix} := \begin{bmatrix} \frac{\partial \boldsymbol{\chi}_0}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{\chi}_1}{\partial \boldsymbol{p}} \end{bmatrix}_{\substack{t_{disc}, \boldsymbol{p} \\ \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-}} = \left[\frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{p}}\right]_{\substack{t_{disc}, \boldsymbol{p} \\ \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-}} .$$

(3) Apply finite difference approximations in order to approximate the directional derivatives of the back-tracing function

$$\left[\frac{\partial \boldsymbol{\chi}}{\partial t}\right] \doteq \left[\frac{\boldsymbol{\chi}(t_{disc}+\epsilon_t, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p}) - \boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p})}{\epsilon_t}\right],$$

$$\left[\frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{\xi}^-}\dot{\boldsymbol{\xi}}^-\right] \doteq \left[\frac{\boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^- + \epsilon_{\boldsymbol{\xi}}\cdot\dot{\boldsymbol{\xi}}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p}) - \boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p})}{\epsilon_{\boldsymbol{\xi}}}\right],$$

$$\left[\frac{\partial \boldsymbol{\chi}}{\partial \dot{\boldsymbol{\xi}}^-}\ddot{\boldsymbol{\xi}}^-\right] \doteq \left[\frac{\boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^- + \epsilon_{\dot{\boldsymbol{\xi}}}\cdot\ddot{\boldsymbol{\xi}}^-, \boldsymbol{p}) - \boldsymbol{\chi}(t_{disc}, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p})}{\epsilon_{\dot{\boldsymbol{\xi}}}}\right].$$

(4) Calculate the total time derivative of the back-tracing functions

$$\begin{bmatrix} \frac{d\boldsymbol{\chi}_0}{dt} \\ \frac{d\boldsymbol{\chi}_1}{dt} \end{bmatrix} = \left[\frac{d\boldsymbol{\chi}}{dt}\right] = \left[\frac{\partial \boldsymbol{\chi}}{\partial t} + \frac{\partial \boldsymbol{\chi}}{\partial \boldsymbol{\xi}^-}\dot{\boldsymbol{\xi}}^- + \frac{\partial \boldsymbol{\chi}}{\partial \dot{\boldsymbol{\xi}}^-}\ddot{\boldsymbol{\xi}}^-\right].$$

(vi) Determine the second order time derivative $\ddot{\boldsymbol{\xi}}^+ := [(\ddot{\boldsymbol{x}}^+)^T, (\ddot{\boldsymbol{y}}^+)^T]^T$ *by*

$$\left[\frac{\partial \boldsymbol{f}^+}{\partial \dot{\boldsymbol{x}}}\right]\ddot{\boldsymbol{x}}^+ = -\left[\frac{\partial \boldsymbol{f}^+}{\partial t} + \frac{\partial \boldsymbol{f}^+}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^+ + \frac{\partial \boldsymbol{f}^+}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}}^+\right],$$

$$\ddot{\boldsymbol{y}}^+ := 0 \quad \textit{(see the discussion in Section 4.3.2.b below)}.$$

(vii) For each parameter $\boldsymbol{p}_\nu$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$, *calculate the corresponding corrected column of the sensitivity matrices according to Eqs.* (4.37a)–(4.37b)

$$[\boldsymbol{\omega}^+]_\nu := -\left[\dot{\boldsymbol{\xi}}^+ - \frac{d\boldsymbol{\chi}_0}{dt}\right]\cdot\rho_\nu + [\widetilde{\boldsymbol{\omega}}]_\nu,$$

$$[\dot{\boldsymbol{\omega}}^+]_\nu := -\left[\ddot{\boldsymbol{\xi}}^+ - \frac{d\boldsymbol{\chi}_1}{dt}\right]\cdot\rho_\nu + \left[\widetilde{\dot{\boldsymbol{\omega}}}\right]_\nu.$$

Remark 4.13:
The additional computational effort required to handle the special case of more than one active real-valued switching function in steps **ii(2)c** and **ii(2)d** is small. On the other hand, the conditions for the activation of this feature are rather restrictive. However, we required the additional logic in the context of state dependent discontinuities for a model where during an earlier stage of development of the flowsheet two nearly identical parts of the plant were present (due to "copy and paste"). See also Remark 4.11 on page 158. ◊

4.3.2.b Discussion of the Method

A weak point in Algorithm 10 is the assignment of the second order derivatives $\ddot{\boldsymbol{y}}^-$ and $\ddot{\boldsymbol{y}}^+$ in steps **iii** and **vi**. A better estimate for $\ddot{\boldsymbol{y}}^-$ is available if the BDF-integrator had chosen at least order 2 for the integration step during which the

discontinuity occurs. Similarly, a more reliable value for $\ddot{\boldsymbol{y}}^+$ is available if at least order 2 is achieved in the last backward integration step made in order to compute the undisturbed back-tracing function $\boldsymbol{\chi}(t_n, \boldsymbol{\xi}^-, \dot{\boldsymbol{\xi}}^-, \boldsymbol{p})$ in **v(2)a**.

Remark 4.14:
Strictly speaking the latter is equivalent to the introduction of a third back-tracing function $\boldsymbol{\chi}_2$ which gives $\ddot{\boldsymbol{\xi}}^+(t_n; \boldsymbol{p}) := \boldsymbol{\chi}_2(t, \boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \boldsymbol{p})\Big|_{t_n^-, \boldsymbol{p}}$. $\boldsymbol{\chi}_2$ was not explicitly used in Algorithm 10 as its availability is strongly dependent on the actual integration history. Though it may be used as an *optional* feature in case of a sufficiently high local order of the integrator. ◊

The error in $\ddot{\boldsymbol{\xi}}^+$ can be neglected as it only affects $\partial \dot{\boldsymbol{y}}^+ / \partial \boldsymbol{p}$ in step **vii**. This has no effect on the further evolution of $\boldsymbol{\omega}(t; \boldsymbol{p})$ as $\partial \dot{\boldsymbol{y}}^+ / \partial \boldsymbol{p}$ is not required for the integration of the sensitivity equations. A deviation in $\ddot{\boldsymbol{\xi}}^-$ is more severe as $\ddot{\boldsymbol{\xi}}^-$ is used in order to approximate $[\partial \boldsymbol{\chi}_0 / \partial \dot{\boldsymbol{\xi}}^-] \ddot{\boldsymbol{\xi}}^-$ in step **v3** and thus enters $\boldsymbol{\omega}^+$ and $\dot{\boldsymbol{\omega}}^+$ in step **vii** via the approximation of $\mathrm{d}\boldsymbol{\chi}/\mathrm{d}t$ in step **v4**.

In our numerical tests the back-tracing based method for the computation of sensitivities after discontinuities has shown deficits in robustness, especially when applied to non-trivial problems from chemical engineering. In case of failure mainly the forward and backward integration phases of the disturbed problems **v(2)b** – **v(2)d** required for the numerical differentiation of the jump functions could not be performed. In some cases failure of the integrator already occurred in the evaluation of the undisturbed jump function, cf. the numerical tests of our algorithms for computation of consistent initial conditions summarised in Section 6.2. We see at least two possible explanations for this behaviour:

1. In the context of models for processes from chemical engineering disturbing the state variables can severely harm computations regarding physical properties based on underlying root-finding methods. Moreover, the disturbed trajectories may head into regions of the state space which have no physical significance. In such regions the physical models loose their validity, or they may even fail.

2. By backward integration error contributions may get dominant which are damped during forward integration.

4.3.3 Sensitivity Transfer in the Index-2 Consistent Initialisation Problem

In Section 3.2 we have developed an algorithm for the consistent initialisation of semi-explicit index-2 DAE systems. The point in time t of consistent initialisation was not specified in closer detail as consistent initialisation may be performed at any fixed but otherwise arbitrary time t within the integration horizon, given corresponding transition conditions.

We now assume that our algorithm introduced in Section 3.2 has been executed successfully, i.e., given the semi-explicit index-2 system Eqs. (4.31a)–(4.31b), and

the time of a discontinuity $t := t_n$ fixed by the switching function Eq. (4.32) reduced derivative array equations have been generated (cf. Eqs. (3.36a)–(3.36d))[iv]

$$0 = \tilde{\boldsymbol{f}}_n^+(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \dot{\tilde{\boldsymbol{x}}}, \boldsymbol{p}) , \tag{4.38a}$$

$$0 = \tilde{\boldsymbol{g}}_n^+(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}, \boldsymbol{p}) , \tag{4.38b}$$

$$0 = \tilde{\boldsymbol{s}}_n^+(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}, \boldsymbol{p}) , \tag{4.38c}$$

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}_n^+(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}, \boldsymbol{p}) \\ &= \frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial t}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}, \boldsymbol{p}) + \frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}, \boldsymbol{p}) \cdot \dot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}(t, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}, \boldsymbol{p}) \cdot \dot{\tilde{\boldsymbol{z}}} , \end{aligned} \tag{4.38d}$$

with $\tilde{\boldsymbol{f}}_n^+ : \mathbb{R}^{1+n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{y}}}+n_{\tilde{\boldsymbol{z}}}+n_{\tilde{\boldsymbol{x}}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{m_{\tilde{\boldsymbol{f}}}}$, $\tilde{\boldsymbol{g}}_n^+ : \mathbb{R}^{1+n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{y}}}+n_{\tilde{\boldsymbol{z}}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{m_{\tilde{\boldsymbol{g}}}}$, and $\tilde{\boldsymbol{s}}_n^+ : \mathbb{R}^{1+n_{\tilde{\boldsymbol{x}}}+n_{\tilde{\boldsymbol{z}}}+n_{\boldsymbol{p}}} \to \mathbb{R}^{m_{\tilde{\boldsymbol{s}}}}$, and the transition (or initial) conditions Eq. (3.57)

$$\boldsymbol{k}_\mu^{\mathrm{ini}}(\tilde{\boldsymbol{x}}_{K_\mu}^+(t_n), \boldsymbol{\xi}_{K^{\boldsymbol{\xi}}(\mu)}^-(t_n)) := \tilde{\boldsymbol{x}}_{K_\mu}^+(t_n) - \boldsymbol{\xi}_{K^{\boldsymbol{\xi}}(\mu)}^-(t_n) = 0 ; \, \mu = 1, \ldots, n_{\mathrm{ddf}}(n) . \tag{4.39}$$

The transition conditions will be denoted as $\tilde{\boldsymbol{h}}_n(\tilde{\boldsymbol{x}}^+, \boldsymbol{x}^-, \boldsymbol{y}^-)$, $\tilde{\boldsymbol{h}}_n : \mathbb{R}^{n_{\tilde{\boldsymbol{x}}}+n_{\boldsymbol{x}}+n_{\boldsymbol{y}}} \to \mathbb{R}^{n_{\mathrm{ddf}}(n)}$. The implicit notation for the original differential part of the DAE Eq. (4.38a) has only been chosen for convenience. Whenever appropriate, however, we exploit the special semi-explicit structure of our class of DAEs Eqs. (4.31a)–(4.31b) where Eq. (4.38a) is explicit in $\dot{\tilde{\boldsymbol{x}}}$.

The solution of the reduced system of consistency equations Eqs. (4.38a)–(4.38d), Eq. (4.39) gives the consistent initial conditions at t_n^+ for a fixed value of the vector of parameters $\boldsymbol{p}$:

$$\tilde{\Xi}^+(t_n(\boldsymbol{p}); \boldsymbol{p}) := \begin{bmatrix} \tilde{\boldsymbol{x}}^+(t_n(\boldsymbol{p}); \boldsymbol{p}) \\ \tilde{\boldsymbol{y}}^+(t_n(\boldsymbol{p}); \boldsymbol{p}) \\ \tilde{\boldsymbol{z}}^+(t_n(\boldsymbol{p}); \boldsymbol{p}) \\ \dot{\tilde{\boldsymbol{x}}}^+(t_n(\boldsymbol{p}); \boldsymbol{p}) \\ \dot{\tilde{\boldsymbol{z}}}^+(t_n(\boldsymbol{p}); \boldsymbol{p}) \end{bmatrix} . \tag{4.40}$$

4.3.3.a The Index-1 System and Higher Order State Derivatives

According to the information obtained from Pantelides' Algorithm an index-1 system can be generated from the index-2 DAE Eqs. (4.31a)–(4.31b) by replacing the equations $\tilde{\boldsymbol{s}}^+$ with their total time derivatives $\mathrm{d}\tilde{\boldsymbol{s}}_n^+/\mathrm{d}t$, i.e., by dropping Eq. (4.38c)

[iv] We stick to the notation introduced in Section 3.2, in combination with the notational conveniences that have been used in the preceeding sections, e.g., $\tilde{\boldsymbol{f}}_n^+(\cdot) := \lim_{t \searrow t_n} \tilde{\boldsymbol{f}}_{n+1}(\cdot)$.

from the reduced derivative array equations Eqs. (4.38a)–(4.38d):

$$\begin{aligned} 0 &= \tilde{\boldsymbol{F}}_n^+(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\dot{\tilde{\boldsymbol{z}}},\boldsymbol{p}) \\ &:= \begin{bmatrix} \tilde{\boldsymbol{f}}_n^+(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}},\tilde{\boldsymbol{z}},\dot{\tilde{\boldsymbol{x}}},\boldsymbol{p}) \\ \tilde{\boldsymbol{g}}_n^+(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}},\tilde{\boldsymbol{z}},\boldsymbol{p}) \\ \frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial t}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\boldsymbol{p}) + \frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\boldsymbol{p})\cdot\dot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}(t,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{z}},\boldsymbol{p})\cdot\dot{\tilde{\boldsymbol{z}}} \end{bmatrix}. \end{aligned} \tag{4.41}$$

If a rank condition similar to Eq. (4.19) is valid at the consistent point $[t_n^+, \tilde{\Xi}^+, \boldsymbol{p}]$

$$\operatorname{rank}\left(\left[\begin{array}{c|ccc} \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \dot{\tilde{\boldsymbol{y}}}} & \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \dot{\tilde{\boldsymbol{x}}}} & \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \dot{\tilde{\boldsymbol{z}}}} \end{array} \right]_{t_n^+,\tilde{\Xi}^+,\boldsymbol{p}} \right) = (n_{\tilde{\boldsymbol{x}}} + n_{\tilde{\boldsymbol{z}}}) + n_{\tilde{\boldsymbol{y}}} = n_{\boldsymbol{x}} + n_{\boldsymbol{y}}\,,$$

the higher order time derivatives $\ddot{\tilde{\boldsymbol{x}}}^+$, $\ddot{\tilde{\boldsymbol{z}}}^+$, and $\dot{\tilde{\boldsymbol{y}}}^+$ can be obtained from (cf. Eq. (4.20))

$$\begin{aligned} 0 = \frac{\mathrm{d}\tilde{\boldsymbol{F}}_n^+}{\mathrm{d}t} &= \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \tilde{\boldsymbol{x}}}\dot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \tilde{\boldsymbol{y}}}\dot{\tilde{\boldsymbol{y}}} + \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \tilde{\boldsymbol{z}}}\dot{\tilde{\boldsymbol{z}}} + \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \dot{\tilde{\boldsymbol{x}}}}\ddot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \dot{\tilde{\boldsymbol{z}}}}\ddot{\tilde{\boldsymbol{z}}} + \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial t} \\ \Leftrightarrow \quad \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \tilde{\boldsymbol{y}}}\dot{\tilde{\boldsymbol{y}}} &+ \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \dot{\tilde{\boldsymbol{x}}}}\ddot{\tilde{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \dot{\tilde{\boldsymbol{z}}}}\ddot{\tilde{\boldsymbol{z}}} = -\frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \tilde{\boldsymbol{x}}}\dot{\tilde{\boldsymbol{x}}} - \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial \tilde{\boldsymbol{z}}}\dot{\tilde{\boldsymbol{z}}} - \frac{\partial \tilde{\boldsymbol{F}}_n^+}{\partial t}\,, \end{aligned} \tag{4.42}$$

or, in detail,

$$\begin{bmatrix} -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{y}}}\right] & \boldsymbol{0} \\ \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{y}}}\right] & \boldsymbol{0} \\ \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] & \boldsymbol{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] \end{bmatrix} \begin{bmatrix} \ddot{\tilde{\boldsymbol{x}}} \\ \dot{\tilde{\boldsymbol{y}}} \\ \ddot{\tilde{\boldsymbol{z}}} \end{bmatrix} = - \begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] \\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] \\ \left[\frac{\partial}{\partial \tilde{\boldsymbol{x}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right] & \left[\frac{\partial}{\partial \tilde{\boldsymbol{z}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right] \end{bmatrix} \begin{bmatrix} \dot{\tilde{\boldsymbol{x}}} \\ \dot{\tilde{\boldsymbol{z}}} \end{bmatrix} - \begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial t}\right] \\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial t}\right] \\ \left[\frac{\partial}{\partial t}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right] \end{bmatrix}. \end{bmatrix}$$

All Jacobians are evaluated at $[t_n^+, \tilde{\Xi}^+, \boldsymbol{p}]$, cf. Eq. (4.40).

4.3.3.b Parametric Sensitivity of the Time of Discontinuity

Additionally, the sensitivity of the switching time $t_n = t_n(\boldsymbol{p})$ with respect to the parameters $\boldsymbol{p}$ is required. It is obtained by total differentiation of the switching condition Eq. (4.32) with respect to the parameters:

$$\left[\frac{\partial \boldsymbol{q}_n^-}{\partial t} + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}}^-\right]\left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right] = -\left[\frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{x}}\boldsymbol{\rho}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{y}}\boldsymbol{\sigma}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{p}}\right],$$

under the transversality condition

$$\frac{\mathrm{d}\boldsymbol{q}_n^-}{\mathrm{d}t} = \left[\frac{\partial \boldsymbol{q}_n^-}{\partial t} + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}^- + \frac{\partial \boldsymbol{q}_n^-}{\partial \boldsymbol{y}}\dot{\boldsymbol{y}}^-\right] \neq 0\,.$$

Remark 4.15:

The parametric sensitivity of the switching time is based on the active switching function(s) evaluated *before* the discontinuity is executed. ◊

In our case the Jacobians of the switching functions are not available in explicit form. Therefore, the Jacobians are numerically approximated by finite differences as in steps **i**, **ii1**, and **ii2** of Algorithm 10 in Section 4.3.2.a.

4.3.3.c Sensitivity Transfer

The consistent initial conditions after the discontinuity are obtained by solving the reduced system of consistency equations Eqs. (4.38a)–(4.38d), Eq. (4.39). Thus the corresponding system describing the sensitivity transfer across the discontinuity can be obtained by total differentiation of Eqs. (4.38a)–(4.38d), Eq. (4.39) with respect to the parameters $\boldsymbol{p}$. With the Jacobian of the reduced system of consistency equations (cf. Eq. (3.37))

$$\boldsymbol{M} := \begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{y}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} & \mathbf{0} \\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{y}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] & \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] & \mathbf{0} & \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] & \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial}{\partial \tilde{\boldsymbol{x}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right] & \mathbf{0} & \left[\frac{\partial}{\partial \tilde{\boldsymbol{z}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] & \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] \\ \left[\frac{\partial \boldsymbol{h}_n}{\partial \tilde{\boldsymbol{x}}^+}\right] & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}_{t_n^+, \tilde{\Xi}^+, \boldsymbol{p}},$$

the sensitivity matrices $\tilde{\boldsymbol{\rho}}^+$, $\tilde{\boldsymbol{\sigma}}^+$, $\tilde{\boldsymbol{\tau}}^+$, $\dot{\tilde{\boldsymbol{\rho}}}^+$, and $\dot{\tilde{\boldsymbol{\tau}}}^+$ on the right hand side of the discontinuity are determined by the linear system of equations

$$\begin{aligned} \boldsymbol{M} \cdot \begin{bmatrix} \tilde{\boldsymbol{\rho}}^+ \\ \tilde{\boldsymbol{\sigma}}^+ \\ \tilde{\boldsymbol{\tau}}^+ \\ \dot{\tilde{\boldsymbol{\rho}}}^+ \\ \dot{\tilde{\boldsymbol{\tau}}}^+ \end{bmatrix} = &- \left\{ \boldsymbol{M} \cdot \begin{bmatrix} \dot{\tilde{\boldsymbol{x}}}^+ \\ \dot{\tilde{\boldsymbol{y}}}^+ \\ \dot{\tilde{\boldsymbol{z}}}^+ \\ \ddot{\tilde{\boldsymbol{x}}}^+ \\ \ddot{\tilde{\boldsymbol{z}}}^+ \end{bmatrix} + \begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial t}\right] \\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial t}\right] \\ \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial t}\right] \\ \left[\frac{\partial}{\partial t}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right] \\ \mathbf{0} \end{bmatrix}_{t_n^+, \tilde{\Xi}^+, \boldsymbol{p}} \right\} \cdot \left[\frac{\partial t_n}{\partial \boldsymbol{p}}\right] - \begin{bmatrix} \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \boldsymbol{p}}\right] \\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \boldsymbol{p}}\right] \\ \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \boldsymbol{p}}\right] \\ \left[\frac{\partial}{\partial \boldsymbol{p}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right] \\ \mathbf{0} \end{bmatrix}_{t_n^+, \tilde{\Xi}^+, \boldsymbol{p}} \\ &- \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^-}\right] & \left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^-}\right] \end{bmatrix}_{t_n^+, \tilde{\Xi}^+, \boldsymbol{p}} \cdot \begin{bmatrix} \dot{\boldsymbol{x}}^- \frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{p}} \\ \dot{\boldsymbol{y}}^- \frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{y}^-}{\partial \boldsymbol{p}} \end{bmatrix}. \end{aligned} \tag{4.43}$$

After reordering of the terms on the right hand side it can be seen that Eq. (4.43) exactly corresponds to Eq. (4.24) already encountered in the review of the results of [Feeh 98] in Section 4.1.4.

Eq. (4.43) can be further simplified. For this purpose we utilise that the expression within the curly braces $\{\cdot\}$ is by rows equivalent to

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{f}}_n^+(t,\tilde{\boldsymbol{x}}^+,\tilde{\boldsymbol{y}}^+,\tilde{\boldsymbol{z}}^+,\dot{\tilde{\boldsymbol{x}}}^+,\boldsymbol{p})\,, \tag{4.44a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{g}}_n^+(t,\tilde{\boldsymbol{x}}^+,\tilde{\boldsymbol{y}}^+,\tilde{\boldsymbol{z}}^+,\boldsymbol{p})\,, \tag{4.44b}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}_n^+(t,\tilde{\boldsymbol{x}}^+,\tilde{\boldsymbol{z}}^+,\boldsymbol{p})\,, \tag{4.44c}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\tilde{\boldsymbol{s}}_n^+(t,\tilde{\boldsymbol{x}}^+,\tilde{\boldsymbol{z}}^+,\boldsymbol{p})\,, \tag{4.44d}$$

$$\left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+}\right]\cdot\left[\dot{\boldsymbol{x}}^+\right]\,. \tag{4.44e}$$

Due to the consistency of the states (collected in $\tilde{\Xi}^+$) $\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{s}}_n^+$ (Eq. (4.44c)) is zero. Furthermore, $\tilde{\Xi}^+$ are a consistent state of the index-1 DAE Eq. (4.41) at t_n^+ for the fixed value of the vector of parameters $\boldsymbol{p}$. Thus application of the same argument as in Eq. (4.20) to Eq. (4.42) gives that the total time derivative $\mathrm{d}\tilde{\boldsymbol{F}}_n^+/\mathrm{d}t$ is zero, i.e., $\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{f}}_n^+$ (Eq. (4.44a)), $\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\boldsymbol{g}}_n^+$ (Eq. (4.44b)), and $\frac{\mathrm{d}^2}{\mathrm{d}t^2}\tilde{\boldsymbol{s}}_n^+$ (Eq. (4.44d)) vanish. Inserting these relations Eq. (4.43) can be reduced to

$$\boldsymbol{M}\cdot\begin{bmatrix}\tilde{\boldsymbol{\rho}}^+\\ \tilde{\boldsymbol{\sigma}}^+\\ \tilde{\boldsymbol{\tau}}^+\\ \dot{\tilde{\boldsymbol{\rho}}}^+\\ \dot{\tilde{\boldsymbol{\tau}}}^+\end{bmatrix} = -\begin{bmatrix}\left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \boldsymbol{p}}\right]\\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \boldsymbol{p}}\right]\\ \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \boldsymbol{p}}\right]\\ \left[\frac{\partial}{\partial \boldsymbol{p}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right]\\ \boldsymbol{0}\end{bmatrix} - \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \Delta\tilde{\boldsymbol{\omega}}_n\end{bmatrix}\,, \tag{4.45}$$

where $\Delta\tilde{\boldsymbol{\omega}} \in \mathbb{R}^{n_{\mathrm{ddf}}(n)\times n_{\boldsymbol{p}}}$ is defined as

$$\begin{aligned}\Delta\tilde{\boldsymbol{\omega}}_n := &\left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+}\right]\cdot\left[\dot{\boldsymbol{x}}^+\frac{\partial t_n}{\partial \boldsymbol{p}}\right]\\ &+\left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^-}\right]\cdot\left[\dot{\boldsymbol{x}}^-\frac{\partial t_n}{\partial \boldsymbol{p}}+\frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{p}}\right]+\left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^-}\right]\cdot\left[\dot{\boldsymbol{y}}^-\frac{\partial t_n}{\partial \boldsymbol{p}}+\frac{\partial \boldsymbol{y}^-}{\partial \boldsymbol{p}}\right]\,.\end{aligned} \tag{4.46}$$

Again, all partial derivatives are evaluated at $[t_n^+,\tilde{\Xi}^+,\boldsymbol{p}]$.

An advantage of the newly derived system Eq. (4.45) over Eq. (4.43) as proposed in [Feeh 98] (see Section 4.1.4) is that the higher order time derivatives of the state variables $\ddot{\tilde{\boldsymbol{x}}}$, $\ddot{\tilde{\boldsymbol{y}}}$, and $\ddot{\tilde{\boldsymbol{z}}}$ are no longer required. Especially, the solution of

the index-1 system Eq. (4.41) for these values is unnecessary when the transition of the sensitivities is performed according to Eq. (4.45).

4.3.3.d Consistency of Transferred Sensitivities

The sensitivity equations

$$0 = \left[\frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{x}}\right] \boldsymbol{\rho}(t;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{y}}\right] \boldsymbol{\sigma}(t;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{p}}\right] - \dot{\boldsymbol{\rho}}(t;\boldsymbol{p}) \,, \quad t \in [t_{n-1}, t_n], n = 1, 2, \ldots \,, \tag{4.47a}$$

$$0 = \left[\frac{\partial \boldsymbol{g}_n}{\partial \boldsymbol{x}}\right] \boldsymbol{\rho}(t;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{g}_n}{\partial \boldsymbol{y}}\right] \boldsymbol{\sigma}(t;\boldsymbol{p}) + \left[\frac{\partial \boldsymbol{g}_n}{\partial \boldsymbol{p}}\right] \,, \tag{4.47b}$$

of the DAE Eqs. (4.31a)–(4.31b) constitute a DAE again, see Section 1.4. This motivates the question for the consistency of the transferred sensitivities $\tilde{\boldsymbol{\rho}}^+$, $\tilde{\boldsymbol{\sigma}}^+$, $\tilde{\boldsymbol{\tau}}^+$, $\dot{\tilde{\boldsymbol{\rho}}}^+$, and $\dot{\tilde{\boldsymbol{\tau}}}^+$ with the sensitivity DAE after a discontinuity.

The Jacobian of the sensitivity DAE Eqs. (4.47a)–(4.47b) with respect to $\boldsymbol{\sigma}$ and $\dot{\boldsymbol{\rho}}$, i.e., with respect to the highest order time derivatives present

$$\begin{bmatrix} \left[\frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{y}}\right] & -\boldsymbol{Id}_{n_{\tilde{\boldsymbol{x}}}} \\ \left[\frac{\partial \boldsymbol{g}_n}{\partial \boldsymbol{y}}\right] & \boldsymbol{0} \end{bmatrix}$$

is identical to the corresponding matrix of the original DAE obtained by differentiation of Eqs. (4.31a)–(4.31b) with respect to $\boldsymbol{y}$ and $\dot{\boldsymbol{x}}$. Thus, the sensitivity DAE inherits the structural properties of the original DAE. I.e., Pantelides' Algorithm (Section 3.1.5) applied to the sensitivity DAE Eqs. (4.47a)–(4.47b) returns the same index, the same number of dynamic degrees of freedom, the same set of equations to be differentiated in order to achieve index reduction, and the corresponding partitioning of the algebraic sensitivities as when applied to the structural analysis of the DAE Eqs. (4.31a)–(4.31b). Additionally, the possible sets of sensitivities that can be assigned as dynamic degrees of freedom is equivalent to the sets of dynamic degrees of freedom for the state variables. Altogether, at $t = t_n^+$ the reduced derivative array equations of the sensitivity DAE Eqs. (4.47a)–(4.47b) associated to the DAE Eqs. (4.31a)–(4.31b) can be set up as

$$0 = \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] \tilde{\boldsymbol{\rho}}^+ + \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{y}}}\right] \tilde{\boldsymbol{\sigma}}^+ + \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] \tilde{\boldsymbol{\tau}}^+ + \left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \boldsymbol{p}}\right] - \dot{\tilde{\boldsymbol{\rho}}}^+ \,, \tag{4.48a}$$

$$0 = \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] \tilde{\boldsymbol{\rho}}^+ + \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{y}}}\right] \tilde{\boldsymbol{\sigma}}^+ + \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] \tilde{\boldsymbol{\tau}}^+ + \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \boldsymbol{p}}\right] \,, \tag{4.48b}$$

$$0 = \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right] \tilde{\boldsymbol{\rho}}^+ + \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right] \tilde{\boldsymbol{\tau}}^+ + \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \boldsymbol{p}}\right] \,, \tag{4.48c}$$

$$
\begin{aligned}
0 &= \frac{\mathrm{d}}{\mathrm{d}t}\left\{\left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right]\tilde{\boldsymbol{\rho}}^+ + \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right]\tilde{\boldsymbol{\tau}}^+ + \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \boldsymbol{p}}\right]\right\} \\
&= \left[\frac{\partial}{\partial \tilde{\boldsymbol{x}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right]\tilde{\boldsymbol{\rho}}^+ + \left[\frac{\partial}{\partial \tilde{\boldsymbol{z}}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right]\tilde{\boldsymbol{\tau}}^+ + \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{x}}}\right]\dot{\tilde{\boldsymbol{\rho}}}^+ + \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \tilde{\boldsymbol{z}}}\right]\dot{\tilde{\boldsymbol{\tau}}}^+ + \left[\frac{\partial}{\partial \boldsymbol{p}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right].
\end{aligned}
\tag{4.48d}
$$

Furthermore, the subset of the sensitivities $\tilde{\boldsymbol{\rho}}^+$ can be assigned as dynamic degrees of freedom which corresponds to the differential states that are assigned as dynamic degrees of freedom for the original DAE. As transition conditions we *choose*

$$
\begin{aligned}
0 &= \left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+}\right]\cdot\left[\dot{\boldsymbol{x}}^+\frac{\partial t_n}{\partial \boldsymbol{p}} + \tilde{\boldsymbol{\rho}}^+\right] \\
&\quad + \left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^-}\right]\cdot\left[\dot{\boldsymbol{x}}^-\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{x}^-}{\partial \boldsymbol{p}}\right] + \left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{y}^-}\right]\cdot\left[\dot{\boldsymbol{y}}^-\frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{y}^-}{\partial \boldsymbol{p}}\right] \\
&\quad \Leftrightarrow \quad \left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+}\right]\tilde{\boldsymbol{\rho}}^+ = -\Delta\tilde{\boldsymbol{\omega}}_n\,.
\end{aligned}
\tag{4.49}
$$

Remark 4.16:
The matrix $\left[\frac{\partial \boldsymbol{h}_n}{\partial \boldsymbol{x}^+}\right] \in \mathbb{R}^{n_{\mathrm{ddf}}(n)} \times n_{\boldsymbol{x}}$ has by construction of $\boldsymbol{h}_n$ full row rank $n_{\mathrm{ddf}}(n)$. I.e., $n_{\mathrm{ddf}}(n)$ components in each column of $\tilde{\boldsymbol{\rho}}^+$ are determined by Eq. (4.49). ◊

The reduced system of consistency equations for the sensitivities Eqs. (4.48a)–(4.48d), Eq. (4.49) can then be written as

$$
\boldsymbol{M}\cdot\begin{bmatrix}\tilde{\boldsymbol{\rho}}^+\\ \tilde{\boldsymbol{\sigma}}^+\\ \tilde{\boldsymbol{\tau}}^+\\ \dot{\tilde{\boldsymbol{\rho}}}^+\\ \dot{\tilde{\boldsymbol{\tau}}}^+\end{bmatrix} + \begin{bmatrix}\left[\frac{\partial \tilde{\boldsymbol{f}}_n^+}{\partial \boldsymbol{p}}\right]\\ \left[\frac{\partial \tilde{\boldsymbol{g}}_n^+}{\partial \boldsymbol{p}}\right]\\ \left[\frac{\partial \tilde{\boldsymbol{s}}_n^+}{\partial \boldsymbol{p}}\right]\\ \left[\frac{\partial}{\partial \boldsymbol{p}}\frac{\mathrm{d}\tilde{\boldsymbol{s}}_n^+}{\mathrm{d}t}\right]\\ \boldsymbol{0}\end{bmatrix}_{t_n^+,\tilde{\Xi}^+,\boldsymbol{p}} = -\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \boldsymbol{0}\\ \Delta\tilde{\boldsymbol{\omega}}_n\end{bmatrix}, \tag{4.50}
$$

which is the same as Eq. (4.45). Therefore we conclude that Eq. (4.45) describing the transfer of sensitivities at discontinuities enforces *consistency* in the sensitivities.

In consideration of this property the (relatively small) step from Eq. (4.43) to Eq. (4.45) may be important from a numerical point of view, as due to round-off errors arising from Leimkuhler's approximation to $\mathrm{d}\tilde{\boldsymbol{s}}^+/\mathrm{d}t$ perturbations are introduced to the right hand side of Eq. (4.43). Thus the numerical values for the sensitivities $\tilde{\boldsymbol{\rho}}^+$, $\tilde{\boldsymbol{\sigma}}^+$, $\tilde{\boldsymbol{\tau}}^+$, $\dot{\tilde{\boldsymbol{\rho}}}^+$, and $\dot{\tilde{\boldsymbol{\tau}}}^+$ obtained by the solution of Eq. (4.43) may not be consistent with the sensitivity equations for the underlying DAE model.

Remark 4.17:
A comparison of the transition conditions Eq. (4.49) with the results of [Roze 67] discussed in Section 4.1.1, especially in the reordered form Eq. (4.5), shows that Eq. (4.45) derived

for the DAE case is equivalent to the formulae valid in the ODE case. This result on the compatibility of the formulae for the sensitivity transfer in DAEs with the ODE case has also been obtained by [Feeh 98] (see Section 4.1.4).

In detail, the restriction of the transition conditions to continuity in a subset of the differential states Eq. (4.39) is mirrored in the transfer of the sensitivities of the differential states Eq. (4.49) equivalent to the formulae for ODE case under a continuity condition on the state variables. The main difference between ODE and DAE is that in the DAE case the remaining sensitivities are determined by the consistency conditions for the sensitivity DAE. ◊

Chapter 5

Software Engineering

There are two ways to write error-free programs.
Only the third one works.

FORTUNE cookie

In the previous chapters we have presented our investigations regarding model predictive control of large-scale systems, and consistent initialisation and transfer of sensitivities for discontinuous semi-explicit index-2 DAEs primarily originating from chemical engineering problems.

Based on these investigations we have implemented our fast update method for disturbance rejection of open-loop controls aiming at preserving feasibility of the system dynamics (cf. Section 2.6.2). Further, we have implemented the back-tracing method (cf. Section 3.1.10) and the hybrid technique (cf. Section 3.2) for consistent initialisation. Finally, we have implemented the algorithms for the correction of parametric sensitivities after discontinuities (cf. Section 4.3.2 and Section 4.3.3) corresponding to the two different methods for the computation of consistent initial conditions. All algorithms have been integrated in the in-house simulation and optimisation tool OPTISIM® [Burr 93] of the Linde AG, Linde Engineering Division.

In the following sections the implementation of the algorithms is discussed, starting from the direct single shooting algorithm for the computation of open-loop parameterised optimal controls as the root of the work presented in this treatise. The main algorithms are depicted in flowcharts. A shaded item in a flowchart is explained in greater detail by a subordinate flowchart. If a flowchart is directly derived from another flowchart modifications are indicated by boldly lined items.

5.1 Direct Single Shooting Algorithm

Figure 5.1 depicts the main components of our direct single shooting algorithm as described in Section 2.4.2. We have integrated most of these components into

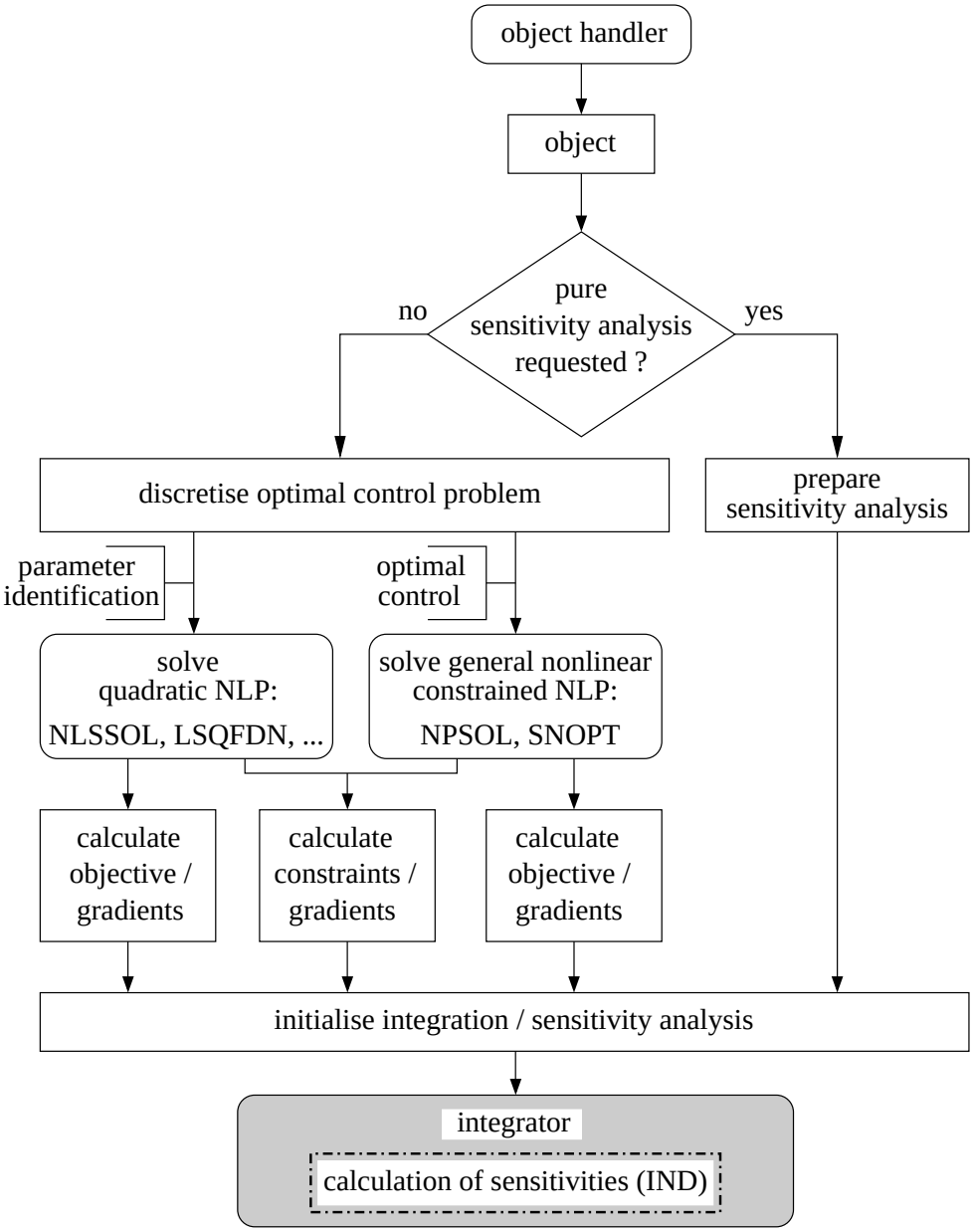

Figure 5.1: Flowchart for direct single shooting algorithm. For a flowchart of the integrator see Figure 5.2 (with sensitivity analysis) and Figure 5.3 (discontinuity treatment).

the simulation and optimisation tool OPTISIM® (cf. Section 2.2) within a previous project [Kron 98], [EKKS 99]. This thesis basically extends this direct single shooting algorithm with respect to the computation of closed-loop controls (see

Section 2.6), and with respect to the treatment of discontinuities in both simulation (by back-tracing, cf. Section 3.1.10, and solution of the reduced consistency equations, cf. Section 3.2) and parametric sensitivity analysis (cf. Section 4.3).

OPTISIM® is designed according to the object oriented programming paradigm [BMS 94]. Therefore, our algorithm starts when the object for the computation of parameterised optimal controls is invoked by the OPTISIM® object request broker.

If only parametric sensitivity analysis is requested then the integration of the sensitivity functions by the IND algorithm is started immediately. Otherwise, an optimisation problem is derived from the given optimal control problem by parameterisation of the controls and discretisation of the path inequality constraints, cf. Section 2.4.2. The user specified options and parameters for the optimisation algorithms are set, and the specified optimiser is started.

The optimisation algorithms require evaluation of the objective function and of the point constraints (i.e., of the sampled path constraints) for fixed values of the vector of optimisation parameters, as well as of the respective gradients[(i)]. In both cases the model DAE has to be integrated numerically. The gradients are obtained from the parametric sensitivity functions. These sensitivities are calculated simultaneously with the integration of the model DAE by IND, cf. the subsequent Section 5.2.

The basic implementation developed in [Kron 98] contained interfaces to the SQP algorithms NPSOL and NLSSOL. In the course of our recent work we have implemented interfaces to additional optimisation algorithms as listed in Table 5.1. The basic reason for extending the repertory of optimisers is that different algorithms work differently on the optimisation problems to be solved. Thus, on the one hand, if an algorithm fails for an optimisation problem the solution may be found by another one. On the other hand, [GMW 95] recommend to confirm the result from one optimisation algorithm by solving the same problem with a second one.

The optimisers can be classified into algorithms for the solution of general NLPs (NPSOL ([NAG 94b], [GMSW 98]), SNOPT ([GMS 97a], [GMS 97b]), E04CCF ([NAG 94a])), and algorithms designed for the solution of optimisation problems with a quadratic objective (LSQFDN ([GiMu 78], [NAG 94a]), LSQFDQ ([GiMu 78], [NAG 94a]), NLSSOL ([NAG 94c][(ii)]), VA07 ([AEA 93])). The latter type of optimisers is restricted to parameter estimation problems, while the former can be used for general optimal control problems. The second major characteristics is whether an optimisation algorithm is suitable for constrained optimisation,

[(i)] In contrast to all other optimisers mentioned below the simplex algorithm (E04CCF) does not utilise derivative information. However, due to performance issues it is intended for test purposes only.

[(ii)] With Mark 17 of the NAG® Fortran Library E04UPF has been superseded. E04UNF has been recommended as replacement.

Table 5.1: Optimisation algorithms accessible for the solution of a discretised optimal control problem.

routine	src.	alias	algorithm type	application
SNOPT	S	SNOPT	general SQP (sparse)	general constr. NLP
E04GBF	N	LSQFDQ	quasi Gauß-Newton	unconstr. least-squares
E04GDF	N	LSQFDN	modified Gauß-Newton	unconstr. least-squares
E04UCF	N	NPSOL	general SQP (dense)	general constr. NLP
E04UNF	N	NLSSOL	special SQP (dense)	constr. least-squares
(E04UPF)	N	NLSSOL		
E04CCF	N		simplex	general unconstr. NLP
VA07	H		modified Marquardt	unconstr. least-squares

source (src.): H: Harwell Subroutine Library, N: NAG® Fortran Library, S: Stanford Business Software Inc. SOL Optimization Software

or whether it is limited to unconstrained optimisation problems. We have found that the unavailability of a means to specify bounds at least for the optimisation variables by simple box constraints severely limits the applicability of an optimiser to our real-world parameter identification problems, cf. Section 6.3.2. Moreover, constraints constitute an important ingredient in our class of optimal control problems from industrial application, cf., e.g., Section 6.3.5.

SNOPT has proven exceptionally suitable for the solution of optimisation problems arising from the calculation of open-loop optimal load-changes for cryogenic air separation plants, see Section 2.1.1, Section 2.1.2, Section 6.3.5, and Section 6.3.6. Its key capabilities are an extended feasibility phase and robustness against failures in the evaluation of objective, constraints, and gradients. In our load-change problems the initial guess for the optimisation parameters is in general infeasible. Here the feasibility phase provides a powerful means for the detection of a set of optimisation variables such that the nonlinear constraints are not violated. Robustness of the optimiser against failures to provide requested data is the second important feature of SNOPT. This special robustness is required as due to the high nonlinearity and complex structure of the models success of an integration and of the computation of sensitivity functions cannot be guaranteed for all possible sets of optimisation variables even if the space of optimisation variables has been reasonably bounded a priori, e.g., according to physical reasons.

5.2 Internal Numerical Differentiation

In Section 2.4.3.b internal numerical differentiation has been discussed. As we have seen, IND constitutes an efficient and reliable technique for the computation

of sensitivity functions for parameter dependent DAE-IVPs. Additionally, within Section 2.4.3.b the relevant formulae for fixed leading-coefficient BDF integrators applied to semi-explicit index-2 DAEs have been derived.

In the staggered direct method the major effort for the integration of the sensitivity equations is in the solution of the linear corrector system Eq. (2.12). This linear system has to be solved directly after an integration step for the state variables has been finished. In case of large-scale DAE models with a sparse Jacobian the system matrix of these linear corrector systems is also large and sparse. If a direct linear solver is employed, e.g., MA48 ([AEA 93], [DuRe 96]), the numerically most expensive steps in the solution of a linear system of equations are the analysis of the structure of the system matrix and, based on this information, its LU-decomposition. The final forward and backward substitution steps are relatively cheap. From the integration of the state variable trajectories the LU-decomposition of the system matrix of the Newton system for the nonlinear corrector equations Eq. (2.11) is readily available. However, due to performance issues the Newton iteration is in general implemented as a quasi-Newton method with the system matrix fixed as long as possible. Thus the matrix available from the integration of the model DAE is in general not identical to the system matrix required for the integration of the sensitivity DAE, i.e., the LU-decomposition itself cannot be reused. But at least numerically expensive information about the formation of the LU-decomposition of the system matrix of the sensitivity DAE can be taken from above.

Figure 5.2 shows the major extensions of the BDF integrator in OPTISIM® required for the implementation of the staggered direct method. If the sensitivities are to be integrated without truncation error control the modifications in the original BDF code can be reduced to a small number of calls to auxiliary subroutines. The flow of information is then unidirectional from the main BDF integrator to the subroutines regarding the integration of the sensitivities.

In case of truncation error control for the sensitivity functions the integration of the sensitivity equations may influence the integration of the state variable trajectories. Especially for the case that the truncation error in the sensitivities exceeds the given threshold a major modification in the BDF code is necessary, as then repetition of the *entire* integration step with a reduced step-size is required.

5.3 Computation of Consistent Initial Conditions

Though interesting as a problem in itself, the computation of consistent initial conditions in practice is only a subordinate task within the numerical integration of a DAE-IVP. Consistent initial conditions are required either at the start of an integration or at a discontinuity during integration. Figure 5.3 (which has been derived from Figure 1.1 in Section 1.3.2 depicting the flowchart of an integrator

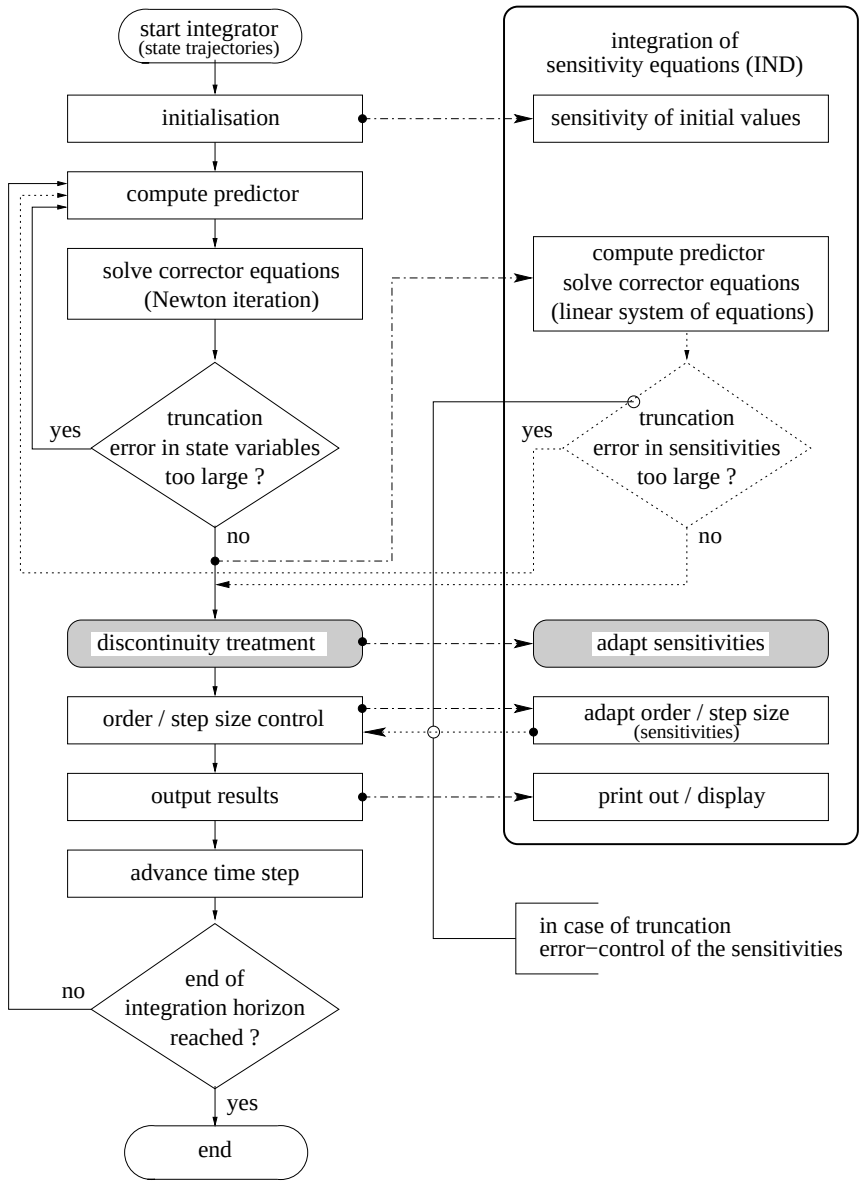

Figure 5.2: Computation of parametric sensitivity functions by IND of the BDF integrator in OPTISIM®, with optional truncation error control for the sensitivity equations. For the handling of discontinuities see Figure 5.3.

with discontinuity detection by discontinuity locking) shows the corresponding extensions in the integrator.

The computation of consistent initial conditions proceeds as shown in Figure 5.4.

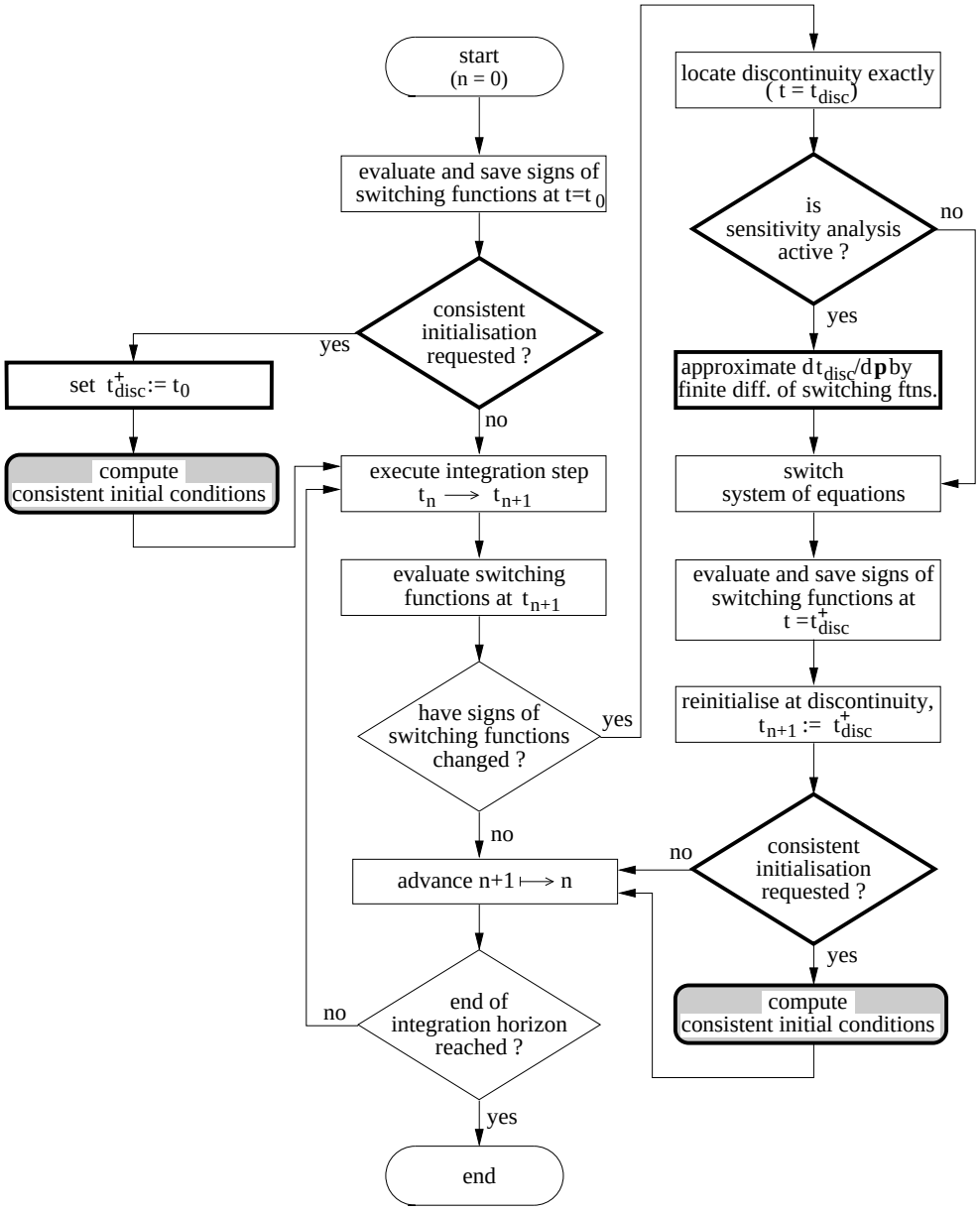

Figure 5.3: Extending an integrator with discontinuity detection by an algorithm for the computation of consistent initial conditions, cf. Figure 5.4. The gradient $\mathrm{d}t_{\mathrm{disc}}/\mathrm{d}\boldsymbol{p}$ is required for the transfer of sensitivity matrices at discontinuities, cf. Section 5.4.

According to the choice of the user back-tracing as described in Section 3.1.10 and Section 4.3.2 may be used in order to obtain numerically consistent initial conditions. At first the DAE is integrated forward in time for a fixed number of

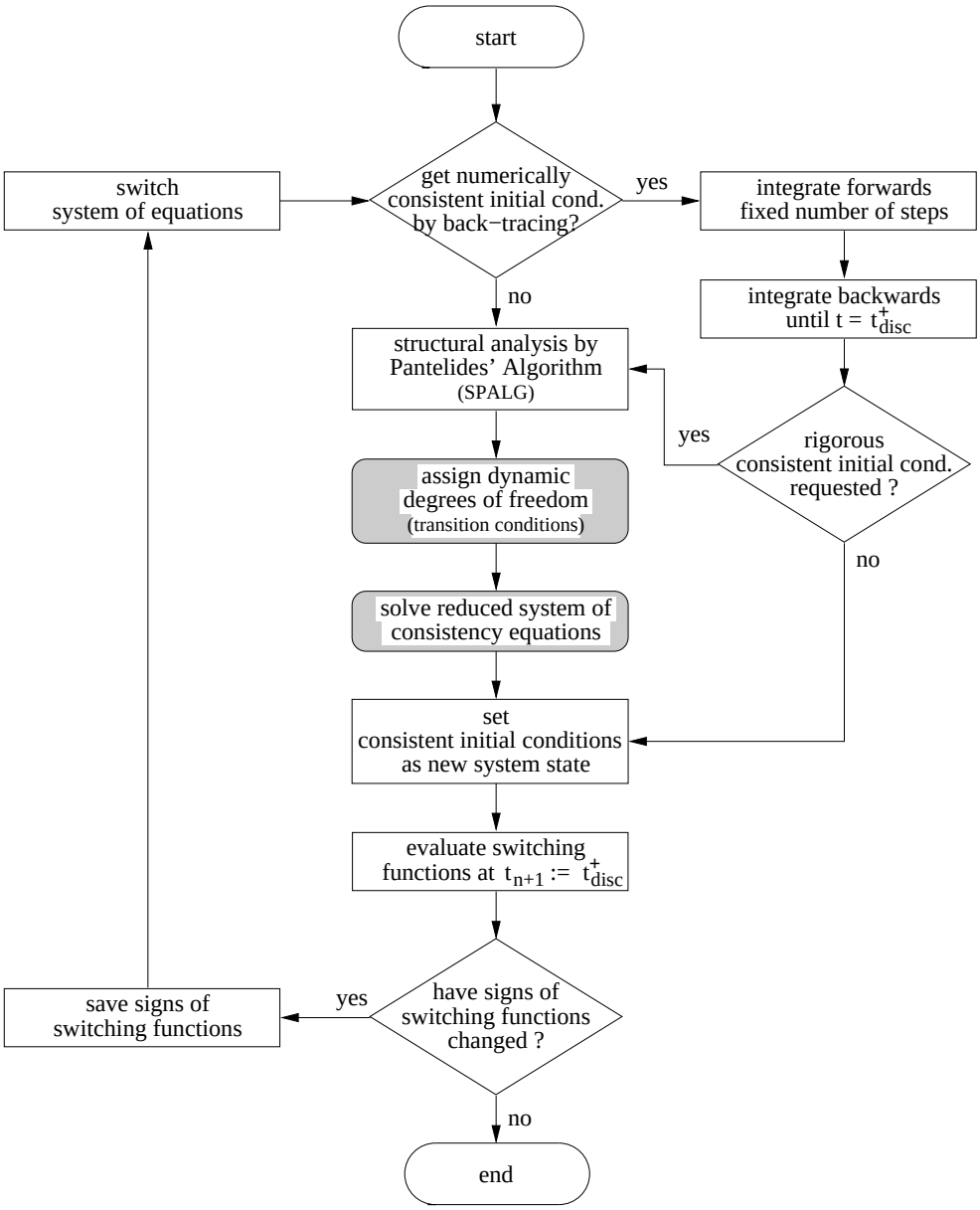

Figure 5.4: Computation of consistent initial conditions. The algorithm for the automatic choice of the transition conditions is sketched in Figure 5.5. Forward and backwards integration during back-tracing amount to a call of the (accordingly extended) integrator.

steps (based on our experience we use 4 integration steps). Then the direction of integration is reversed, and the DAE is integrated backwards in time until the starting point of the back-tracing task is reached. Algorithmically, several

low-level modifications of the BDF code in OPTISIM® have been necessary for implementation of this technique.

If consistent initial conditions are to be computed by the solution of the reduced consistency equations (Section 3.2) – optionally with the result of a back-tracing run as improved initial guess – the DAE is analysed by Pantelides' Algorithm in its implementation SPALG [UKM 95], cf. Section 3.2.1. The data obtained from SPALG provides the instructions for setting up the reduced derivative array equations. In the next step continuity conditions for a suitable set of differential variables are determined automatically by the method developed in Section 3.2.4, i.e., an adequate set of the state variables is chosen as dynamic degrees of freedom (see Figure 5.5). These transition conditions together with the reduced derivative array equations constitute the reduced system of consistency equations. In our case, the reduced system of consistency equations is in general nonlinear and large, and owns a sparse Jacobian without special structure. For the solution of this nonlinear system of equations different solvers can be employed in sequence, cf. Section 3.2.5 and Figure 5.6. A Levenberg-Marquardt algorithm (NS13, [AEA 93]), and the SQP method SNOPT ([GMS 97a], [GMS 97b]) are offered as pre-solvers with extended global convergence properties in order to obtain an improved start estimation for the more efficient main solvers. Primarily, however, Powell's dog-leg method (NS02, [AEA 93]), and preferably the affine-invariant Newton method NLEQ1S ([NoWe 91]) are employed to obtain the final value of the consistent initial conditions.

Return to Figure 5.4. After consistent initial conditions have been determined the switching functions have to be evaluated at the new system state. If a switching function has changed its sign the switching manifold has been passed. In this case the model equations have to be switched accordingly, and the corresponding set of consistent initial conditions has to be computed again (care has to be taken in order to avoid dead-locks).

In Section 6.2 our algorithms for consistent initialisation of semi-explicit index-2 DAEs are tested on a small, a medium, and a large-scale example.

5.4 Transfer of Sensitivities at Discontinuities

In Section 4.3.2 and Section 4.3.3 algorithms for the transfer of sensitivity matrices at discontinuities have been developed. These two algorithms are related to the algorithm for the computation of consistent initial conditions by back-tracing (cf. Section 3.1.10), and to the algorithm for the computation of consistent initial conditions by solution of the reduced consistency equations (cf. Section 3.2), respectively. The coupling of the methods for consistent initialisation to the corresponding method for the transfer of the sensitivity matrices is mirrored in the implementation. As depicted in Figure 5.7 the transfer of the sensitivity matrices directly extends the routine for the computation of consistent initial conditions.

In case of the back-tracing based approach derivatives of the jump function are

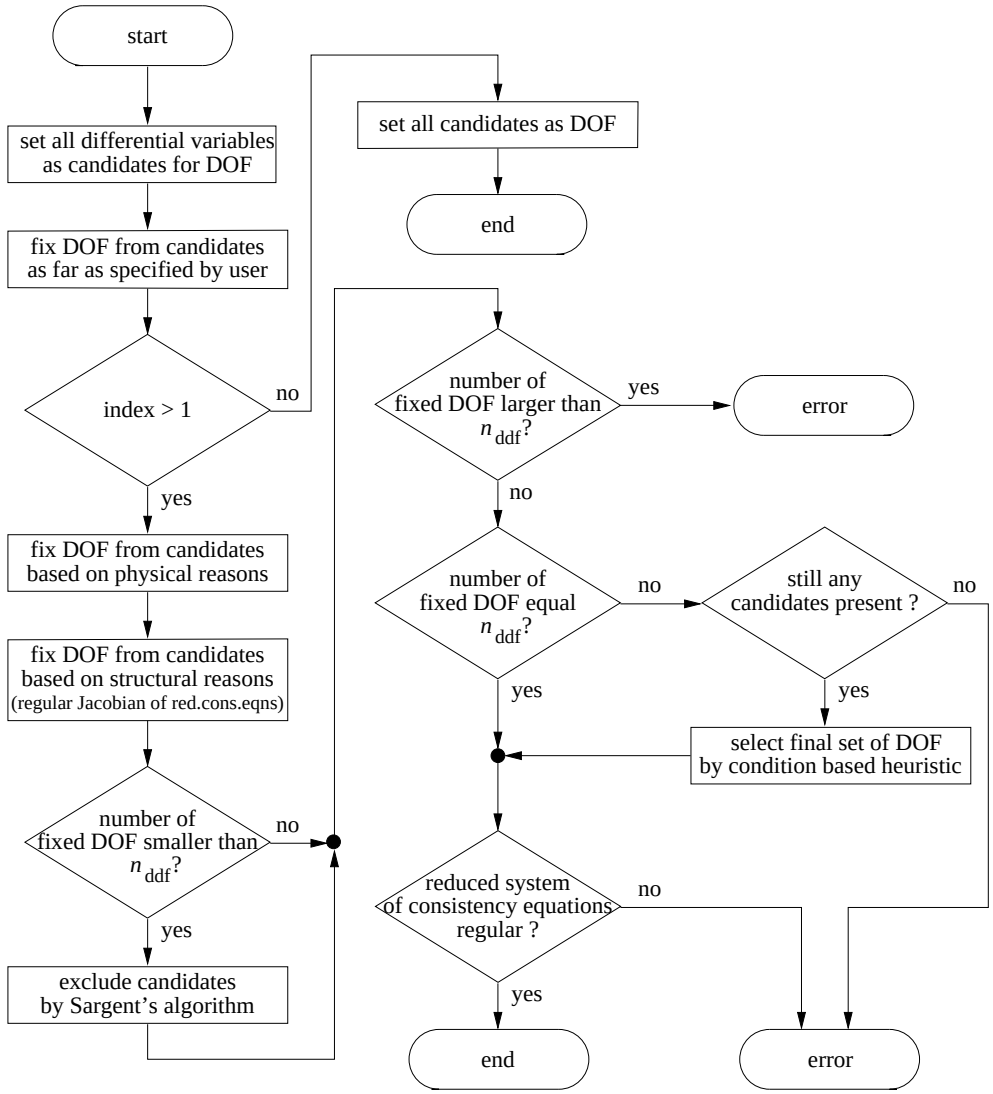

Figure 5.5: Automatic generation of a suitable set of transition conditions by Algorithm 8 developed in Section 3.2.4. n_{ddf} is the number of dynamic degrees of freedom. Note that once a candidate is *fixed* as DOF it is removed from the set of candidates.

numerically approximated by finite differences, cf. Section 4.3.2. Thus the BDF integrator has been extended for back-tracing starting from disturbed initial values. Additionally, as shown in [Kieh 99], during the disturbed back-tracing phases the same step-size and order history has to be employed as during the primary, undisturbed back-tracing phase in order to obtain reliable results. Therefore, during undisturbed back-tracing the adaptively determined step-size and order sequence is recorded. Then during disturbed back-tracing the order and step-size selection

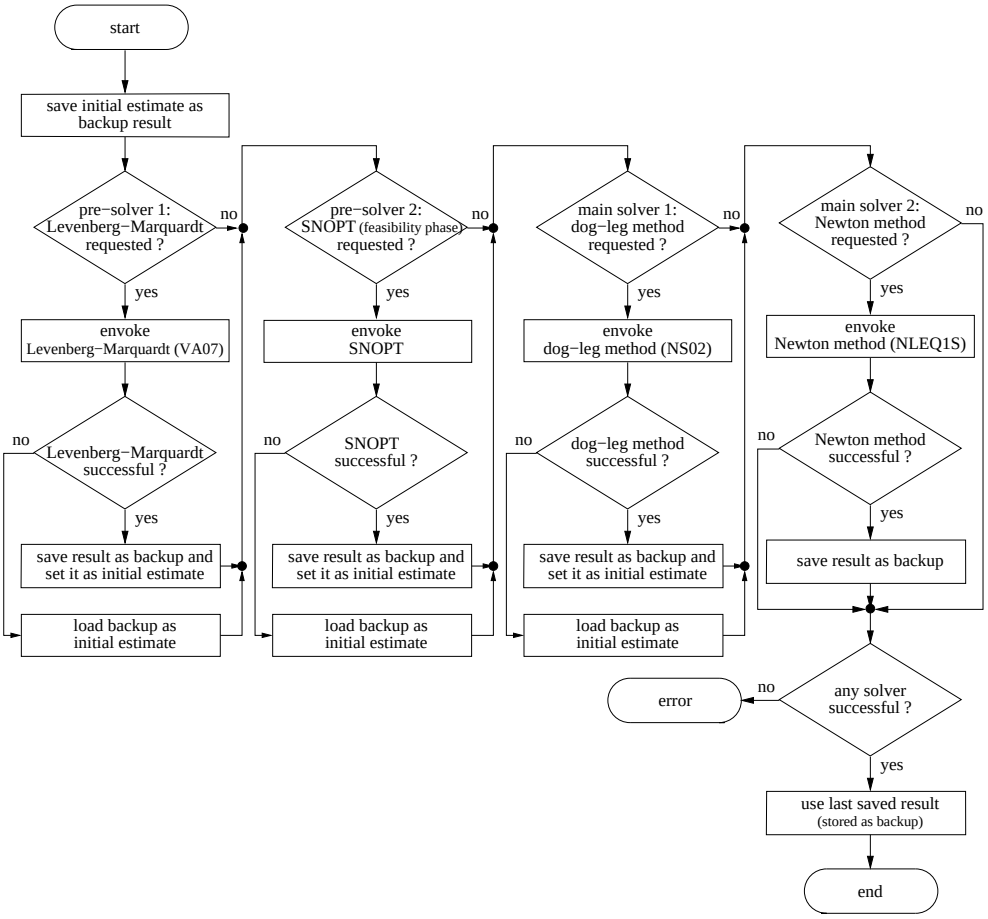

Figure 5.6: Solution of the reduced system of consistency equations by a cascade of algorithms for the solution of large, nonlinear systems of equations with sparse Jacobian, cf. Section 3.2.5.

strategy of the BDF integrator is temporarily overridden by the recorded data. In order to increase the robustness of the method the algorithm is restarted with the undisturbed back-tracing phase in case of a failure during a disturbed back-tracing phase, employing a reduced step-size for the initial integration step. Note that during the undisturbed back-tracing phase sensitivity analysis is active, while it is deactivated during disturbed back-tracing.

If consistent initial conditions are to be obtained by the solution of the reduced consistency equations the transfer of the sensitivity matrices basically reduces to the solution of the linear systems of equations Eq. (4.45) (or Eq. (4.43)). If back-tracing is employed in order to find an improved initial estimate for the solution of the reduced consistency equations sensitivity analysis has to be deactivated during this (undisturbed) back-tracing phase.

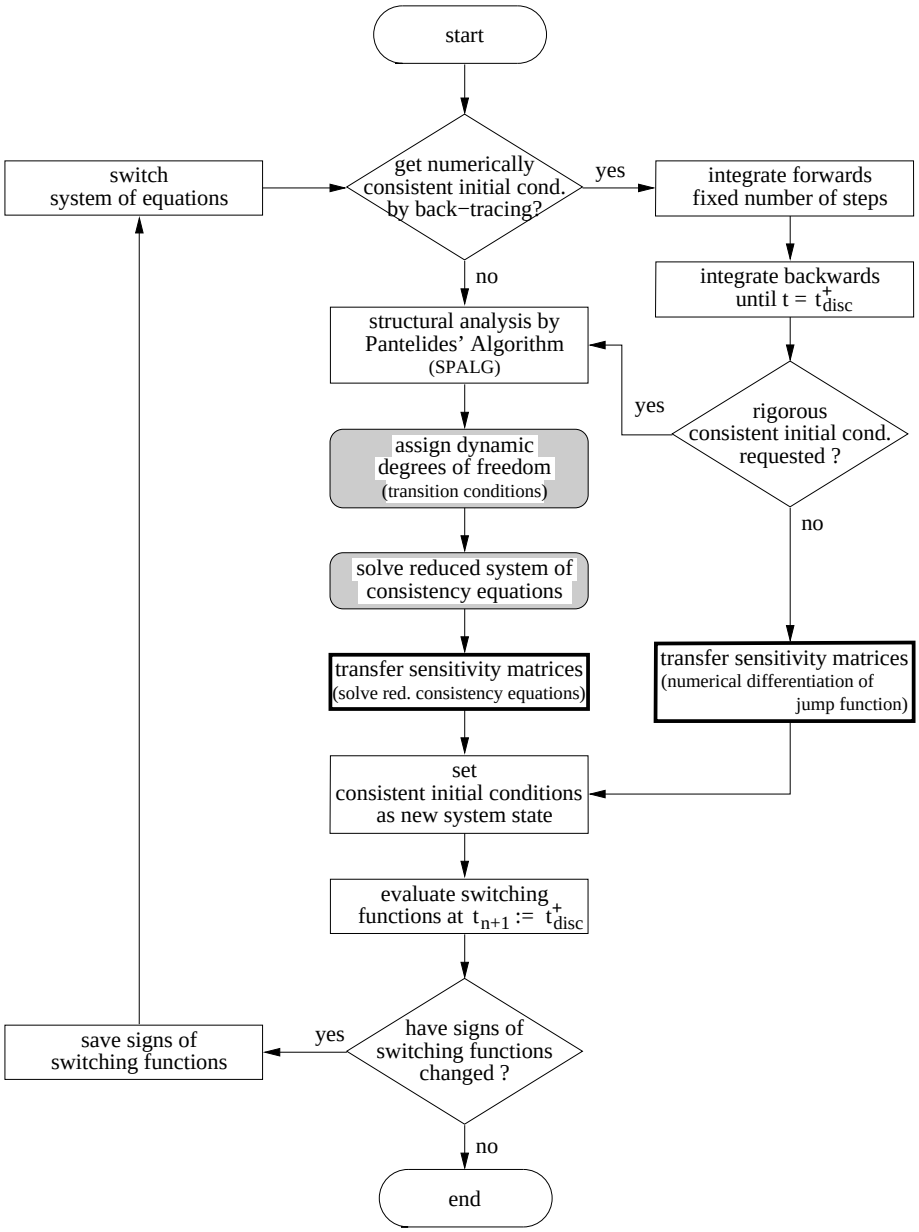

Figure 5.7: Transfer of sensitivities at discontinuities: Extension of the routine for the computation of consistent initial conditions, cf. Figure 5.3.

Each of the two algorithms requires the gradient $\mathrm{d}t_{\mathrm{disc}}/\mathrm{d}\boldsymbol{p}$ (i.e., first order information about the dependency of the location of the discontinuity on the parameters of interest). This data is approximated by numerical differentiation of the switching functions directly after location of the discontinuity, cf., e.g., step **i** in Algorithm 10. Thus it is located in the routine responsible for the detection

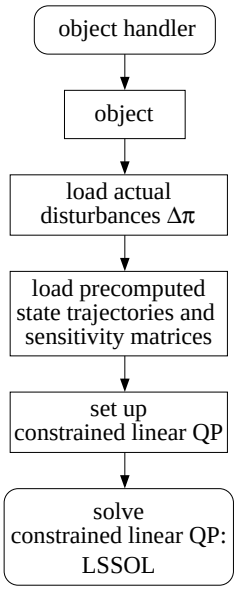

Figure 5.8: Fast disturbance rejection: Obtain updates to the control parameters by solution of a constrained linear QP.

of the discontinuities and for the basic management of consistent initialisation (cf. Figure 5.3).

5.5 Fast Disturbance Rejection

In Section 2.6.2 a method for the fast rejection of disturbances present during the operation of larger processes has been proposed. The main idea is to predict the evolution of the constrained states in the presence of disturbance parameters employing sensitivity information. This sensitivity information may have been computed offline, or it may be available from a preceding solution of an open-loop optimal control problem within an NMPC control strategy, cf., e.g., the optimal control concept discussed in Section 2.3.3. Based on this prediction updates to the optimisation parameters of the master optimal control problem are calculated by solving the constrained linear quadratic programming problem (QP) Eqs. (2.25a)–(2.25b). The updates are determined such that a predicted potential violation of the path inequality constraints of the master optimal control problem is avoided. If the quadratic objective Eq. (2.26) is used instead of Eq. (2.25a) the updates for the optimisation parameters are chosen such that additionally the deviation of the predicted state variable trajectories from a reference trajectory is minimised.

The computation of the updates is started from the object handling layer in OPTISIM®, cf. Figure 5.8. The next steps basically consist in data handling, i.e., actual values of the disturbance parameters, precomputed sensitivity matri-

ces, as well as trajectory data are loaded. Based on this data either the linearly constrained QP Eqs. (2.25a)–(2.25b), or the linearly constrained QP Eq. (2.26) and Eq. (2.25b) is set up. The QP is then solved by the algorithm LSSOL [GHMSW 86] (NAG® routine E04NCF [NAG 94a]). In contrast to the solution of the NLPs (which are at the core of our direct single shooting method discussed in Section 5.1), special routines for the evaluation of the objective function and of the constraints are not required. In the case of constrained linear QPs these operations reduce to matrix-vector multiplications. These multiplications are handled internally within LSSOL.

Chapter 6

Numerical Results

6.1 Introductory Remarks

The problem with engineers is that they tend to cheat in order to get results.
The problem with mathematicians is that they tend to work on toy problems in order to get results.
The problem with program verifiers is that they tend to cheat at toy problems in order to get results.

FORTUNE cookie

In Section 6.2 and Section 6.2.3 we present a series of example problems as an assessment of the capabilities of our algorithms for consistent initialisation (cf. Section 3.1.10, Section 3.2, and Section 5.3) and correct treatment of sensitivities at discontinuities (cf. Section 4.3.2, Section 4.3.3, and Section 5.4). Our algorithm for fast disturbance rejection proposed in Section 2.6.2 could not be sufficiently tested during this project, and will be part of further work.

In the course of the project there has been a perpetual interaction between theory and numerical examples. Theoretical considerations gave rise to modifications and/or extensions of the existing code, while the necessity of more refined methods arose from numerical tests. At this point we emphasise the importance of tests with *real world* problems from industry. Not very surprisingly we have made the experience that an algorithm that would work well for small to medium size academic test examples revealed weaknesses when applied to a real world problem. E.g., initially the back-tracing algorithm was suitable for the calculation of (numerically) consistent initial conditions in the pendulum example (cf. Section 6.2.1.b), while it would fail when applied to the C3-splitter example in Section 6.2.2. After some amendments it worked for this example but there were pertinent failures for the air separation plant examples (cf. Section 6.2.3, Section 6.3.5). At this point we decided to consider the solution of the reduced system of consistency equations as the practically more demanding but theoretically better founded approach.

As one of the main reasons for this phenomenon we see that mathematical models of real-world processes often suffer from limitations, e.g., due to their dependency on the convergence of lower-level iterative processes, unexpected singularities, hard (physically given) bounds on variable values, or extrapolation required in order to broaden their numerical applicability. These limitations can cause practical problems which are in common unknown to academic examples, e.g., if numerical differentiation is applied, or if an iterative algorithm drives the system state to the border of the domain of a model.

Next, in the context of complex models numerical methods have to take into account that functions or derivatives may only be available in limited precision. In contrast, academic examples seldom suffer from problems with low precision due to their simplicity.

Finally, by their mere size the large-scale examples set high demands on the numerical software used, mainly onto the linear algebra of large, sparse, and unstructured systems. In the solution of large nonlinear systems we additionally had to deal with the problem of transformation to well-scaled problems, and with the choice of adequate norms and global parameters.

Remark 6.1:
We have found that especially our rigorous algorithm for consistent initialisation is extremely sensitive to coding errors in both the model equations and in their Jacobian. Due to this property in some cases even subtle errors in complex unit models have been detected that had not caught any attention in the past. Other standard numerical methods within OPTISIM®, e.g., steady-state solver or integrator simply had not been affected adversely (which accounts to some extent for their robustness), or the detrimental effects had been too intricate than would have allowed to assign them to a special portion of code. ◊

Therefore we examine our algorithms on both, small academic examples and large-scale, industrial real-world problems as far as possible.

6.2 Consistent Initialisation

Grabel's Law:
2 is not equal to 3
– not even for large values of 2.

FORTUNE cookie

In the sequel mainly results for our algorithm for consistent initialisation discussed in Section 3.2 are reported. In Section 6.2.2 and Section 6.2.3 the reliability of our algorithm for consistent initialisation based on the solution of the reduced consistency equations is tested by disturbing the initial estimate for the nonlinear solvers with the relative factor *disturb*. Additionally, a fixed bias of $\sqrt{\epsilon_{\text{mach}}} \approx 1 \cdot 10^{-8}$ is added in order to disturb small values (especially we intended

to avoid artefacts in the tests caused by zeros)[(i)]. The *residual* measures the Euclidian norm of the final residual of the reduced consistency equations, Δ_{rel} denotes a measure for the relative difference between result and reference. We use

$$\Delta_{\text{rel}} := \frac{1}{n_{\boldsymbol{\xi}}} \sum_{\nu=1}^{n_{\boldsymbol{\xi}}} \frac{\left|\boldsymbol{\xi}_\nu - \boldsymbol{\xi}_\nu^{\text{ref}}\right|}{1 + \left|\boldsymbol{\xi}_\nu^{\text{ref}}\right|}, \tag{6.1}$$

where $\boldsymbol{\xi} := [\tilde{\boldsymbol{x}}^T, \tilde{\boldsymbol{y}}^T, \tilde{\boldsymbol{z}}^T, \dot{\tilde{\boldsymbol{x}}}^T, \dot{\tilde{\boldsymbol{z}}}^T]^T \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$, $n_{\boldsymbol{\xi}} = 2n_{\tilde{\boldsymbol{x}}} + n_{\tilde{\boldsymbol{y}}} + 2n_{\tilde{\boldsymbol{z}}}$, is the result of a consistent initialisation run. $\boldsymbol{\xi}^{\text{ref}} \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$ is a given fixed reference value. The value given as *evaluations* counts the number of evaluations of the reduced consistency equations, separated into pure residual evaluations (*res.*), and combined Jacobian / residual evaluations (*Jac. & res.*). Finally, *time* is the computer time in seconds elapsed for the solution of the reduced consistency equations measured on a Pentium™ II/350 MHz PC with a WINDOWS NT® operating system. The program was compiled using Compaq™ Visual Fortran 6.6. During these tests run-time error checks have been enabled so that a speed-up of at least a factor 2 can be expected for fully optimised code.

In some cases the solvers returned an unphysical solution which is marked with the flag [p!] behind the residual value. In the example in Section 6.2.2 in some cases at least one switching function repeatedly changed its sign immediately after consistent initialisation. This effect is marked with [s!].

In Section 6.2.1 the back-tracing algorithm is briefly considered. In Section 6.2.2 and Section 6.2.3 some additional results are reported. The back-tracing algorithm will also be used in some of the examples regarding sensitivity analysis in Section 6.2.3.

6.2.1 The Planar Pendulum (Index-2)

One of *the* standard examples in DAE analysis is the two dimensional mathematical pendulum with fixed length (cf., e.g., [Heim 92]). As sketched in Figure 6.1 a mass point is located at the end of a weightless rod of fixed length, which itself is mounted on a frictionless pivot. After a disturbance from its stationary points (the upper-most and lower-most positions) the mass point oscillates around the centre, i.e., the pivot.

The equations of motion can be derived by the Euler-Lagrange formalism, see, e.g., [GLG 85], [Gold 63]. When Cartesian coordinates are used the result is a system of two second order ODEs coupled with an algebraic condition. This

[(i)] "It is remarkable how often the values $x = 0$ or $x = 1$ are used to test function evaluation procedures, and how often the special properties of these numbers make the test meaningless." ([GMW 95], p. 296)

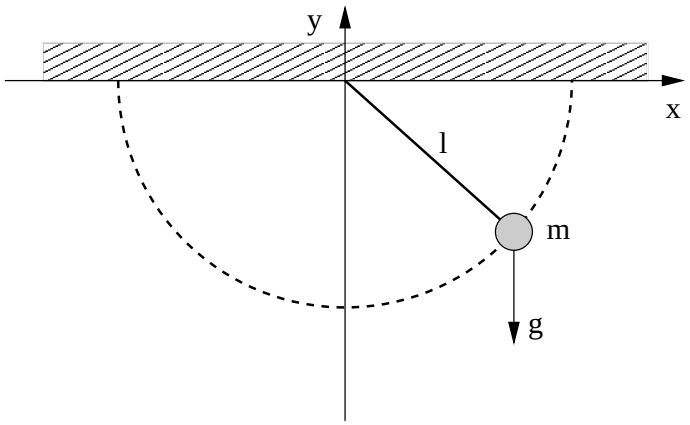

Figure 6.1: The two dimensional mathematical pendulum with fixed length l.

system is easily transformed into the first order DAE

$$\dot{x} = u\,, \tag{6.2a}$$
$$\dot{y} = v\,, \tag{6.2b}$$
$$\dot{u} = \frac{\lambda}{m} x\,, \tag{6.2c}$$
$$\dot{v} = \frac{\lambda}{m} y - g\,, \tag{6.2d}$$
$$0 = x^2 + y^2 - l^2\,, \tag{6.2e}$$

where x and y are the Cartesian coordinates of position, u and v are the respective velocities, m is the mass of the mass point, l is the length of the rod, and g is the constant of gravitational acceleration. λ is a Lagrangian multiplier which can be identified with the tractive force of the mass point exerted on the rod mounted on the pivot.

The DAE Eqs. (6.2a)–(6.2e) has a differential index of three. This is always the case for models derived from Euler-Lagrange equations with holonomic constraints[(ii)], c.f., e.g., [Rhei 84].

We restrict to the pendulum in index-2 formulation. It is obtained by replacing the algebraic constraint Eq. (6.2e) with its first total time derivative, see Eq. (6.3e)

[(ii)] *Holonomic constraints* are defined as equality constraints relating the coordinate variables of a system [Gold 63]. They may also explicitly depend on time. Eq. (6.2e) is such a holonomic constraint. It restricts the movement of the mass point to a circle with a radius of the length of the rod.

below. The index-2 DAE is then

$$\dot{x} = u\,, \tag{6.3a}$$
$$\dot{y} = v\,, \tag{6.3b}$$
$$\dot{u} = \frac{\lambda}{m} x\,, \tag{6.3c}$$
$$\dot{v} = \frac{\lambda}{m} y - g\,, \tag{6.3d}$$
$$0 = xu + yv\,. \tag{6.3e}$$

Remark 6.2:
In Section 6.3.3 the formalism is demonstrated at an extended pendulum example. ◊

The task in the sequel is to calculate consistent initial conditions for the index-2 pendulum at $t = 0$. In Table 6.1 the initial guess for the state variables is denoted. The estimate for $\lambda(0)$ is marked with the qualifier $\widehat{\cdot}$. As the index-2 pendulum contains the length constraint Eq. (6.2e) only in its differentiated form Eq. (6.3e) it needs to be *enforced* at least at the initial time [(iii)]. In our tests both the hybrid algorithm as well as the back-tracing algorithm preserve the suitably chosen initial values for $x(0)$ and $y(0)$ without additional measurements, see the results in Table 6.2 and Table 6.3 below.

Table 6.1: Initial guess and constants for pendulum example.

$x(0)$	0.4	[m]
$y(0)$	-0.3	[m]
$u(0)$	0.0	[m/s]
$v(0)$	0.0	[m/s]
$\widehat{\lambda}(0)$	0.0	[N/m]
m	0.3	[kg]
l	0.5	[m]
g	9.81	$[\mathrm{m/s^2}]$

6.2.1.a Solution of the Reduced Consistency Equations

Pantelides' Algorithm (or SPALG, respectively) correctly reports that the system Eqs. (6.3a)–(6.3e) is an index-2 DAE with 3 dynamic degrees of freedom. Without appropriate physical information regarding the choice of the dynamic degrees of freedom the continuity condition for one of the four differential variables has to be

[(iii)] In the index-2 DAE Eqs. (6.3a)–(6.3e) the length of the rod is a *constant of integration.*

Table 6.2: Consistent initial conditions for pendulum: Solution of the reduced consistency equations.

$x(0)$	0.4	[m]
$y(0)$	-0.3	[m]
$u(0)$	0.0	[m/s]
$v(0)$	0.0	[m/s]
$\lambda(0)$	-3.53160	[N/m]

dropped by our heuristic rule (step **4(c)ii** in Algorithm 8). Based on the condition estimate for the Jacobian of the reduced consistency equations at the initially guessed value of the state vector the algorithm selects v. In turn, x, y, and u are chosen as dynamic degrees of freedom. Thus these variables are fixed at their initially given values by the corresponding continuity conditions, and the length constraint is implicitly enforced. Note that in the hybrid algorithm x and y also could have been explicitly assigned as dynamic degrees of freedom by the user in order to guarantee that the initial position of the mass point and thus the length of the rod is fixed.

The affine-invariant Newton algorithm NLEQ1S requires 3 iterations for the solution of the reduced consistency equations. The results are given in Table 6.2. At the solution the residual of the reduced consistency equations is exactly zero in double precision arithmetics. $v(0)$ has to vanish due to Eq. (6.3e). $\lambda(0)$ is then easily confirmed by solving the index-1 condition for the pendulum

$$0 = u^2 + v^2 + \frac{l^2}{m}\lambda - yg\,,$$

which is obtained from total differentiation of Eq. (6.3e) with respect to time.

6.2.1.b Back-Tracing

The behaviour of the back-tracing method can be seen from Figure 6.2 which depicts the integration history for $\lambda(t)$. In order to obtain a suitable graphical output a loose integration tolerance of $1.0 \cdot 10^{-4}$ has been chosen.

After forward and backward integration at $t = 0$ the values noted in Table 6.3 are returned. Recall that in contrast to the hybrid algorithm the back-tracing based method does not explicitly contain continuity conditions. Though, the result of the back-tracing based method is similar to the result of the hybrid algorithm. With a more strict integration tolerance of $1.0 \cdot 10^{-6}$ the back-tracing algorithm even nearly reproduces the result of the rigorous consistent initialisation.

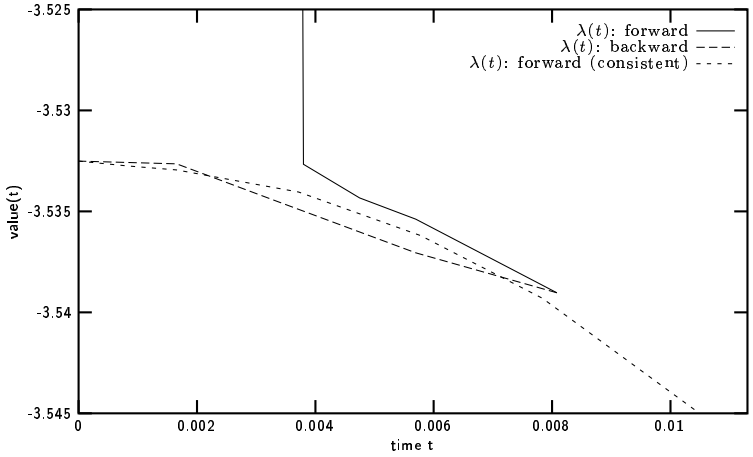

Figure 6.2: Pendulum example: back-tracing history of $\lambda(t)$.

Table 6.3: Consistent initial conditions for pendulum: Back-tracing with loose and strict integration tolerance.

variable	integration tolerance		unit
	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	
$x(0)$	0.399930	0.399999	[m]
$y(0)$	-0.300093	-0.300001	[m]
$u(0)$	$0.554837 \cdot 10^{-6}$	$0.478937 \cdot 10^{-9}$	[m/s]
$v(0)$	$0.739703 \cdot 10^{-6}$	$0.772240 \cdot 10^{-9}$	[m/s]
$\lambda(0)$	-3.53251	-3.53161	[N/m]

6.2.2 A C3-Splitter

As a first example from application we consider the so-called C3-splitting section of a petro-chemical plant. The section splits a mixture of hydrocarbons containing components with two to eight carbon atoms (e.g., propylene has three carbon atoms, butane has four) mainly into a C3-fraction rich in propylene (the fraction also contains the C2 components) and a heavier fraction. Figure 6.3 depicts the flowsheet of the process.

The DAE model in OPTISIM® consists of $n_{\boldsymbol{x}} = 305$ differential and $n_{\boldsymbol{y}} = 1442$ algebraic equations. During a typical simulation run the model exhibits several state and time dependent discontinuities. At the initial time and at all discontinuities Pantelides' Algorithm reports that the DAE is of index two, that there are $n_{\text{ddf}} = 304$ dynamic degrees of freedom, and that $m_{\tilde{\boldsymbol{s}}} = 6$ algebraic constraints

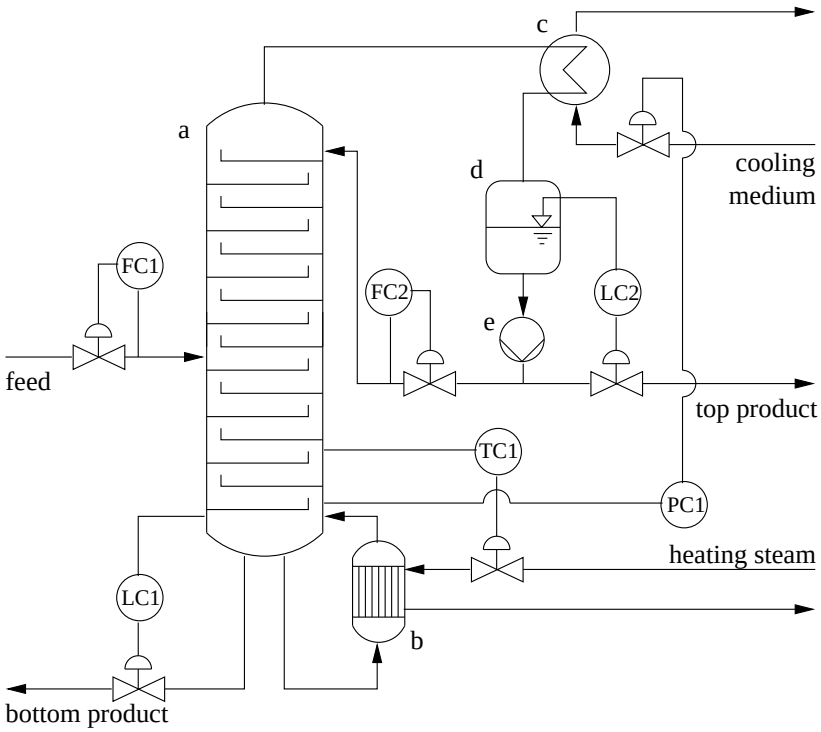

Figure 6.3: Flowsheet of the C3-splitting process including control structure.
main units: a: rectification column, b: reboiler, c: condenser, d: flash drum, e: pump
controllers ($\nu \in \mathbb{N}$)**:** FCν: flow controller, LCν: level controller, TCν: temperature controller, PCν: pressure controller

are to be differentiated in order to obtain an index-1 system, introducing the time derivatives of $n_{\tilde{z}} = 5$ originally algebraic variables. At the initial time and at each discontinuity the transition conditions are uniquely determined by our physical and structural criteria. I.e., exactly one set out of $\binom{305}{304} = 305$ potential sets of differential variables can be directly assigned as dynamic degrees of freedom, cf. Section 3.2.4. The condition number based heuristic rule (cf. Algorithm 8, step **4(c)ii**) is not required.

The main purpose of this medium-scale example is to compare some of the options for the solution of the reduced consistency equations (cf. Section 3.2.5.b). The performance of our hybrid algorithm is tested at the initial point $t = t_0$ and at the first state dependent discontinuity at $t = t_{\text{disc}}$. In order to examine robustness and precision the entire initially guessed state vector – i.e., the quasi steady-state solution at the initial point, and the instationary state before the discontinuity – is perturbed with a relative disturbance of up to 5%. Such a disturbance can be regarded as rather large in the context of chemical engineering.

Table 6.4: C3-Splitter: Algorithm for consistent initialisation using NLEQ1S.

t	disturb.	residual	Δ_{rel}	evaluations res.	evaluations Jac. & res.	time
t_0	0	$1.1 \cdot 10^{-11}$	0 (reference)	0	5	11
	0.01	$1.5 \cdot 10^{-11}$	$1.7 \cdot 10^{-10}$	1	10	32
	0.02	$7.4 \cdot 10^{-12}$	$1.9 \cdot 10^{-10}$	0	9	31
	0.03	$2.2 \cdot 10^{-09}$	$5.4 \cdot 10^{-09}$	1	13	45
	0.05	$7.5 \cdot 10^{-12}$ [p!]	$2.4 \cdot 10^{-01}$	0	13	45
t_{disc}	0	$4.3 \cdot 10^{-13}$	0 (reference)	0	4	9
	0.01	$3.7 \cdot 10^{-12}$ [s!]	$2.6 \cdot 10^{-10}$	0	8	26
	0.02	$7.7 \cdot 10^{-11}$ [s!]	$8.1 \cdot 10^{-10}$	1	10	35
	0.03	$7.2 \cdot 10^{-09}$ [s!],[p!]	$1.4 \cdot 10^{-01}$	0	9	33
	0.05	failure	n/a	2	13	73

Here, especially the physical property computations are sensitively affected. For all experiments the total time derivatives of the derived algebraic equations as well as their Jacobian are evaluated using the partial approximation mode with error controlled disturbances and subdivision of the disturbed variables (see Section 3.2.3).

Table 6.4 contains the results when solely NLEQ1S is used for the solution of the reduced consistency equations. The consistent initial state obtained from the undisturbed initial estimate is taken as the reference for all other computations. Table 6.5 shows the corresponding results for the dog-leg method. The tables indicate that the hybrid algorithm delivers highly accurate results with each of the two nonlinear solvers. The norm of the residual for NLEQ1S ranges from $\mathcal{O}(1.0 \cdot 10^{-12})$ to $\mathcal{O}(1.0 \cdot 10^{-09})$, while the residual for the dog-leg algorithm is typically larger, ranging from $\mathcal{O}(1.0 \cdot 10^{-13})$ to $\mathcal{O}(1.0 \cdot 10^{-06})$. If the perturbation exceeds a certain bound the results may be unphysical, or at least unsuitable as consistent initial values for further simulation as indicated by the [p!] flags. In these cases the solvers converge to points which differ considerably from the reference value. This is indicated by the corresponding values of Δ_{rel} which are then in the order of $1.0 \cdot 10^{-01}$. At the discontinuity we also observe that in several of the perturbed cases the new consistent initial conditions cause an immediate reversal of the sign of the active switching function leading to repeated switching (see the [s!] flags). However, at the discontinuity the absolute value of the active switching function is in the range of $1 \cdot 10^{-8}, \ldots, 1 \cdot 10^{-7}$ so that this problem could be removed by extending the formulation of the switching functions with a small hysteresis ("ϵ-band") as proposed in [Otte 95] (see Section 1.3.2).

We have also applied back-tracing in order to solve the consistent initialisation problem at $t = t_0$ and at $t = t_{\text{disc}}$, starting from sets of undisturbed initial esti-

Table 6.5: C3-Splitter: Algorithm for consistent initialisation using the dog-leg method. The reference value is the corresponding consistent initial state obtained by NLEQ1S starting from an undisturbed initial guess, cf. Table 6.4

				evaluations		
t	disturb.	residual	Δ_{rel}	res.	Jac. & res.	time
t_0	0	$5.6 \cdot 10^{-13}$	$5.6 \cdot 10^{-11}$	3	2	7
	0.01	$3.3 \cdot 10^{-08}$	$5.0 \cdot 10^{-08}$	9	5	21
	0.02	$1.4 \cdot 10^{-06}$	$1.6 \cdot 10^{-06}$	15	8	44
	0.03	$2.9 \cdot 10^{-09}$ [p!]	$2.4 \cdot 10^{-01}$	14	8	33
	0.05	$5.4 \cdot 10^{-07}$ [p!]	$2.4 \cdot 10^{-01}$	22	12	50
t_{disc}	0	$2.3 \cdot 10^{-08}$	$8.5 \cdot 10^{-09}$	1	1	4
	0.01	$1.1 \cdot 10^{-08}$	$1.3 \cdot 10^{-08}$	8	5	23
	0.02	$5.2 \cdot 10^{-11}$ [s!]	$1.4 \cdot 10^{-09}$	13	8	34
	0.03	$2.2 \cdot 10^{-11}$ [p!]	$1.4 \cdot 10^{-01}$	11	7	32
	0.05	$1.4 \cdot 10^{-07}$ [p!]	$3.2 \cdot 10^{-01}$	22	15	63

mates. In both cases the back-tracing algorithm terminates successfully. However, the results differ considerably from the reference values obtained by the solution of the reduced consistency equations with NLEQ1S (cf. Table 6.4). At $t = t_0$ the relative difference between the values of the state variables generated by back-tracing and the reference value is $\Delta_{\text{rel}} = 2.1 \cdot 10^{-1}$, at $t = t_{\text{disc}}$ the relative difference is $\Delta_{\text{rel}} = 2.3 \cdot 10^{-1}$. Such large relative differences in the results have been typical for the unusable results obtained from the numerical solution of the reduced consistency equations for highly disturbed initial guesses. As a consequence we consider the results of the back-tracing method for this example as unreliable.

6.2.3 A Complex Cryogenic Air Separation Plant

As the last example problem for our consistent initialisation algorithm we consider a complex real-world cryogenic air separation plant. Basically the process works as described in Section 2.1.2. However, in place of a single crude argon column (cf. Figure 2.2 on page 28) the plant contains an additional section where the crude argon is further refined to pure argon.

With $n_{\boldsymbol{x}} = 1832$ differential and $n_{\boldsymbol{y}} = 7292$ algebraic equations the process model is about 2.6 times larger than the largest model considered so far in our previous work, cf., e.g., [EKKS 99]. Additionally, the highly nonlinear plant characteristics render the automation of this plant a demanding task. The determination of an open-loop optimal load-change policy by our optimal control algorithm is discussed in Section 6.3.6.

Table 6.6: Air separation plant: Algorithm for consistent initialisation with NLEQ1S using Leimkuhler's approximation. The reference value has been computed using the partial approximation method with subdivision of the disturbed variables and truncation error based choice of the disturbance step-size, cf. Table 6.7.

disturb.	residual	Δ_{rel}	evaluations res.	evaluations Jac. & res.	time
Leimkuhler's approximation (plain)					
0	failure	n/a	12	14	171
0.001	$4.5 \cdot 10^{-08}$	$1.6 \cdot 10^{-08}$	0	6	45
0.010	$3.5 \cdot 10^{-08}$	$1.6 \cdot 10^{-08}$	0	11	94
0.015	$1.4 \cdot 10^{-07}$ [p!]	$7.7 \cdot 10^{-03}$	2	19	161
0.020	failure	n/a	3	55	545
Leimkuhler's approximation (subdivision of variables)					
0	$3.7 \cdot 10^{-11}$	$7.9 \cdot 10^{-14}$	1	8	85
0.001	$4.0 \cdot 10^{-09}$	$1.7 \cdot 10^{-08}$	0	6	80
0.010	$7.0 \cdot 10^{-10}$	$1.6 \cdot 10^{-08}$	0	11	163
0.015	$1.4 \cdot 10^{-07}$ [p!]	$7.7 \cdot 10^{-03}$	2	19	559
0.020	failure	n/a	1	16	225

According to Pantelides' Algorithm at the initial time $t = t_0$ the model of the air separation plant is an index-2 DAE which owns $n_{\text{ddf}} = 1827$ dynamic degrees of freedom. In difference to the C3-splitter examined in the previous Section 6.2.2 only 1823 dynamic degrees of freedom can be assigned by the deterministic rules. I.e., for 1823 differential states continuity conditions are generated based on expert knowledge on the physics of the process and on its mathematical modelling, as well as by the rank criterion Eq. (3.66) (cf. Algorithm 8, step **2**). Thus additional $1827 - 1823 = 4$ continuity conditions have to be determined by the condition-based heuristics (cf. Algorithm 8, step **4(c)ii**). As only differential variables are admitted for assignment as dynamic degrees of freedom $1832 - 1823 = 9$ candidates for assignment are available at this stage. From these 9 candidates 2 are excluded by the block triangularisation technique (see step **3** in Algorithm 8 and Section 3.1.13), finally leaving $\binom{7}{4} = 35$ possible DOF-assignments to the condition-based heuristics. Altogether, the assignment of the dynamic degrees of freedom is completed within $\mathcal{O}(1)$ [s]. The final system of reduced consistency equations has a dimension of 11178 variables/equations.

In the first series of tests the reduced consistency equations are solved at $t = t_0$ by NLEQ1S given a disturbed quasi steady-state as initial estimate. The results are collected in Table 6.6.

Initially, the total time derivatives of the derived algebraic equations are ap-

proximated by Leimkuhler's original formulae (residual according to Eq. (3.45), Jacobian according to Eqs. (3.46a)–(3.46d)). For disturbances of 0.1% and 1.0% the algorithm is successful. When the disturbance is increased to 1.5% the generated solution is not suitable as initial data for further dynamic simulation (the integrator fails at the second step after initialisation). At 2.0% disturbance NLEQ1S fails. The failure of the solver to establish convergence in the undisturbed case is a major problem as consistent initialisation starting from a quasi steady-state is one of the standard tasks which we expect our algorithm to solve.

By applying subdivision of the variables in Leimkuhler's formulae (residual using Eq. (3.48), Jacobian according to Eqs. (3.49a)–(3.49b), cf. Section 3.2.3.b) this failure is avoided. Additionally, with the decomposition of the variables in the case of 0.1% and 1.0% disturbance the residual is slightly smaller. In these disturbed cases, however, the consistent initialisation procedure requires roughly twice the computational time in comparison to the algorithm without subdivision of the variables. For 1.5% disturbance the result is again unsuitable, while the solver still fails at a disturbance of 2.0%.

In the second test series summarised in Table 6.7 the same problem is solved again. In difference to the first test series analytical expressions for the residual of the derived algebraic equations Eq. (3.54) are used, and the partially approximated Jacobian Eqs. (3.55a)–(3.55d) is employed. This series itself is divided into three parts. In the first part the approximated portion of the Jacobian is built according to Leimkuhler's original formulae, in the second part Leimkuhler's formulae are used in connection with subdivision of the disturbed variables, and in the third part Leimkuhler's formulae are applied employing subdivision of the disturbed variables and truncation error control. The results using the various approximation methods are almost identical. In each case for 1.5% disturbance an unusable initial condition is generated, and at 2.0% disturbance the solver fails. Also the number of residual and Jacobian evaluations only shows minor differences. In comparison to Table 6.6 the results are in general slightly better. Additionally, the second setting without subdivision of the disturbed variables does not suffer from the convergence problem which appeared in the case of the plain Leimkuhler approximation. Although in this test series the setting without subdivision of the disturbed variables delivers the quickest results we stick to the partially approximated Jacobian with subdivision of the disturbed variables and truncation error control as the standard, as the latter has shown more reliable in some small academic test examples.

In the last two series of tests the second test scenario is extended by using either the Levenberg-Marquardt solver or the dog-leg solver in sequence with NLEQ1S. In each case the derivative array equations are generated with subdivision of the disturbed variables and truncation error control for the finite difference approximations.

Table 6.8 shows the results when using the dog-leg method as initial solver. In this example the dog-leg solver does not contribute to the robustness of the

Table 6.7: Air separation plant: Algorithm for consistent initialisation with NLEQ1S using the direct evaluation mode.

disturb.	residual	Δ_{rel}	evaluations res.	evaluations Jac. & res.	time
partially approximated Jacobian (subdivision & error control)					
0	$3.9 \cdot 10^{-12}$	0 (reference)	1	8	61
0.001	$4.0 \cdot 10^{-09}$	$1.7 \cdot 10^{-08}$	0	6	79
0.010	$6.9 \cdot 10^{-10}$	$1.6 \cdot 10^{-08}$	0	11	135
0.015	$2.4 \cdot 10^{-09}$ [p!]	$7.5 \cdot 10^{-03}$	0	14	187
0.020	failure	n/a	3	16	194
partially approximated Jacobian (with subdivision)					
0	$3.9 \cdot 10^{-12}$	0.0	1	8	67
0.001	$4.0 \cdot 10^{-09}$	$1.7 \cdot 10^{-08}$	0	6	70
0.010	$6.9 \cdot 10^{-10}$	$1.6 \cdot 10^{-08}$	0	11	140
0.015	$1.4 \cdot 10^{-07}$ [p!]	$7.5 \cdot 10^{-03}$	2	19	256
0.020	failure	n/a	3	16	179
partially approximated Jacobian (plain)					
0	$3.9 \cdot 10^{-12}$	0.0	1	8	60
0.001	$4.0 \cdot 10^{-09}$	$1.7 \cdot 10^{-08}$	0	6	53
0.010	$6.9 \cdot 10^{-10}$	$1.6 \cdot 10^{-08}$	0	11	94
0.015	$1.4 \cdot 10^{-07}$ [p!]	$7.7 \cdot 10^{-03}$	2	19	188
0.020	failure	n/a	3	16	121

combined method. As indicated by the numbers in brackets the dog-leg algorithm even fails at lower disturbance levels than NLEQ1S. In these cases bad convergence of the dog-leg method leads to excessive computational times.

Table 6.9 summarises the results of the combination of Levenberg-Marquardt solver and Newton's method. When using the Levenberg-Marquardt method as pre-solver the reduced consistency equations can be successfully solved for higher disturbance levels in the initial estimate than with the pure Newton solver. The cascaded solvers succeed to generate the correct consistent initial conditions even at a disturbance of 4% (recall that in the preceding examples NLEQ1S on its own returned an unsuitable solution for 2% disturbance, while it completely failed at a disturbance level of 4%, cf. Table 6.7). Unluckily also the time required for the solution of the problem drastically increases – at 2% disturbance the solution of the reduced consistency equations required approximately 1/2 hour, at 4% disturbance an hour of CPU time was elapsed. In summary this example indicates that the Levenberg-Marquardt method may considerably increase the robustness of the solution procedure, but it should only be chosen in case of severe problems with NLEQ1S (in these cases it is better to have slow convergence than fast failure)

Table 6.8: Air separation plant: Algorithm for consistent initialisation with dog-leg method and NLEQ1S in sequence. The first numbers are the data for the dog-leg solver, the second numbers are the data for NLEQ1S. If these numbers are in brackets the dog-leg solver failed and the original estimate was restored as initial guess for NLEQ1S.

			evaluations		
disturb.	residual	Δ_{rel}	res.	Jac. & res.	time
0	$3.9 \cdot 10^{-12}$	0 (reference)	1 ; 1	1 ; 8	15 ; 67
0.001	$3.0 \cdot 10^{-11}$	$1.7 \cdot 10^{-08}$	7 ; 0	4 ; 6	57 ; 55
0.010	$6.9 \cdot 10^{-10}$	$1.6 \cdot 10^{-08}$	(89) ; 0	(78) ; 11	(1148) ; 133
0.015	$2.4 \cdot 10^{-09}$ [p!]	$7.5 \cdot 10^{-03}$	(3) ; 0	(2) ; 14	(28) ; 178
0.020	failure	n/a	(58) ; 3	(47) ; 16	(731) ; 179

Table 6.9: Air separation plant: Algorithm for consistent initialisation with Levenberg-Marquardt method and NLEQ1S in sequence. The first numbers are the data for the Levenberg-Marquardt solver, the second numbers are the data for NLEQ1S.

			evaluations		
disturb.	residual	Δ_{rel}	res.	Jac. & res.	time
0	$3.9 \cdot 10^{-12}$	0 (reference)	0 ; 1	5 ; 8	158 ; 61
0.001	$3.0 \cdot 10^{-11}$	$1.8 \cdot 10^{-08}$	0 ; 2	24 ; 14	1120 ; 146
0.010	$9.6 \cdot 10^{-11}$	$2.5 \cdot 10^{-08}$	0 ; 0	29 ; 9	1366 ; 99
0.015	$3.0 \cdot 10^{-10}$	$2.1 \cdot 10^{-08}$	0 ; 1	29 ; 13	1197 ; 151
0.020	$4.4 \cdot 10^{-11}$	$7.1 \cdot 10^{-09}$	0 ; 1	34 ; 16	1406 ; 195
0.040	$5.9 \cdot 10^{-11}$	$2.5 \cdot 10^{-08}$	0 ; 7	61 ; 31	3557 ; 366
0.080	$3.0 \cdot 10^{-11}$ [p!]	$1.3 \cdot 10^{-01}$	0 ; 0	15 ; 13	1532 ; 162

unless considerably more powerful computer hardware is used.

The back-tracing algorithm fails to solve the consistent initialisation problem for this model at the initial time. During the backwards integration phase the step-size is steadily decreased in order to preserve the bound for the local truncation error until the minimum step-size is reached. Backwards integration then aborts far from the destination point as the local truncation error still exceeds the user given bound.

6.3 Sensitivity Matrices and Optimisation Problems

I regard as quite useless the reading of large treatises of pure analysis:
too large a number of methods pass at once before the eyes.
It is in the works of applications that one must study them;
one judges their ability there and one apprises the manner of making use of them.

Joseph-Louis Lagrange

In this section our algorithms for the treatment of sensitivities at discontinuities are examined. The examples are given in increasing order of complexity and difficulty, ranging from a simple ODE problem to a complex air separation plant modelled by a very large-scale index-2 DAE system.

In the nontrivial cases analytical expressions that could be used in order to validate the correctness of the numerically computed sensitivity matrices are in general not available. For the large-scale examples with a larger number of parameters also the numerical approximation of the sensitivity matrices by finite differences as a potential reference is in general impractical. Additionally, these finite differences are not reliable as the BDF integrator employed is not designed for the integration of a DAE at the perturbed parameter sets with the same (fixed) order and step size history as used during the integration at the undisturbed parameter set. The reliable approximation of parametric sensitivities by numerical differentiation can only be guaranteed when the same step-size and order sequence is used for both disturbed and undisturbed integration [Kieh 99].

Remark 6.3:
We have extended the BDF integrator of OPTISIM® in such a way that the adaptive step-size and order selection strategy can be temporarily replaced by a recorded step-size and order sequence, cf. Section 5.4 (disturbed back-tracing phase). After some minor modifications this logic might as well be used during an entire integration horizon. However, approximation of sensitivity matrices by finite differences and *external numerical differentiation* is not the subject of this work. ◊

We have chosen to follow a more instructive way and use the sensitivities as basic data for the solution of optimal control problems by our direct single shooting algorithm (see Section 2.4.2). However, slow convergence of the optimiser or even failure is not necessarily caused by inaccurate or wrong sensitivities. Both may have their source in the problem setting or, in the worst case, in programming errors. On the other hand, to some extent the state-of-the-art SQP methods applied have shown robust against small errors in the sensitivities, see, e.g., Section 6.3.3 and Section 6.3.5. Thus a successfully solved optimal control problem may indicate the validity of the sensitivities, but it cannot provide a numerical proof.

In the optimisation examples failures of the optimiser due to an excessive number of iterations are marked with [m!]. If better values of the optimisation variables could not be found although the conditions for an optimum have not all been satisfied the result is marked with [c!]. If the conditions for an optimum have

been satisfied, but the requested accuracy of the result could not be established the solution is marked as [a!].

6.3.1 A Simple ODE Example

Consider the ODE-IVP[iv] on $[t_0, t_f] \equiv [0, 1]$

$$\dot{\boldsymbol{\xi}}(t;\boldsymbol{p}) = \begin{cases} -\boldsymbol{p}\,; & \boldsymbol{\xi}(t;\boldsymbol{p}) > 1\,, \\ -\boldsymbol{p}^2\,; & \text{otherwise}\,, \end{cases} \tag{6.4a}$$

$$\boldsymbol{\xi}(t_0;\boldsymbol{p}) = 2\,, \tag{6.4b}$$

$\boldsymbol{\xi} \in \mathbb{R}$, with the parameter $\boldsymbol{p} \in \mathbb{R}_+ \backslash \{0\}$. The right hand side of the ODE switches if

$$-\boldsymbol{p} \cdot t_{\text{disc}} + 2 = 1\,,$$

i.e., at $t_{\text{disc}}(\boldsymbol{p}) = 1/\boldsymbol{p}$. Therefore, the parametric sensitivity of the location of the discontinuity with respect to $\boldsymbol{p}$ is $\partial t_{\text{disc}}(\boldsymbol{p})/\partial \boldsymbol{p} = -1/\boldsymbol{p}^2$. Further at $t = t^-_{\text{disc}}$ the sensitivity of the state with respect to $\boldsymbol{p}$ is

$$\boldsymbol{\omega}^-(\boldsymbol{p}) := \frac{\partial \boldsymbol{\xi}^-(\boldsymbol{p})}{\partial \boldsymbol{p}} = \frac{\partial}{\partial \boldsymbol{p}}(2 - \boldsymbol{p}t)\big|_{t=t^-_{\text{disc}}} = -t^-_{\text{disc}} = -\frac{1}{\boldsymbol{p}}\,.$$

At the discontinuity we enforce continuity in the state $\boldsymbol{\xi}$, i.e.[v],

$$\boldsymbol{\xi}^+ = \boldsymbol{h}(t, \boldsymbol{\xi}^-, \boldsymbol{p}) = \boldsymbol{\xi}^-\,.$$

Then the sensitivity $\boldsymbol{\omega}^+(\boldsymbol{p}) = \partial \boldsymbol{\xi}^+(\boldsymbol{p})/\partial \boldsymbol{p}$ immediately after the discontinuity computes as

$$\begin{aligned} \boldsymbol{\omega}^+ &= \left\{ -\dot{\boldsymbol{\xi}}^+ + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial t} \right\} \cdot \frac{\partial t_n}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{h}^-}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}^-}{\partial \boldsymbol{p}} \\ &= (\boldsymbol{p}^2 - \boldsymbol{p}) \frac{-1}{\boldsymbol{p}^2} + \frac{-1}{\boldsymbol{p}} = -1\,, \end{aligned}$$

cf. Eq. (4.5). In this special example $\boldsymbol{\omega}^+(\boldsymbol{p})$ is independent of $\boldsymbol{p}$.

In Figure 6.4 the graph of the sensitivity matrix $\boldsymbol{\omega}(t;\boldsymbol{p})$ is plotted for $\boldsymbol{p} = 1.61803$ (this value is motivated below). The solid line depicts the correct sensitivities obtained when using our hybrid algorithm discussed in Section 4.3.3 for the transfer of the sensitivities at the discontinuity. The back-tracing based algorithm introduced in Section 4.3.2 gives the same sensitivities as indicated by the crosses. Without correction wrong sensitivity information is computed after the discontinuity, see the dashed line.

[iv] Due to the internal treatment of freely formulated differential or algebraic equations within OPTISIM® the actual model equations constitute a DAE of differential index 1.

[v] For this simple example the notation used in Section 4.1.1 is applied.

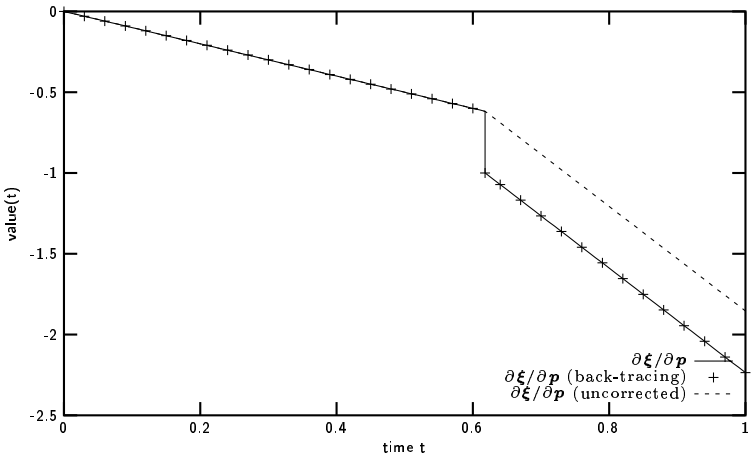

Figure 6.4: Sensitivity trajectories for simple example, $\boldsymbol{p} = 1.61803$.

In order to illustrate the effects of the uncorrected sensitivity information the behaviour of the SQP algorithm NPSOL [GMSW 98] (NAG® routine E04UCF [NAG 94b]) is examined at the optimal control problem

$$\mathcal{J}[\boldsymbol{p}] = \boldsymbol{\xi}(t_\mathrm{f};\boldsymbol{p}) \longrightarrow \min_{\boldsymbol{p}}! \tag{6.5a}$$

$$0 \leq \boldsymbol{\xi}(t;\boldsymbol{p})\,;\; t \in [t_0, t_\mathrm{f}]\,, \tag{6.5b}$$

with the dynamics defined in Eqs. (6.4a)–(6.4b). The solution of the optimal control problem Eqs. (6.5a)–(6.5b), Eqs. (6.4a)–(6.4b) is given by

$$\boldsymbol{\xi}(t_\mathrm{f};\boldsymbol{p}) = 1 + (-\boldsymbol{p}^2 + \boldsymbol{p}^2 t_\mathrm{disc}(\boldsymbol{p})) \stackrel{!}{=} 0\,.$$

Due to the restriction of $\boldsymbol{p}$ to strictly positive reals the optimiser is

$$\boldsymbol{p}^* = \frac{1+\sqrt{5}}{2} \approx 1.61803\,.$$

For the numerical solution of the optimal control problem the path inequality constraint Eq. (6.5b) is discretised on an equidistant grid of 30 points. In the experiment an optimisation precision of $1.0 \cdot 10^{-5}$ is requested. The result of the experiment is collected Table 6.10. Using the sensitivities generated employing either the hybrid algorithm or the back-tracing based method for the sensitivity transfer at the discontinuity the optimal value is computed up to a maximum absolute difference of $4.0 \cdot 10^{-5}$. Only a moderate number of integrations with and without sensitivity analysis is required. The differences between hybrid and back-tracing based algorithm are neglectable. In case of uncorrect sensitivity in-

formation NPSOL terminates successfully starting from an initial guess of 3.0 after requesting more than twice the number of optimisation function and gradient evaluations. For all other initial guesses the optimisation algorithm finally fails due to an excessive number of iterations as indicated by the requested number of integrations. Though it succeeds in locating the optimum roughly. A closer examination of the iteration history in the failed tests shows that initially the optimisation quickly converges closely towards the solution, but then the optimum value cannot be determined to the requested accuracy.

6.3.2 Parameter Identification for a Bottle Filling Process

As a first numerical illustration for the correct transfer of sensitivities across state-dependent discontinuities with an industrial background we consider a parameter identification problem introduced in [KKES 01], [KSBK 01]. The model DAE is of differential index 1. In extension of the results published in [KKES 01] and [KSBK 01] the number of identified parameters is increased from 3 to 7. Additionally, the two algorithms for the treatment of sensitivities at discontinuities, i.e., our algorithm based on back-tracing and numerical differentiation of the jump function (see Section 4.3.2) as well as our algorithm based on the solution of the reduced system of consistency equations (cf. Section 4.3.3) are examined.

The problem arising from industrial application is the filling of industrial gases (e.g., N_2, O_2, or Ar) into high pressure gas bottles for retail. Initially, the bottles are depressurised to vacuum in order to avoid contamination of the pure product with residue in the bottle. The process in view is the subsequent filling of the bottles with the gas from a high pressure gas tank up to a final pressure of 200 [bar]. An important point in the investigation of potentials for the acceleration of the filling process is the heat balance of the bottle system. For this investigation

Table 6.10: Simple optimal control problem, with and without corrected sensitivities. Note that in the case of $\boldsymbol{p} < 1$ the structure of the problem changes during the course of the optimisation as the discontinuity is initially located outside the prediction horizon $[t_0, t_f]$.

initial value	cons. init.		back-tracing		no correction	
	$\boldsymbol{p}$	eval	$\boldsymbol{p}$	eval	$\boldsymbol{p}$	eval
0.15	1.618036	7	1.618037	7	1.623465	136 [m!]
0.50	1.618036	7	1.618037	7	1.623096	130 [m!]
2.00	1.618036	6	1.618037	6	1.620091	201 [m!]
3.00	1.618035	11	1.618036	11	1.618034	26

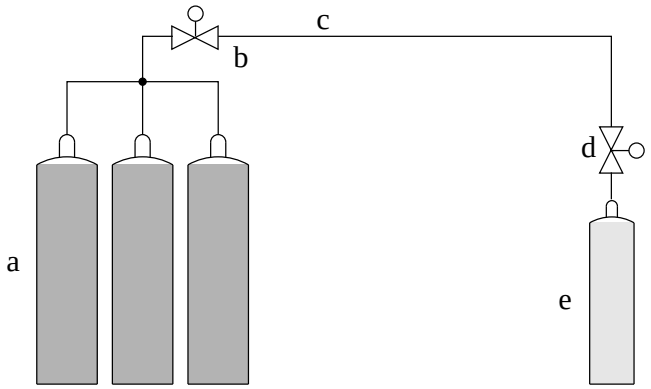

Figure 6.5: High pressure gas bottle filling.
units: a: high pressure bottle battery, b: valve, c: tube, d: valve, e: retail bottle

knowledge on the coefficients for the gas-bottle heat transfer is required in order to determine the heat flows in the system. As no model for the prediction of heat transfer coefficients in a fast pressurised gas volume is available from the literature, the heat transfer coefficients must be determined from measurements. In a first approximation we consider a parameter fitting problem where constant heat transfer coefficients are to be identified based on experimental data.

In the physical experiment a single depressurised gas bottle **(e)** is filled from a bottle battery **(a)**, cf. Figure 6.5. During the experiment the valve on the battery side **(b)** is always 100% opened. The second valve **(d)** between bottle battery and bottle to be filled is opened at time zero and closed after 120 seconds when pressure equilibrium has been achieved. Measured observables are the temperature at the filling pipe T_{pipe}, the temperature at the outside of a bottle in the bottle battery T_{battery}, and the temperature at the outside of the retail gas bottle T_{bottle}. Based on measurements from the experiment with a non-insulated bottle, $n_{\boldsymbol{p}} = 7$ parameters $\boldsymbol{p}$ (heat transfer coefficients $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ for the bottle, heat transfer coefficients $\boldsymbol{p}_3$, $\boldsymbol{p}_4$ for the pipe, initial temperature $\boldsymbol{p}_5$ and pressure $\boldsymbol{p}_6$ of the bottles in bottle battery, and the ambient temperature $\boldsymbol{p}_7$) are to be determined by the dynamic parameter identification algorithm. The simulation model for the entire process consisting of bottle battery, the single gas bottle, piping, and valves has $n_{\boldsymbol{x}} + n_{\boldsymbol{y}} = 162$ differential and algebraic equations, and – as already mentioned – the differential index 1.

Remark 6.4:
Some special aspects of parameter identification have been treated in Remark 2.3 on page page 39, in Section 2.4.2, and in Section 5.1. ◊

The parameter identification problem has been solved with the specialised SQP algorithm NLSSOL (NAG® routine E04UNF [NAG 94c]) and with the general SQP solver SNOPT [GMS 97a], [GMS 97b]. NLSSOL terminates after

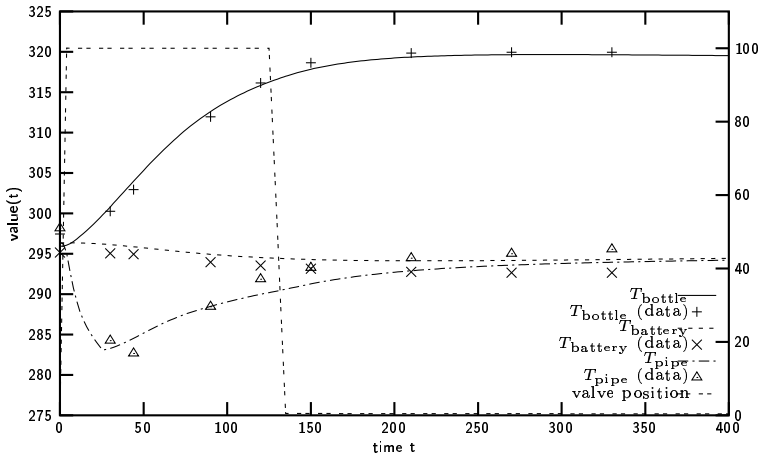

Figure 6.6: Non-insulated bottling process: Fitted and measured data. Retail bottle temperature T_{bottle}, bottle battery temperature T_{battery}, and pipe temperature T_{pipe} are within $[275; 325]$ [K]. The valve position of valve **(d)** ranges between 0% and 100%.

23 major iterations, 99 minor iterations, and 59 integrations including sensitivity analysis with an objective value of 40.40. SNOPT terminates after 26 major iterations, 63 minor iterations, and 56 integrations including sensitivity analysis with an objective value of 40.94. In both cases the same (bad) initial guess for the parameters was used which gives an initial objective value of 2718. The optimised sets of parameters show only major differences in the heat transfer coefficient for the pipe and in the initial temperature of the bottle battery. By physical reasons the parameters obtained with SNOPT are taken as the final result. The results of the optimisation (solid lines) and the 8 sampling points entering the optimisation problem are shown in Figure 6.6.

Optimisation with the modified Gauß-Newton algorithm LSQFDN [GiMu 78] (NAG® routine E04GDF [NAG 94a]) starting from the initially estimated parameters fails as the optimiser tends to drive the parameters into unphysical regions of the parameter space. In consequence, the mathematical model breaks down, numerical integration and sensitivity analysis is aborted, and optimisation fails. As LSQFDN is an optimisation method for the solution of unconstrained optimisation problems these failures cannot be avoided. Moreover, LSQFDN shows the same behaviour when (externally) restarted from the last suitable set of parameters. Thus this optimisation algorithm it is not applicable in this example. However, starting from the optimised parameters obtained from SNOPT LSQFDN successfully indicates local optimality of this set, confirming the former result.

In Figure 6.7 measurements originating from another physical experiment with

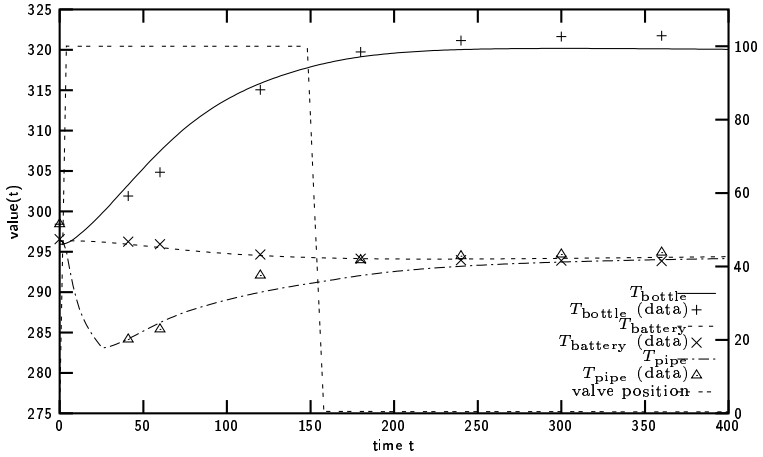

Figure 6.7: Insulated bottling process: Measured data and simulation with the parameters obtained for non-insulated case. For an explanation of the graph cf. Figure 6.6.

an insulated bottle are compared with a numerical simulation using the parameters identified above. The measurements are available at 9 points in time. The comparison indicates that the fitted parameters are applicable even in a not too closely related scenario.

In Figure 6.8 example trajectories of the parametric sensitivity functions for the (non-insulated) bottle filling process at the optimised parameter set are depicted. The solid and dotted lines are the sensitivities obtained with our hybrid algorithm for consistent initialisation and corresponding sensitivity correction at discontinuities. The symbols $+$, $\times$, and $\triangle$ indicate the numerical results for the sensitivity functions without correction. Remarkably, the numerical approximations for the sensitivity functions are almost independent from the treatment of the discontinuities although at least two state dependent discontinuities are present. This observation is also valid for all other sensitivities entering the optimisation. The sensitivities obtained using the back-tracing based algorithm are not shown as they are identical with the sensitivities obtained employing rigorous consistent initialisation.

To some extent this behaviour of the sensitivities can be explained directly. The first state dependent discontinuity (at $t = t_1$) is independent from all parameters, i.e., $\partial t_1 / \partial \boldsymbol{p}_\nu = 0$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$, and thus the corresponding terms in $\Delta\tilde{\boldsymbol{\omega}}_1$ (cf. Eq. (4.45)) primarily responsible for the modification of the sensitivities in Eq. (4.46) after the discontinuity are nullified. Additionally, at this discontinuity the numerically consistent initial conditions computed by back-tracing are nearly the same as the consistent initial conditions obtained from the solution of

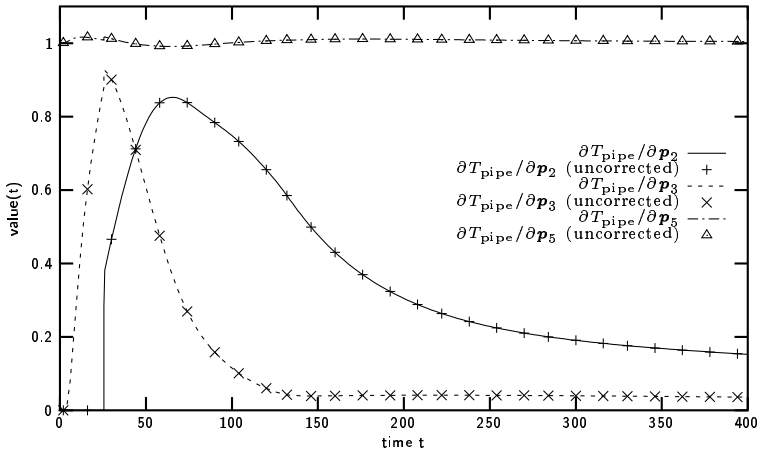

Figure 6.8: Non-insulated bottling process: Sensitivities of some of the state variables depicted in Figure 6.6. $\partial T_{\text{pipe}}/\partial \boldsymbol{p}_2$ has been divided by 3000, $\partial T_{\text{pipe}}/\partial \boldsymbol{p}_3$ has been divided by 25.

the reduced derivative array equations. In detail, the relative deviation of the numerically consistent initial conditions from the rigorously computed consistent initial conditions measured according to Eq. (6.1) is $\Delta_{\text{rel}} = 5.94 \cdot 10^{-5}$ in the state variables, and $\Delta_{\text{rel}} = 9.47 \cdot 10^{-5}$ for the first order time derivatives. Therefore, integration without consistent initialisation gives the same trajectory as with rigorous consistent initialisation. This, in connection with the independence of the discontinuity from the parameters explains why sensitivity analysis without special correction generates appropriate sensitivity functions. However, the situation is different for the the second discontinuity as t_2 non-trivially depends on all parameters, i.e., $\partial t_2/\partial \boldsymbol{p}_\nu \neq 0$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$. As above, the results of back-tracing and rigorous consistent initialisation for the state variables are close ($\Delta_{\text{rel}} = 7.18 \cdot 10^{-6}$), but the first order time derivatives differ significantly ($\Delta_{\text{rel}} = 4.67 \cdot 10^{-2}$). The discontinuous model equations do not exhibit a special structure at this point, either.

6.3.3 The Planar Pendulum (Index-2), Immersing in a Liquid

In this section the two dimensional pendulum introduced in Section 6.2.1 is extended. Consider a pendulum mounted above a liquid medium, e.g., a pendulum located above a water-filled bath tub. As depicted in Figure 6.9 the distance from the liquid surface to the pivot of the pendulum h is chosen smaller than the length of the rod l. Therefore the mass point immerses in the liquid for some time during

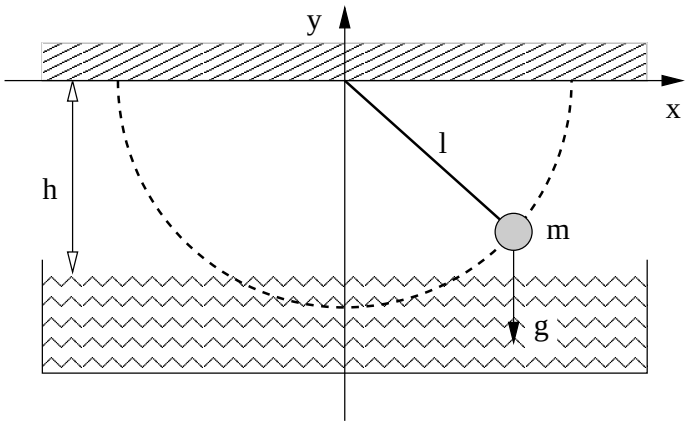

Figure 6.9: The two dimensional mathematical pendulum with fixed length l, immersing in a liquid.

each of its oscillations. Now assume that the mass point is a metal ball. Then the ball is decelerated while immersed in liquid due to friction with the liquid, and it is decelerated while outside the liquid due to friction with the atmosphere.

6.3.3.a A Model for the Immersing Pendulum

The absolute value of the frictional force for turbulent flow in fluid or gas is $|\boldsymbol{F}_{\mathrm{w}}| = c_{\mathrm{w}} \frac{\rho}{2} A |\boldsymbol{v}|^2$ [Stöc 93]. In two dimensions the directed force is

$$\begin{aligned} \boldsymbol{F}_{\mathrm{w}} &= -c_{\mathrm{w}} \frac{\rho}{2} A |\boldsymbol{v}|^2 \frac{\boldsymbol{v}}{|\boldsymbol{v}|} = -c_{\mathrm{w}} \frac{\rho}{2} A |\boldsymbol{v}| \boldsymbol{v} \\ &= -c_{\mathrm{w}} \frac{\rho}{2} A \sqrt{\dot{x}^2 + \dot{y}^2} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\gamma \sqrt{u^2 + v^2} \begin{bmatrix} u \\ v \end{bmatrix} , \end{aligned}$$

where $\gamma := c_{\mathrm{w}} \rho A/2$ contains all constant terms. Here, $\rho \in \mathbb{R}$ denotes the density of the medium, $A \in \mathbb{R}$ is the active cross sectional area between body and medium, and $c_{\mathrm{w}} \in \mathbb{R}$ is a geometric constant[(vi)]. $\boldsymbol{v} \in \mathbb{R}^2$ is the velocity vector. The Lagrangian for the two dimensional pendulum is

$$\begin{aligned} L &= T - U + \frac{1}{2} \lambda G \\ &= \frac{1}{2} m(\dot{x}^2 + \dot{y}^2) - mgy + \frac{1}{2} \lambda (x^2 + y^2 - l^2) , \end{aligned}$$

[(vi)] c_{w} depends on the Reynold's number. In the case of a smooth ball in a medium the Reynold's number is given by $Re = (r\rho |\boldsymbol{v}|)/\eta$, where r is the radius of the ball, ρ is the density of the medium, $|\boldsymbol{v}|$ is the speed of the ball, and η is the viscosity of the medium. Taking c_{w} as a constant is a simplification.

where T is the kinetic energy, U is the potential energy, and G is the length constraint (cf. Eq. (6.2e)). The Euler-Lagrange equations for systems containing forces $\boldsymbol{F}$ that cannot be derived from a potential [Gold 63]

$$L_{\boldsymbol{\xi}_{(\nu)}} - \frac{\mathrm{d}}{\mathrm{d}t} L_{\dot{\boldsymbol{\xi}}_{(\nu)}} = -\boldsymbol{F}_{(\nu)}\,, \quad \nu = 1, \ldots, n_{\boldsymbol{\xi}}\,,$$

with $\boldsymbol{\xi} = [\boldsymbol{\xi}_{(1)}, \boldsymbol{\xi}_{(2)}, \boldsymbol{\xi}_{(3)}]^T := [x, y, \lambda]^T$, $n_{\boldsymbol{\xi}} = 3$, and $\boldsymbol{F} := -\gamma\sqrt{\dot{x}^2 + \dot{y}^2}[\dot{x}, \dot{y}, 0]^T$ provide the equations of motion

$$\begin{aligned}
\lambda x - \frac{\mathrm{d}}{\mathrm{d}t}(m\dot{x}) &= \gamma\dot{x}\sqrt{\dot{x}^2 + \dot{y}^2}\,,\\
-mg + \lambda y - \frac{\mathrm{d}}{\mathrm{d}t}(m\dot{y}) &= \gamma\dot{y}\sqrt{\dot{x}^2 + \dot{y}^2}\,,\\
\frac{1}{2}(x^2 + y^2 - l^2) &= 0\,.
\end{aligned}$$

The corresponding first order system is then

$$\dot{x} = u\,, \tag{6.6a}$$

$$\dot{y} = v\,, \tag{6.6b}$$

$$\dot{u} = \frac{\lambda}{m}x - \frac{\gamma}{m}u\sqrt{u^2 + v^2}\,, \tag{6.6c}$$

$$\dot{v} = \frac{\lambda}{m}y - \frac{\gamma}{m}v\sqrt{u^2 + v^2} - g\,, \tag{6.6d}$$

$$0 = x^2 + y^2 - l^2\,. \tag{6.6e}$$

Eqs. (6.6a)–(6.6e) set up an index-3 DAE. By replacing the length constraint Eq. (6.6e) with its first total derivative with respect to time

$$0 = xu + yv\,, \tag{6.7}$$

an index-2 DAE is obtained.

In simplification we assume that the ball is totally immersed in the liquid when its barycentre is on or below the liquid surface, i.e., the density ρ of the medium is given by

$$\rho = \rho(y) = \begin{cases} \rho_{\mathrm{liq}}\,; & y \le -h\,,\\ \rho_{\mathrm{gas}}\,; & \text{else}\,. \end{cases}$$

This condition creates a state dependent discontinuity in Eq. (6.6c) and Eq. (6.6d) at $y = -h$ as there γ changes discontinuously.

We have chosen to fix a platinum ball of mass $m = 1\,[\mathrm{kg}]$ at the end of the rod with length $l = 10.0\,[\mathrm{m}]$. Due to the density of platinum $\rho_{\mathrm{Pt}} = 21.090 \cdot 10^3\,[\mathrm{kg/m^3}]$ the ball has a radius of $0.0225\,[\mathrm{m}]$ and a cross-sectional area of $1.58 \cdot 10^{-3}\,[\mathrm{m^2}]$. We take $c_{\mathrm{w}} = 0.13$, assuming in simplification strongly turbulent flow around the

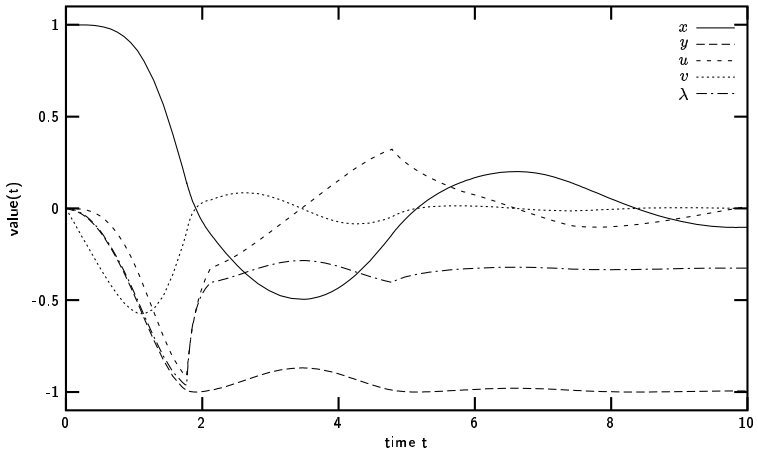

Figure 6.10: Trajectories for the immersing pendulum. The values for x and y have been divided by a factor 10.0, u and v have been divided by a factor 15.0, and λ has been divided by a factor 3.0.

ball at any time. The liquid medium is water with a density of $\rho_{\text{liq}} := \rho_{H_2O} = 1.003 \cdot 10^3 \,[\text{kg/m}^3]$, the gaseous medium is air with an average density of $\rho_{\text{gas}} := \rho_{\text{air}} = 1.2928 \,[\text{kg/m}^3]$, cf. [Stöc 93]. The distance between pivot and liquid surface is $h = 9.90\,[\text{m}]$.

In Figure 6.10 the trajectories for the immersing pendulum have been plotted. Figure 6.11 shows the results of a numerical sensitivity analysis with respect to ρ_{liq} and ρ_{gas} with correctly transferred sensitivities at the discontinuities computed by our algorithm based in the solution of the reduced consistency equations, and wrong sensitivity information obtained without additional numerical treatment of the sensitivities at the discontinuities. The corrected sensitivities have been confirmed in an additional experiment by plain finite differences.

In Figure 6.12 the result of the back-tracing based algorithm is compared with the sensitivity functions obtained from the algorithm based on the solution of the reduced consistency equations. While $\partial x/\partial \rho_{\text{liq}}$ is computed correctly, incorrect values are generated for $\partial u/\partial \rho_{\text{gas}}$. We have observed that in this example all sensitivities with respect to ρ_{liq} are computed correctly by the back-tracing based algorithm, while it generates incorrect sensitivity information with respect to ρ_{gas}.

6.3.3.b Parameter Identification for the Immersing Pendulum

Starting from different initial guesses for the liquid and gas densities (cf. Table 6.11) the task is to reconstruct the nominal values of these parameters by our direct sin-

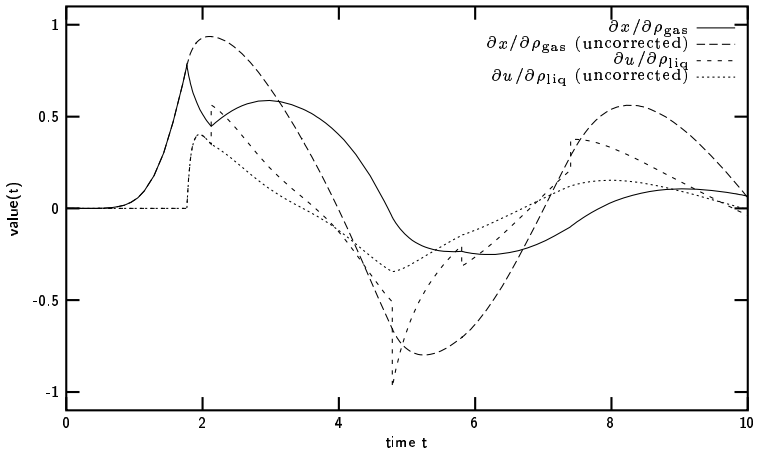

Figure 6.11: Sensitivities for the immersing pendulum: Correct treatment of the discontinuities by hybrid algorithm, and no special treatment of discontinuities. The values for $\partial x/\partial\rho_{\text{gas}}$ have been divided by a factor 0.03, $\partial u/\partial\rho_{\text{liq}}$ has been divided by a factor 0.0085. The spikes in the corrected sensitivities $\partial u/\partial\rho_{\text{liq}}$ indicate penetrations of the waterline.

gle shooting method. In the experiment 21 equidistant samples of the nominal trajectory in the interval $[0; 10]$ are provided as measurement data. As optimisation methods the modified Gauß-Newton algorithm LSQFDN [GiMu 78] (NAG® routine E04GDF [NAG 94a]), the specialised SQP algorithm NLSSOL (NAG® routine E04UNF [NAG 94c]), and the general SQP solver SNOPT [GMS 97a], [GMS 97b] are employed.

Table 6.11: Initial guesses for parameter identification of immersing pendulum.

set	ρ_{gas}	ρ_{liq}
nominal	1.2928	1003
1	1.5	700
2	0.4	1500
3	0.5	1200

In the first series of tests (cf. Table 6.12) the optimisers are provided with the correct sensitivities. The modified Gauß-Newton method delivers the best results with a small number of function and gradient evaluations. Both SQP algorithms encounter difficulties with the identification of the density of the gas ρ_{gas} (0.5 has been set as the lower bound for this parameter). Not surprisingly, SNOPT as

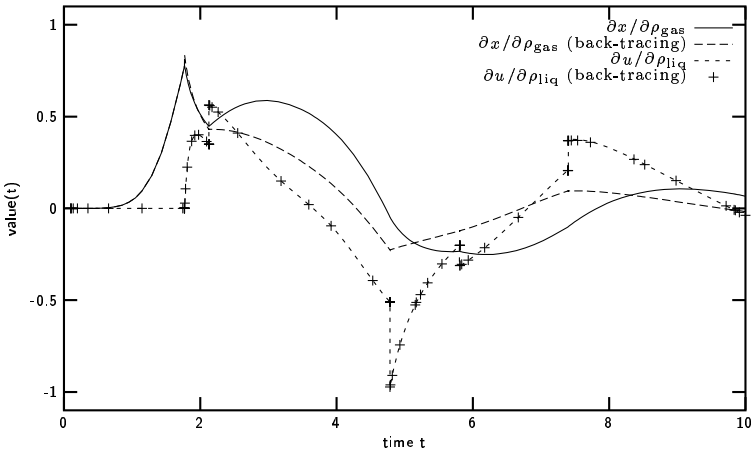

Figure 6.12: Sensitivities for the immersing pendulum: Back-tracing based approach in comparison with correction based on solution of reduced consistency equations (reference). The graphs are scaled as in Figure 6.11.

a general nonlinear optimisation method requires the most function and gradient evaluations.

In the second series of experiments summarised in Table 6.13 partially incorrect sensitivities are used which result from numerical sensitivity analysis in connection with discontinuity treatment by the back-tracing based algorithm. NLSSOL and SNOPT in general return inacceptable results. LSQFDN shows higher robustness against the erroneous gradient information. Only the estimate of ρ_{gas} deteriorates slightly, and the number of NLP function and gradient evaluations increases by a factor 2 to 3.

In the final series of tests uncorrected numerical sensitivities are used. Table 6.14 shows the resulting behaviour of the optimisers. LSQFDN returns acceptable

Table 6.12: Parameter identification of immersing pendulum: Correct sensitivities by solution of the reduced consistency equations.

	LSQFDN			NLSSOL			SNOPT		
set	ρ_{gas}	ρ_{liq}	eval	ρ_{gas}	ρ_{liq}	eval	ρ_{gas}	ρ_{liq}	eval
1	1.327	1004	5	0.500	998	4	1.960	1005	13
2	1.327	1004	6	0.500	1007	6	0.500	1007	10
3	1.327	1004	5	1.124	1006	4	0.500	1008	8

Table 6.13: Parameter identification of immersing pendulum: Sensitivities adapted by back-tracing based algorithm.

	LSQFDN			NLSSOL			SNOPT		
set	ρ_{gas}	ρ_{liq}	eval	ρ_{gas}	ρ_{liq}	eval	ρ_{gas}	ρ_{liq}	eval
1	1.373	1004	18	13.39	965.2	9	4.022	986.1	8
2	1.374	1004	10	3.921	996.4	5	0.500	1004	12
3	1.374	1004	10	1.325	1005	4	31.90	917.6	91 [m!]

Table 6.14: Parameter identification of immersing pendulum: Sensitivities without correction.

	LSQFDN			NLSSOL			SNOPT		
set	ρ_{gas}	ρ_{liq}	eval	ρ_{gas}	ρ_{liq}	eval	ρ_{gas}	ρ_{liq}	eval
1	1.389	1003	105 [m!]	4.366	1027	3	1.230	1005	11
2	1.353	1004	78 [c!]	3.328	1005	6	3.363	985.2	18
3	1.471	1003	241 [m!]	2.078	991.3	4	0.500	1200	1 [a!]

results, but it suffers from severe problems due to the wrong sensitivities. From the progress of the optimisation it can be seen that the optimiser requires an excessive number of iterations as it cannot establish convergence. NLSSOL and SNOPT tend to state convergence at suboptimal points.

6.3.4 Parametric Sensitivity Analysis for C3-Splitter

In Section 6.2.2 our algorithms for the computation of consistent initial conditions have been applied to the dynamic model of a C3-splitter. The model DAE contains $n_{\boldsymbol{x}} + n_{\boldsymbol{y}} = 1747$ differential and algebraic equations, and it is of differential index 2. During a typical simulation run several state dependent discontinuities occur.

In this section parametric sensitivity analysis of the state variable trajectories of the C3-splitter with respect to $n_{\boldsymbol{p}} = 8$ model parameters $\boldsymbol{p}_\nu \in \mathbb{R}$, $\nu = 1, \ldots, n_{\boldsymbol{p}}$, is performed. $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$ are used for the modelling of the heat transfer within the condenser **(c)**, $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$ are parameters of the pressure controller **(PC1)**, and $\boldsymbol{p}_6$ and $\boldsymbol{p}_7$ are parameters of the column sump level controller **(LC1)**, cf. Figure 6.3 on page 197. In the scenario considered the temperature controller **(TC1)** fails. Parameter $\boldsymbol{p}_8$ is used to model this malfunction.

Table 6.15 shows a varying effect of omitting the correct transfer of the sensitivity matrices at the discontinuities. Due to the system dimensions the test has been restricted to the sensitivities of some of the states of major interest, i.e., on

Table 6.15: Sensitivity analysis for C3-splitter: Numerical evaluation of the sensitivity functions $[\partial(\text{state variable})/\partial \boldsymbol{p}_\nu](t;\boldsymbol{p})$, $\nu = 1,\ldots,8$, with sensitivity transfer based on solution of the reduced system of consistency equations, and sensitivity analysis without correction. In case of · the values obtained by the two methods generate (almost) identical results, in case of △ the results are slightly different, and in case of ▲ they differ considerably.

state variable	parameters							
	$\boldsymbol{p}_1$	$\boldsymbol{p}_2$	$\boldsymbol{p}_3$	$\boldsymbol{p}_4$	$\boldsymbol{p}_5$	$\boldsymbol{p}_6$	$\boldsymbol{p}_7$	$\boldsymbol{p}_8$
T_1	·	·	·	·	△	△	△	·
T_4	·	·	·	·	·	▲	▲	·
T_{top}	·	·	·	·	·	▲	△	·
P_{top}	·	·	·	·	·	△	△	·
F_{bottom}	△	·	·	▲	▲	▲	▲	△

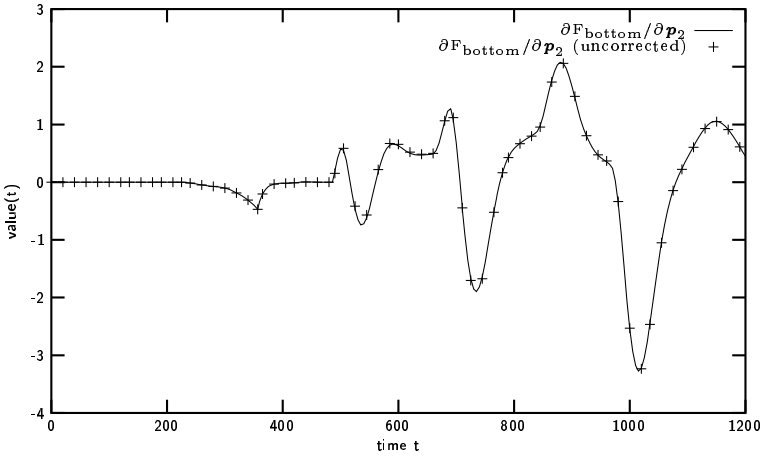

Figure 6.13: Sensitivity analysis for C3-splitter: Numerical evaluation of the sensitivity function $\partial F_{bottom}/\partial \boldsymbol{p}_2$ with and without correction.

the sensitivities of the column temperature at the first tray above the column sump T_1, at the fourth tray T_4, and at the column top T_{top}, on the sensitivities of the column top pressure P_{top}, and on the sensitivities of the molar flow rate F_{bottom} of the product stream leaving the column at the bottom. In most cases the values for the sensitivity functions obtained numerically with and without extended discontinuity treatment are almost identical (an example is plotted in Figure 6.13). In

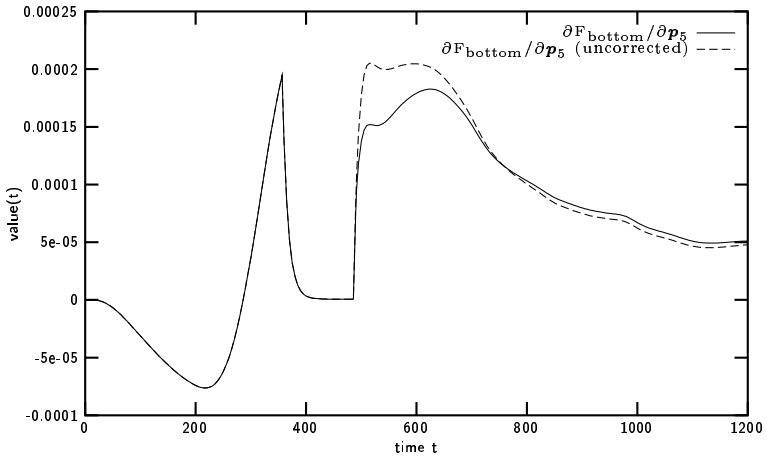

Figure 6.14: Sensitivity analysis for C3-splitter: Numerical evaluation of the sensitivity function $\partial F_{bottom}/\partial \boldsymbol{p}_5$ with and without correction.

some cases minor differences arise, and finally some of the results of the numerical sensitivity analysis differ significantly (see Figure 6.14). In general, the sensitivities with respect to $\boldsymbol{p}_6$ and $\boldsymbol{p}_7$ are affected by the discontinuities, while the sensitivities of the states of interest with respect to $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are not affected. Especially the sensitivities of F_{bottom} require a correct transfer at the discontinuities.

6.3.5 An Optimal Load-Change Policy for a Cryogenic Air Separation Plant (I)

The subject of interest in this section is the load-change of an existing cryogenic air separation plant as described in Section 2.1.2 (cf. Figure 2.2 on page 28). The task is to decrease the load of the plant from 100 [%] to 60 [%] air input. The standard procedure for the load-change effectively takes about one hour. For the computation of optimal control strategies, however, it is necessary to consider an extended prediction horizon in order to account for large time constants in the air separation plant, cf. Section 2.1.2. E.g., [Nijs 96] proposes a prediction horizon which is about two to three times larger than the process response time. For this example a less conservative prediction horizon $[t_0, t_f] := [0, 6000]$ [s] has been chosen. The air separation plant is modelled by a semi-explicit DAE of index 2 (as reported by SPALG), consisting of $n_{\boldsymbol{x}} = 910$ differential and $n_{\boldsymbol{y}} = 3352$ algebraic equations.

For stable operation and for guaranteed product quality it is of utmost importance that several purity restrictions are not violated during the load change. These purity restrictions result in lower and upper bounds on six state variables

(cf. Table 2.2 on page 27). Five constraints refer to product quality, the sixth constraint is a stable operation constraint. The $n_{\boldsymbol{u}} = 5$ control variables (GOX drain, crude Ar drain, Ar condenser turnover, reflux HPC, reflux LPC) describe the positions of the valves **(a)**–**(e)** in Figure 2.2. These control variables are parameterised by $n_{\boldsymbol{p}} = 10$ parameters using global shape functions according to our direct single shooting scheme. The path inequality constraints are discretised on a time grid of 40 equidistant points resulting in[vii] $6 \cdot 2 \cdot 40 = 480$ nonlinear point inequality constraints for the nonlinear programming problem. Furthermore, the controls are constrained to upper and lower bounds on an equidistant grid with 20 points, amounting to $5 \cdot 2 \cdot 20 = 200$ additional nonlinear point inequality constraints. The objective function has been chosen such that the product gain is maximised. However, the primary task is to find a feasible control strategy for this highly complex plant.

In [EKKS 99], [KKES 01] a closely related optimal control problem has been treated using an earlier version of our direct single shooting algorithm integrated into OPTISIM®. This version did not contain an algorithm for the numerically correct treatment of the sensitivities at state and time dependent switching points. Though, the load-change problem has been successfully solved. A detailed inspection of the optimisation history using diagnosis facilities implemented in the course of the presented project shows that during integration only time dependent switching functions have been active. I.e., the switching times were fixed, and thus the sensitivities of the dynamic initialisation problem were identical with the sensitivities of the consistent initialisation problem. As a consequence no special correction of the sensitivities was required.

The initial guess for the optimisation parameters $\boldsymbol{p}$ leads to a breakdown of the air separation process, caused by several state variables severely violating their bounds. In Figure 6.15 the time histories of the relevant purities (cf. Table 2.2) are displayed. After optimisation feasibility of all purities within their lower and upper bounds is achieved, see Figure 6.16. In both graphs the purities are scaled to their admissible ranges given in Table 2.2, and the air input is scaled to [60;100] [%]. The differences between the optimal trajectories displayed in Figure 6.16 and those obtained in [EKKS 99], [KKES 01] can be explained by the usage of a modified and more detailed model for the rectification columns. In relation to our earlier results the optimised product gain has been increased by 23% with respect to [EKKS 99], and by 6% with respect to [KKES 01]. Again, note that the results are not strictly compatible due to the modified dynamic column model.

The above optimisation generates a feasible transition of the air separation plant given a fixed interval in time $t_{\Delta} = 3600\,[\mathrm{s}]$ for the adaptation of the main air intake. The next step is to find the minimum admissible time span for the transition of the main air intake that allows to find controls such that the state

[vii] The states are constrained from below and above. The same holds for the control constraints introduced below.

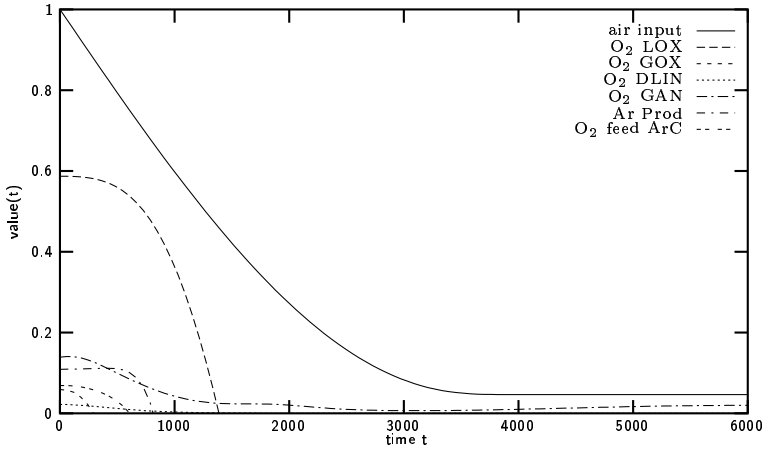

Figure 6.15: Air separation plant: Normalised purities before the optimisation of the parametrised controls. The symbols are explained in Figure 2.2 and Table 2.2.

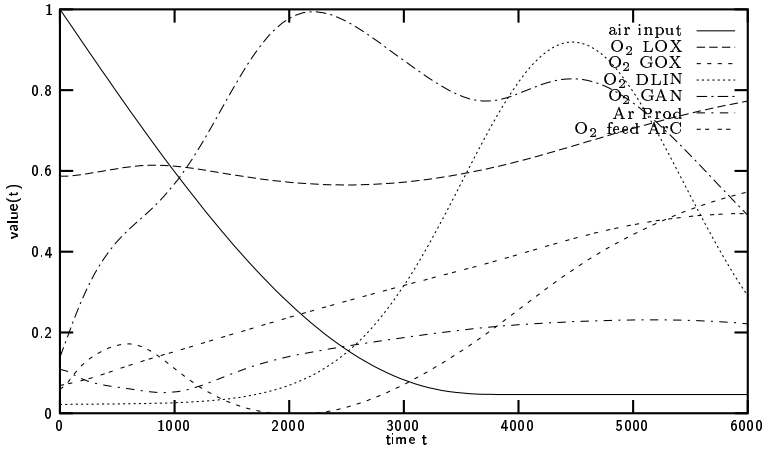

Figure 6.16: Air separation plant: Normalised purities for the optimal parametrised solution maximising the product gain. For an explanation of the symbols see Figure 2.2 and Table 2.2.

constraints are not violated. For this purpose the basic ramp time is subject to optimisation. I.e., the objective is to minimise the interval in which the air intake is driven to its desired value. This is a considerable change in the optimisation problem as the new problem contains a parameter dependent discontinuity at $t = t_\Delta$

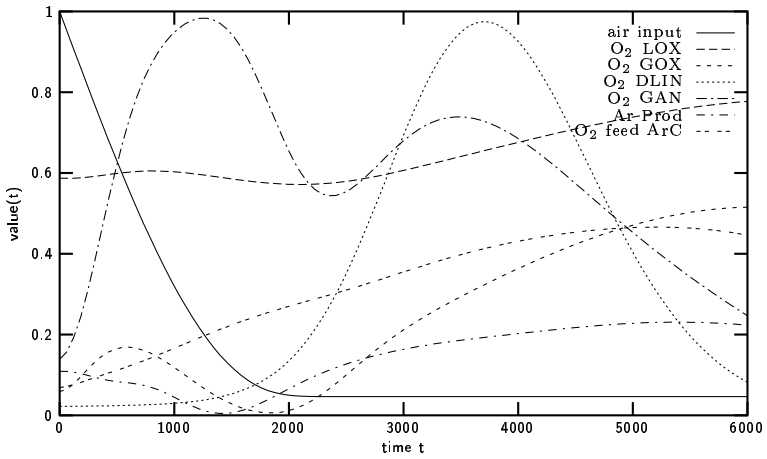

Figure 6.17: Air separation plant: Normalised purities for the optimal parametrised solution with minimum transition time. For an explanation of the symbols see Figure 2.2 and Table 2.2.

when the ramp is switched to its final constant value. According to the theory in such a case consistent initialisation and correct transfer of the sensitivities is generally necessary. Together with the transition time the parameterised optimisation problem includes $n_{\boldsymbol{p}} = 11$ parameters. The nonlinear path constraints are the same as considered above. In order to facilitate the solution of the optimisation problem the feasible set of parameters maximising the product gain for the standard transition time is taken as initial guess. As result of the optimisation we obtain that by employing optimally chosen controls the ramp time can be reduced by 46% from $t_\Delta = 3600\,[\mathrm{s}]$ to $t_\Delta \approx 1960\,[\mathrm{s}]$ without violating the path constraints in a prediction horizon of 6000 [s]. The most important trajectories for the optimised transition are depicted in Figure 6.17. The fast transition is to be payed for with a decrease of 6% in the overall product gain during the entire prediction horizon. In industrial practice this small loss is neglectable.

Again, a closer inspection of the sensitivities with respect to t_Δ shows that at the optimal solution uncorrected sensitivities are identical with the sensitivity functions computed using consistent initialisation and rigorous sensitivity transfer in spite of the parameter dependent discontinuity, cf. Figure 6.18. This behaviour can be explained by the special form of the basic ramp function. The basic ramp is a global $\mathcal{C}^1$-function with zero slope at the discontinuity. According to Eq. (4.45) and Eq. (4.46) the second property can mask off the terms for the corrected sensitivities containing the parametric dependency of the time of discontinuity. Additionally, the smoothness of the ramp can implicate that the state vector at $\lim_{t \nearrow t_\Delta}$ (i.e., immediately before the discontinuity) is consistent with the system at $\lim_{t \searrow t_\Delta}$. In

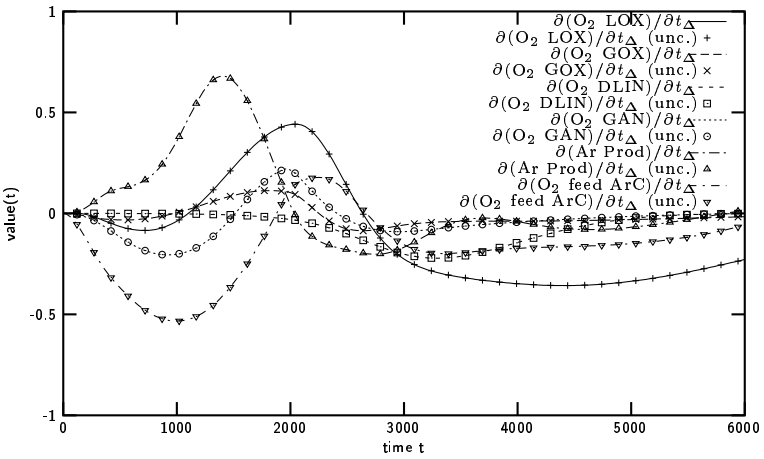

Figure 6.18: Air separation plant: Sensitivities for the optimal parametrised solution with minimum transition time and smooth ramp, with and without correction. $\partial(\mathrm{O_2\ LOX})/\partial t_\Delta$ has been divided by $2.0 \cdot 10^{-7}$, $\partial(\mathrm{O_2\ GOX})/\partial t_\Delta$, $\partial(\mathrm{Ar\ Prod})/\partial t_\Delta$ have been divided by $4.0 \cdot 10^{-6}$, $\partial(\mathrm{O_2\ DLIN})/\partial t_\Delta$, $\partial(\mathrm{O_2\ GAN})/\partial t_\Delta$ have been divided by $2.0 \cdot 10^{-8}$, and $\partial(\mathrm{O_2\ feed\ ArC})/\partial t_\Delta$ has been divided by $6.0 \cdot 10^{-6}$. For an explanation of the symbols see Figure 2.2 and Table 2.2.

this case consistent initialisation has no special effect on the system dynamics, cf., e.g., [KMG 97], [Hins 97].

When the smooth ramp is replaced by a linear ramp the correct sensitivities with respect to the transition time (see Figure 6.19) and the result of an uncorrected sensitivity calculation (see Figure 6.20) differ significantly. Thus in the last sequence of calculations the numerical solution of the optimal control problems (maximum product gain, minimum transition time) is investigated for the linearly ramped process air intake.

If provided with the corrected sensitivities the direct single shooting algorithm successfully solves the maximum product gain problem ($t_\Delta = 3600\,[\mathrm{s}]$ constant). The achieved product gain even improves the result of the computation with the smooth ramp by 3.3%. As can be seen from Figure 6.21 the system dynamics differ considerably from the smooth transition (compare with Figure 6.16). The optimal solution utilises the maximum allowed range in the path constraints, but feasibility is preserved during the entire optimisation horizon of 6000 [s]. Starting from this parameter set optimisation for minimum transition time based on the corrected sensitivities gives $t_\Delta = 2174\,[\mathrm{s}]$. This is about 210 [s] longer than for the smooth ramp. In comparison with the maximum product gain policy for the smooth ramp the gain during the transition reduces by 10%. Shortly after the

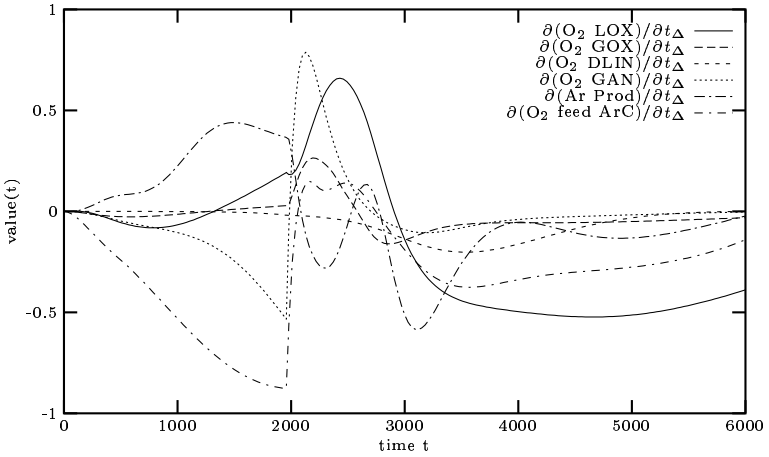

Figure 6.19: Air separation plant: Correct sensitivities with respect to transition time for linear ramp. The sensitivities are scaled as in Figure 6.18.

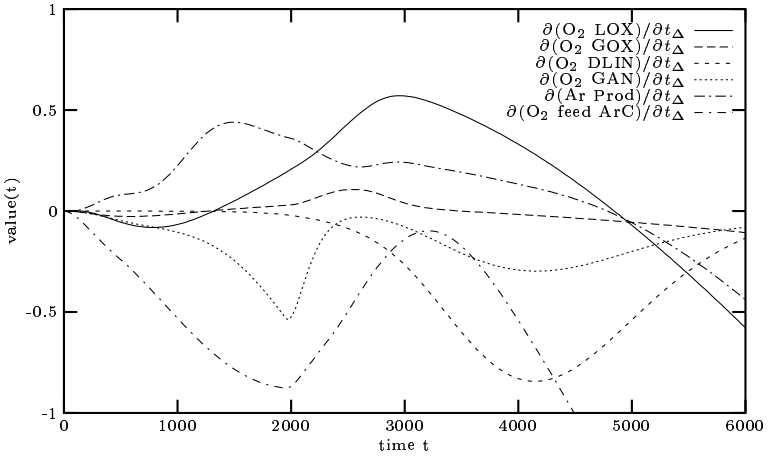

Figure 6.20: Air separation plant: Incorrect sensitivities with respect to transition time for linear ramp due to inappropriate treatment of discontinuity. The sensitivities are scaled as in Figure 6.18.

ramp is finished O_2 GAN violates its upper bound for a small period in time (about 150 [s]). Though the optimiser states feasibility. This behaviour can be explained by the peak of O_2 GAN at the interval in question, cf. Figure 6.22. Due to the discretisation of the path constrains on a mesh with a distance between two consecutive points of about 150 [s] the violation escapes from further treatment.

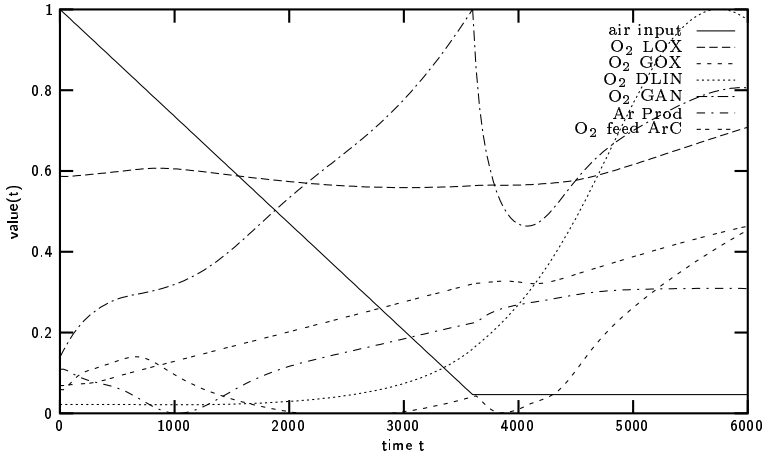

Figure 6.21: Air separation plant: Normalised purities for the optimal parametrised solution with maximum product gain and linear ramp. For an explanation of the symbols see Figure 2.2 and Table 2.2.

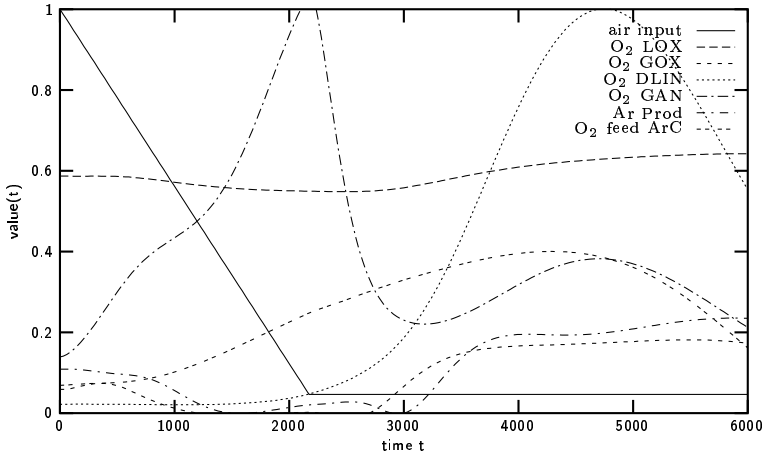

Figure 6.22: Air separation plant: Normalised purities for the optimal parametrised solution with minimum transition time and linear ramp. For an explanation of the symbols see Figure 2.2 and Table 2.2.

As a countermeasure the sampling grid for the path constraints could be refined. Additionally, the objective could be augmented by terms penalising the integral violation of the path constraints, cf. Remark 2.2 on page 38, and Remark 2.6 on page 55. However, from a practical point of view the violation is tolerable.

When the gradient information for the parameterised optimal control problem is generated using uncorrected numerical sensitivity information SNOPT terminates successfully in the maximum product gain scenario with nearly the same set of optimised parameters as above (the final value of the objective function increases marginally by 0.05%). As the transition time is fixed in this optimal control problem (nearly) the same sensitivities are generated as with the extended discontinuity treatment. Therefore the result is as expected. When minimising the transition time, however, the sensitivity information is considerably affected by the parameter dependent discontinuity at the end of the main ramp (cf. Figure 6.19 and Figure 6.20). SNOPT aborts prematurely with the diagnosis that the set of optimisation variables cannot be further improved. The final value of t_Δ is 2635 [s], which is 21% worse than the result of the optimisation with correct sensitivity information.

6.3.6 An Optimal Load-Change Policy for a Cryogenic Air Separation Plant (II)

As the final test for our direct single shooting algorithm the optimal load-change for the complex cryogenic air separation plant introduced in Section 6.2.3 is considered. The mathematical model of this air separation plant in OPTISIM® is set up by an index-2 DAE with $n_{\boldsymbol{x}} + n_{\boldsymbol{y}} = 1832 + 7292 = 9124$ differential and algebraic equations. This is about 2.6 times the size of the DAE used to model the air separation plant in the preceding Section 6.3.5. Thus this model constitutes the basis of the largest optimal control problem from chemical engineering application that has been successfully solved by our algorithm up to now. Additionally, due to the intricate plant characteristics also from the chemical engineering point of view the control of this air separation plant is more difficult than the control of the plant considered in the previous Section 6.3.5.

The load-change task is to decrease the load of the plant from 100 [%] to 60 [%] air input within $t_\Delta = 5400$ [s] (1.5 hours). During the load-change $n_{\boldsymbol{u}} = 13$ control variables are to be chosen such that $2 \cdot 14 = 28$ nonlinear path inequality constraints[(viii)] are not violated. $2 \cdot 5 = 10$ of these path inequality constraint refer to control variables.

The optimal control problem covers a prediction horizon of $[t_0, t_\mathrm{f}] := [0, 10800]$ [s] (3 hours). By sampling on an equidistant mesh of 30 points in $[t_0, t_\mathrm{f}]$ the path inequality constraints are transformed into $2 \cdot 14 \cdot 30 = 840$ point inequality constraints. The control variables are parameterised by $n_{\boldsymbol{p}} = 29$ parameters.

Remark 6.5:
In this example the dense sensitivity matrix $\partial[\boldsymbol{x}^T, \boldsymbol{y}^T]^T / \partial \boldsymbol{p}$ contains $9124 \cdot 29 = 264596$ entries. Thus solely for the integration of the sensitivity matrix during a single integration

[(viii)] I.e., 14 state and control variables are constrained by a lower and an upper bound, compare Table 2.2 on page 27.

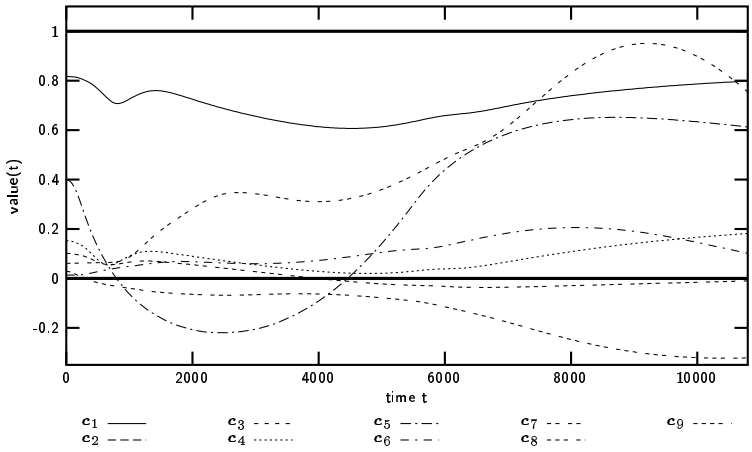

Figure 6.23: Complex air separation plant: Normalised state constraints at initial parameter set. The bold horizontal lines at ordinate 0 and 1 indicate the bounds. The constrained quantities $\boldsymbol{c}_\mu$, $\mu = 1, \ldots, 9$, are both bounded from below and above.

step with lowest integration order 1 (backward Euler's method) roughly[ix] $5 \cdot 264596 \cdot 8 \approx 10.1$ [MB] of memory are accessed (using double precision arithmetics). For an integration step with highest order 5 about $13 \cdot 264596 \cdot 8 \approx 26.2$ [MB] of memory are accessed. ◊

The initial guess for the optimisation parameters leads to a violation of the constraints as depicted in Figure 6.23. In a first optimisation step feasible controls are obtained using the extended feasibility phase of SNOPT. In Figure 6.24 the corresponding trajectories for the constrained states are plotted.

After this a second optimisation is started in order to minimise the nitrogen intake into the argon refinement section during the transition. Minimising this nitrogen intake aims at increasing the robustness of the load-change policy by gaining backup for the nitrogen constraint at the argon transition, cf. Section 2.1.2. As initial estimate the feasible set of parameters from the previous optimisation is used. The trajectories of the state constraints for the result of this optimisation are shown in Figure 6.25. The optimised control strategy decreases the nitrogen intake into the argon refinement section by 25%.

[ix] 1 [MB] = 1024 [kB] = 1048576 [Bytes]

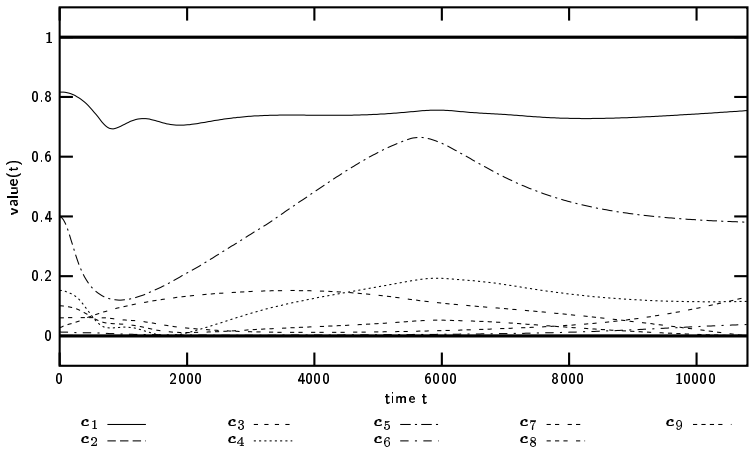

Figure 6.24: Complex air separation plant: Normalised state constraints at the first feasible set of optimisation parameters, cf. Figure 6.23.

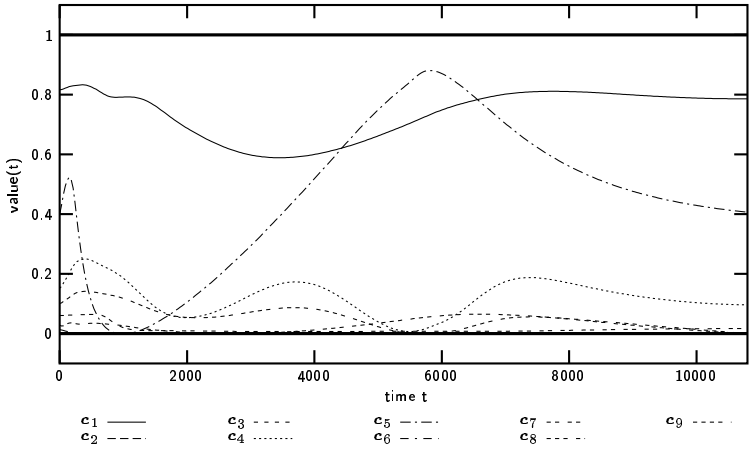

Figure 6.25: Complex air separation plant: Normalised state constraints at the optimised parameter set with minimum nitrogen intake into the argon refinement section, cf. Figure 6.23.

Chapter 7

Summary and Outlook

I have learned to use the word "impossible" with the greatest caution.

Wernher von Braun

Starting from a concrete problem from industrial application a new concept for nonlinear model-based predictive control has been suggested. This concept extends and specifies currently existing NMPC approaches. A core part of the concept consists in the repeated calculation of parameterised open-loop optimal controls by a tailored direct single shooting algorithm. This direct single shooting algorithm has been integrated into the industrial simulation and optimisation environment OPTISIM® of the Linde AG, Linde Engineering Division. The models of the processes of primary interest are given by very large-scale systems of coupled differential and algebraic equations of index two with state and time dependent discontinuities.

In order to tackle with the problem of limited computational time available for the calculation of updated optimal control policies in a real-time setting an algorithm for the fast computation of corrections to a previously determined control policy has been proposed. The algorithm determines corrections which counteract a predicted violation of the nonlinear constraints of the optimal control problem in the presence of disturbances modelled by disturbance parameters. An advantage of the algorithm is that it uses data which may be available from the previous solution of the open-loop optimal control problem. Especially, it requires information on the parametric sensitivity functions of the system trajectory.

A central part of the tailored direct single shooting method is an algorithm for the integration and parametric sensitivity analysis of very large-scale semi-explicit index-2 DAEs. The numerically determined sensitivity functions also constitute a basic piece of information required by the newly proposed algorithm for the fast calculation of corrections to a nominal control strategy. In this work the algorithm for numerical integration and sensitivity analysis of DAEs incorporated into OPTISIM® has been extended with methods for the correct numerical treatment of state and time dependent discontinuities present in the process models of

interest.

In the context of DAEs the correct treatment of discontinuities leads to the problem of generating consistent initial conditions. This problem is numerically demanding especially for large-scale higher index DAEs. Based on a review of existing methods two algorithms for the numerical calculation of consistent initial conditions for very large-scale semi-explicit index-2 DAEs have been developed and implemented:

- In the first approach back-tracing is employed in order to obtain numerically consistent initial conditions by utilisation of special properties of the BDF integrator method.
- In the second approach consistent initial conditions are calculated in a strict sense by solution of the reduced consistency equations. In order to obtain a regular system of reduced consistency equations this rigorous approach employs structural analysis of a given DAE (Pantelides' Algorithm) and a newly developed method for the automatic generation of appropriate transition conditions. For the approximation of total time derivatives Leimkuhler's method is employed. The algorithm only requires numerical evaluations of the DAE and of its Jacobian, and the structural pattern of the Jacobian.

The transfer of parametric sensitivities at discontinuities in DAEs is closely related to the determination of consistent initial conditions. Therefore, corresponding to our algorithms for the numerical computation of consistent initial conditions two algorithms for the transfer of sensitivities at discontinuities have been implemented. The first algorithm is based on back-tracing and numerical differentiation of the jump function. The second, rigorous algorithm is based on the solution of the reduced system of consistency equations for the sensitivity DAE. While the back-tracing based approach has shown applicable to smaller problems only, the capabilities of the rigorous approach have been demonstrated by the successful solution of demanding open-loop optimal control problems from chemical engineering.

A task left to be done is a comparison of the new method for the fast correction of controls with the Newton-type controller reviewed in Section 2.6.3. An interesting point are the similarities between both methods in spite of the different ways that have lead to their development.

Then the next step is – with the proposed NMPC concept, the tailored direct single shooting method, and the algorithm for a fast correction of the controls at hand – in tying the elements together in order to perform the step from open-loop optimal control to closed-loop optimal control of large-scale processes. The missing link is a suitable algorithm providing an estimate of the process state consistent with the large-scale dynamic process model as initial conditions to the open-loop optimal control algorithm.

Chapter 8

Appendix

Algorithms

Definitions

Theorems

Lemmas

Propositions

Index

Bibliography

[ABES 93] Andrzejewski, T.; Bock, H.G.; Eich, E.; von Schwerin, R.:
Recent Advances in the Numerical Integration of Multibody Systems.
in: *Advanced Multibody System Dynamics: Simulation and Software Tools*, Schiehlen, W. (editor),
Kluwer Academic Publishers, Dordrecht 1993, pp. 127-151

[ABQRW 99] Allgöwer, F.; Badgwell, T.A.; Qin, S.J.; Rawlings, J.B.; Wright, S.J.:
Nonlinear Model Predictive Control.
in: *Advances in Control: Highlights of ECC '99*, Frank, P.M. (editor),
Springer, London 1999, pp. 391-449

[AEA 93] AEA Technology:
Harwell Subroutine Library.
Harwell Subroutine Library Specifications, Release **11**, Vol. **1-2**, AEA Technology, Harwell 1993

[AFS 00] Alberts, J.B.; Fabien, B.C.; Storti, D.W.:
Consistent Initial Conditions for Differential-Algebraic Systems by Perturbation Analysis.
submitted to: *The Journal of the Franklin Institute*
`http://faculty.washington.edu/jalberts/`

[AHM 98] Abel, O.; Helbig, A.; Marquardt, W.:
Nichtlineare prädiktive Regelung bei unsicheren Prozeßmodellen.
Schlußbericht zum BMBF-Projekt „Nichtlineare Dynamik in der Chemischen Technik“ (BMBF 03D0020B/5), Lehrstuhl für Prozeßtechnik, RWTH Aachen, Aachen, März 1998

[Albe 99] Alberts, J.B.:
Contributions to Stability Analysis and Numerical Solution of Differential-Algebraic Systems.
Ph.D. thesis, University of Washington, 1999
`http://faculty.washington.edu/jalberts/`

[AlBo 95] Alamir, M.; Bornard, G.:
Stability of a Truncated Infinite Constrained Receding Horizon Scheme: the General Discrete Nonlinear Case.
Automatica **31**, 9 (1995), pp. 1353-1356

[AlDo 97] Allgöwer, F.; Doyle, F.J.:
Nonlinear Process Control: Which Way to the Promised Land?
Proceedings of the Fifth International Conference of Chemical Process Control, Tahoe City, California (January 7 – 12, 1996), Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **93**, 316 (1997), pp. 24-45

[Allg 97] Allgor, R.J.:
Modeling and Computational Issues in the Development of Batch Processes.
Ph.D. thesis, Massachusetts Institute of Technology, June 1997
`http://yoric.mit.edu/PIB/barton-rep.html`

[AlZh 00] Allgöwer, F.; Zheng, A. (eds.):
Nonlinear Model Predictive Control.
Proceedings of the International Symposium on Nonlinear Model Predictive Control – Assessment and Future Directions, Ascona, Switzerland (June 3-5, 1998),
Birkhäuser, Basel 2000

[AmMa 94] Amodio, P.; Mazzia, F.:
Boundary Value Methods for the Solution of Differential-Algebraic Equations.
Numerische Mathematik **66** (1994), pp. 411-421

[AmMa 97] Amodio, P.; Mazzia, F.:
Numerical Solution of Differential Algebraic Equations and Computation of Consistent Initial/Boundary Conditions.
Journal of Computational and Applied Mathematics **87**, 1 (1997), pp. 135-146

[AmMa 98] Amodio, P.; Mazzia, F.:
An Algorithm for the Computation of Consistent Initial Values for Differential-Algebraic Equations.
Numerical Algorithms **19**, 1-4 (1998), pp. 13-23

[Arno 95] Arnold, M.:
Applying BDF to Quasilinear Differential-Algebraic Equations of Index 2: A Perturbation Analysis.
University of Rostock, Department of Mathematics, preprint 95/13, 1995
`http://www.op.dlr.de/~marnold/publ.html`

[Arno 97] Arnold, M.:
Impasse Points and Descriptor Systems with Unilateral Constraints.

in: *Application and Modelling in Simulation, Vol.* ***6***, Sydow, A. (editor),
Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Berlin (August 1997), Wissenschaft & Technik Verlag, Berlin 1997, pp. 189-194
`http://www.op.dlr.de/~marnold/ps/imacs97.ps`

[Asch 90] Ascher, U.M.:
Differential-Algebraic Problems and Semiconductor Device Simulation.
in: *Mathematical Modelling and Simulation of Electrical Circuits and Semiconductor Devices*, Bank, R.E.; Bulirsch, R.; Merten, K. (editors),
Proceedings of a conference held at the Math. Forschungsinst., Oberwolfach (October 30 – November 5, 1988),
International Series of Numerical Mathematics **93** (1990), pp. 201-213

[Asch 99] Ascher, U.M.:
DAEs that Should Not Be Solved.
Proceedings of the IMA workshop on Dynamics of Algorithms, University of Minnesota (November 1997), de la Llave, R.; Petzold, L.R.; Lorenz, J. (editors),
IMA Volumes in Mathematics and its Applications **118** (1999), pp. 55-68
`http://www.cs.ubc.ca/nest/scv/group-info/professors/-ascher/research.html`

[BAFG 98] Barton, P.I.; Allgor, R.J.; Feehery, W.F.; Galán, S.:
Dynamic Optimization in a Discontinuous World.
Industrial & Engineering Chemistry Research **37**, 3 (1998), pp. 966-981

[BaGa 00] Barton, P.I.; Galán, S.:
Linear DAEs with Nonsmooth Forcing.
preprint submitted to SIAM Journal on Scientific Computing
`http://yoric.mit.edu/download/Papers/NonsmoothDAEs.ps`

[BaPa 94] Barton, P.I.; Pantelides, C.C.:
Modeling of Combined Discrete/Continuous Processes.
AIChE Journal **40**, 6 (1994), pp. 966-979

[BaPk 97] Barton, P.I.; Park, T.:
Analysis and Control of Combined Discrete/Continuous Systems: Progress and Chellenges in the Chemical Processing Industries.

Proceedings of the Fifth International Conference of Chemical Process Control, Tahoe City, California (January 7 – 12, 1996), Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **316**, 93 (1997), pp. 102-114

[Bart 00] Barton, P.I.:
The Equation Oriented Strategy for Process Flowsheeting.
preprint, March 2000
`http://yoric.mit.edu/abacuss2/Lecture1.pdf`

[Bart 92] Barton, P.I.:
The Modelling and Simulation of Combined Discrete/Continuous Processes.
Ph.D. thesis, Imperial College of Science, Technology and Medicine, London, 1992
`http://yoric.mit.edu/PIB/barton-rep.html`

[BBAV 02] Balsa Canto, E.; Banga, J.R.; Alonso, A.A.; Vassiliadis, V.S.:
Restricted Second Order Information for the Solution of Optimal Control Problems Using Control Vector Parameterization.
Journal of Process Control **12**, 2 (2002), pp. 243-255

[BBBB 01] Binder, T.; Blank, L.; Bock, H.G.; Bulirsch, R.; Dahmen, W.; Diehl, M.; Kronseder, T.; Marquardt, W.; Schlöder, J.P.; von Stryk, O.:
Introduction to Model Based Optimization of Chemical Processes on Moving Horizons.
in: *Online Optimization of Large Scale Systems*, Grötschel, M.; Krumke, S.O. (editors),
Springer, Heidelberg 2001, pp. 295-339

[BBDM 01] Binder, T.; Blank, L.; Dahmen, W.; Marquardt, W.:
Multiscale Concepts for Moving Horizon Estimation.
in: *Online Optimization of Large Scale Systems*, Grötschel, M.; Krumke, S.O. (editors),
Springer, Heidelberg 2001, pp. 341-361

[BBDM 98] Binder, T.; Blank, L.; Dahmen, W.; Marquardt, W.:
Towards Multiscale Dynamic Data Reconciliation.
in: *Nonlinear Model Based Process Control, NATO ASI Series Vol.* ***E353***, Berber, R.; Kravaris, C. (editors),
Kluwer Academic Publishers, Dordrecht 1998, pp. 623-556
`http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/-Preprint-98-7.html`

[BBiS 99] Bloss, K.F.; Biegler, L.T.; Schiesser, W.E.:
Dynamic Process Optimization through Adjoint Formulations and

Constraint Aggregation.
Industrial & Engineering Chemistry Research **38**, 2 (1999), pp. 421-432

[BBKS 83] Baldus, H.; Baumgärtner, K.; Knapp, H.; Streich, M.:
Verflüssigung und Trennung von Gasen.
in: *Chemische Technologie Band 3: Anorganische Technologie II*, Winnacker, K.; Küchler, L.; (editors),
Carl Hanser Verlag, München 1983, 4th edition

[BBMP 90] Bachmann, R.; Brüll, L.; Mrziglod, Th.; Pallaske, U.:
On Methods for Reducing the Index of Differential Algebraic Equations.
Computers and Chemical Engineering **14**, 11 (1990), pp. 1271-1273

[BBR 01] van Beek, D.A.; Bos, V.; Rooda, J.E.:
Declaration of Unknowns in Hybrid System Specification.
preprint submitted to ACM Transactions on Modeling and Computer Simulation
`http://se.wtb.tue.nl/~vanbeek/pub/fr_pub_intro.htm`

[BCCGH 92] Bischof, C.; Carle, A.; Corliss, G.; Griewank, A.; Hovland, P.:
ADIFOR – Generating Derivative Codes from Fortran Programs.
Scientific Programming **1**, 1 (1992), pp. 11-29

[BCH 97] Betts, J.T.; Carter, M.J.; Huffman, W.P.:
Software for Nonlinear Optimization.
MEA-LR-083 R1, Boeing Information & Support Services, Seattle (1997)

[BCP 96] Brenan, K.E.; Campbell, S.L.; Petzold, L.R.:
Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.
SIAM Classics in Applied Mathematics **14**, SIAM, Philadelphia 1996

[BeHu 97] Betts, J.T.; Huffman, W.P.:
Sparse Optimal Control Software SOCS.
MEA-LR-085, Boeing Information & Support Services, Seattle (1997)

[Bell 97] Bell, M.:
Robust Optimal Receding Horizon Control.
Ph.D. thesis, Centre for Process Systems Engineering and Department of Chemical Engineering and Chemical Technology, Imperial College of Science, Technology and Medicine, University of London, March 1997

[Bellm 65] Bellman, R.E.:
Dynamic Programming.
Princeton University Press, Princeton 1965, 4th printing

[Bellm 71] Bellman, R.E.:
Introduction to the Mathematical Theory of Control Processes: Volume II: Nonlinear Processes.
Academic Press, New York 1971

[Benn 96] Bennet, S.:
A Brief History of Automatic Control.
IEEE Control Systems Magazine **16**, 3 (1996), pp. 17-25

[Bequ 91a] Bequette, B.W.:
Nonlinear Control of Chemical Processes: A Review.
Industrial & Engineering Chemistry Research **30**, 7 (1991), pp. 1391-1413

[Bequ 91b] Bequette, B.W.:
Nonlinear Predictive Control Using Multi-rate Sampling.
Canadian Journal of Chemical Engineering **69** (1991), pp. 136-143

[BeSe 92] Bestle, D.; Seybold, J.:
Sensitivity Analysis of Constrained Multibody Systems.
Archive of Applied Mechanics **62** (1992), pp. 181-190

[Bett 98] Betts, J.T.:
Survey of numerical methods for trajectory optimization.
AIAA Journal of Guidance, Control, and Dynamics **21**, 2 (1998), pp. 193-207

[BGH 72] Brayton, R.K.; Gustavson, F.G.; Hachtel, G.D.:
A New Efficient Algorithm for Solving Differential-Algebraic Systems Using Implicit Backward Differentiation Formulas.
Proceedings of the IEEE **60**, 1 (1972), pp. 98-108

[BGW 90] Bitmead, R.R.; Gevers, M.; Wertz, V.:
Adaptive Optimal Control: The Thinking Man's GPC.
Prentice-Hall, Englewood Cliffs, New Jersey 1990

[BHK 94] Buchauer, O.; Hiltmann, P.; Kiehl, M.:
Sensitivity Analysis of Initial-Value Problems with Application to Shooting Techniques.
Numerische Mathematik **67** (1994), pp. 151-159

[BHP 94] Brown, P.N.; Hindmarsh, A.C.; Petzold, L.R.:
Using Krylov Methods in the Solution of Large-Scale Differential-Algebraic Systems.
SIAM Journal on Scientific Computing **15**, 6 (1994), pp. 1467-1488

[BHP 98] Brown, P.N.; Hindmarsh, A.C.; Petzold, L.R.:
Consistent Initial Condition Calculation for Differential-Algebraic Systems.
SIAM Journal on Scientific Computing **19**, 5 (1998), pp. 1495-1512
`http://www.siam.org/journals/sisc/19-5/28999.html`
`http://www.engineering.ucsb.edu/~cse/publica.html`

[Bieg 88] Biegler, L.T.:
On the Simultaneous Solution and Optimization of Large Scale Engineering Systems.
Computers and Chemical Engineering **12**, 5 (1988), pp. 357-369

[BiRa 91] Biegler, L.T.; Rawlings, J.B.:
Optimization Approaches to Nonlinear Model Predictive Control.
in: *Chemical Process Control CPC-IV*, Arkun, Y.; Ray, W.H. (editors),
Proceedings of the Fourth International Conference on Chemical Process Control, Padre Island, Texas (February 17 – 22, 1991), pp. 543-571

[Björ 96] Björk, Åke:
Numerical Methods for Least Squares Problems.
SIAM, Philadelphia 1996, 2nd printing

[BMP 91a] Bulirsch, R.; Montrone, F.; Pesch, H.J.:
Abort Landing in the Presence of a Windshear as a Minimax Optimal Control Problem. Part 1: Necessary Conditions.
Journal of Optimization Theory and Applications **70**, 1 (1991), pp. 1-23

[BMP 91b] Bulirsch, R.; Montrone, F.; Pesch, H.J.:
Abort Landing in the Presence of a Windshear as a Minimax Optimal Control Problem. Part 2: Multiple Shooting and Homotopy.
Journal of Optimization Theory and Applications **70**, 2 (1991), pp. 223-254

[BMS 94] Burr, P.S.; Moser, B.; Schwab, J.W.:
An Object Request Broker Architecture for Large Technical Systems and its Appliation to Redesign of the OPTISIM Process Simulator.
Linde Reports on Science and Technology **54** (1994), pp. 65-68

[Bock 87] Bock, H.G.:
Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen.
Brieskorn, E.; Frehse, J.; Hildebrandt, St.; Hirzebruch, F.; Harder, G.; Klingenberg, W.; Leis, R.; Lieb, I.; Peschl, E.; Unger, H.; Vogel, W.; Werner, H. (editors),
Bonner Mathematische Schriften Nr. **183**, Bonn, 1987

[BoPl 84] Bock, H.G.; Plitt, K.J.:
A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems.
in: *A Bridge between Control Science and Technology, Vol. **3***, Gertler, J. (editor),
Proceedings of the 9th trienniel World Congress of IFAC, Budapest, Hungary (July 2-6, 1984), pp. 1603-1608

[BrHo 75] Bryson, A.E.; Ho, Y.-C.:
Applied Optimal Control.
John Wiley & Sons, New York 1975, revised printing

[BrPa 92] Brüll, L.; Pallaske, U.:
On Differential Algebraic Equations with Discontinuities.
Zeitschrift für Angewandte Mathematik und Physik **43** (1992), pp. 319-327

[BrTr 96] Brugnano, L.; Trigiante, D.:
Convergence and Stability of Boundary Value Methods for Ordinary Differential Equations.
Journal of Computational and Applied Mathematics **66** (1996), pp. 97-109

[BSBS 00] Buss, M.; von Stryk, O.; Bulirsch, R.; Schmidt, G.:
Towards Hybrid Optimal Control.
at – Automatisierungstechnik **48**, 9 (2000), pp. 448-459

[BSS 95] Bock, H.G.; Schlöder, J.P.; Schulz, V.H.:
Numerik großer Differentiell-Algebraischer Gleichungen: Simulation und Optimierung.
in: *Prozeßsimulation*, Schuler, H. (editor),
VCH, Weinheim 1995, pp. 35-80

[Buli 71] Bulirsch, R.:
Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung.
presented at the Carl-Cranz-Gesellschaft e.V., DLR, Oberpfaffenhofen, 1971

Reprints: Mathematisches Institut, Technische Universität München (1985, 1993).

[BüMa 00] Büskens, C.; Maurer, H.:
SQP-Methods for Solving Optimal Control Problems with Control and State Constraints: Adjoint Variables, Sensitivity Analysis and Real-Time Control.
Schwerpunktprogramm der Deutschen Forschungsgemeinschaft "Echtzeitoptimierung Großer Systeme", Preprint **00-3**
`http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/-Preprint-00-3.html`

[BüMa 01a] Büskens, C.; Maurer, H.:
Sensitivity Analysis and Real-Time Optimization of Parametric Nonlinear Programming Problems.
in: *Online Optimization of Large Scale Systems*, Grötschel, M.; Krumke, S.O. (editors),
Springer, Heidelberg 2001, pp. 3-16

[BüMa 01b] Büskens, C.; Maurer, H.:
Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Nonlinear Programming Problems.
in: *Online Optimization of Large Scale Systems*, Grötschel, M.; Krumke, S.O. (editors),
Springer, Heidelberg 2001, pp. 57-68

[BüMa 98] Büskens, C.; Maurer, H.:
Real-Time Control of an Industrial Robot Using Nonlinear Programming Methods.
Schwerpunktprogramm der Deutschen Forschungsgemeinschaft "Echtzeitoptimierung Großer Systeme", Preprint **98-12**
`http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/-Preprint-98-12.html`

[Burr 91] Burr, P.S.:
The on-line Optimization of an Olefin Plant Complex with the OPTISIM equation-oriented Simulator.
in: *OPTISIM Process Simulation and Optimization System*, Linde AG, Process Engineering and Contracting Division, Höllriegelskreuth 1991, pp. 1-6

[Burr 93] Burr, P.S.:
The Design of Optimal Air Separation and Liquefaction Processes with the OPTISIM equation-oriented Simulator and its Application to on-line and off-line plant Optimization.
presented at the AIChE Spring National Meeting, Houston, Texas,

April 1991
AIChE Symposium Series **89**, 294 (1993), pp. 1-7

[Büsk 98] Büskens, C.:
Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustandsbeschränkungen.
Ph.D. thesis, Westfälische Wilhelms-Universität Münster, 1998

[Büsk 99] Büskens, C.:
private communications, 1999

[ByHi 87] Byrne, G.D.; Hindmarsh, A.C.:
Stiff ODE Solvers: A Review of Current and Coming Attractions.
Journal of Computational Physics **70** (1987), pp. 1-62

[ByPo 88] Byrne, G.D.; Ponzi, P.R.:
Differential-Algebraic Systems, Their Applications and Solutions.
Computers and Chemical Engineering **12**, 5 (1988), pp. 377-382

[CaGe 95] Campbell, S.L.; Gear, C.W.:
The Index of General Nonlinear DAEs.
Numerische Mathematik **72**, 2 (1995), pp. 173-196

[CaGr 95] Campbell, S.L.; Griepentrog, E.:
Solvability of General Differential Algebraic Equations.
SIAM Journal on Scientific Computing **16**, 2 (1995), pp. 257-270

[CaMo 94] Campbell, S.L.; Moore, E.:
Progress on a General Numerical Method for Nonlinear Higher Index DAEs II.
Circuits, Systems and Signal Processing **13**, 2-3 (1994), pp. 123-138

[CaMo 95] Campbell, S.L.; Moore, E.:
Constraint Preserving Integrators for General Nonlinear Higher Index DAEs.
Numerische Mathematik **69**, 4 (1995), pp. 383-399

[Camp 83] Campbell, S.L.:
Consistent Initial Conditions for Singular Nonlinear Systems.
Circuits, Systems and Signal Processing **2**, 1 (1983), pp. 45-55

[CaSt 85] Caracotsios, M.; Stewart, W.E.:
Sensitivity Analysis of Initial Value Problems with Mixed ODEs and Algebraic Equations.
Computers and Chemical Engineering **9**, 4 (1985), pp. 359-365

[CeEl 93] Cellier, F.E.; Elmqvist, H.:
Automated Formula Manipulation Supports Object-Oriented Continuous-System Modeling.
IEEE Control Systems Magazine **13**, 2 (1993), pp. 28-38

[Cell 79] Cellier, F.E.:
Combined Continuous/Discrete System Simulation by Use of Digital Computers: Techniques and Tools.
Ph.D. thesis, ETH Zürich, 1979

[ChAl 96] Chen, H.; Allgöwer, F.:
A Quasi-Infinite Horizon Nonlinear Predictive Control Scheme for Stable Systems: Application to a CSTR.
IFA Technical Report **AUT-96-30**
`http://www.aut.ee.ethz.ch/research/publications/-publications.msql`

[ChAl 98] Chen, H.; Allgöwer, F.:
Nonlinear Model Predictive Control Schemes with Guaranteed Stability.
in: *Nonlinear Model Based Process Control, NATO ASI Series Vol.* ***E353***, Berber, R.; Kravaris, C. (editors),
Kluwer Academic Publishers, Dordrecht 1998, pp. 465-494

[ChSt 81] Chen, H.-S.; Stadtherr, M.A.:
A Modification of Powell's Dogleg Method for Solving Systems of Nonlinear Equations.
Computers and Chemical Engineering **5**, 3 (1981), pp. 143-150

[CKY 96] Campbell, S.L.; Kelley, C.T.; Yeomans, K.D.:
Consistent Initial Conditions for Unstructured Higher Index DAEs: A Computational Study.
Proceedings of Computational Engineering in Systems Applications, Lille, France (1996), pp. 416-421
`http://www4.ncsu.edu/eos/users/s/slc/www/RESEARCH/-NAAA.html`

[CLP 00] Cao, Y.; Li, S.; Petzold, L.R.:
Adjoint Sensitivity Analysis for Differential-Algebraic Equations: Part II, Numerical Solution.
preprint, Department of Computer Science, University of California, Santa Barbara, November 6, 2000,
`http://www.engineering.ucsb.edu/~cse/Files/-adjointII00.pdf`

[CLPS 00] Cao, Y.; Li, S.; Petzold, L.R.; Serban, R.:
Adjoint Sensitivity Analysis for Differential-Algebraic Equations: Part I, The Adjoint DAE System.
preprint, Department of Computer Science, University of California, Santa Barbara, November 6, 2000,
`http://www.engineering.ucsb.edu/~cse/Files/-adjointI00.pdf`

[CMSW 79] Cline, A.K.; Moler, C.B.; Stewart, G.W.; Wilkinson, J.H.:
An Estimate for the Condition Number of a Matrix.
SIAM Journal on Numerical Analysis **16**, 2 (1979), pp. 368-375

[CMZ 96] Campbell, S.L.; Moore, E.; Zhong, Y.:
Constraint Preserving Integrators for Unstructured Higher Index DAEs.
Zeitschrift für Angewandte Mathematik und Mechanik **76** (1996), pp. 83-86
`http://www4.ncsu.edu/eos/users/s/slc/www/RESEARCH/-NAAA.html`

[Cobb 83] Cobb, D.:
A Further Interpretation of Inconsistent Initial Conditions in Descriptor-Variable Systems.
IEEE Transactions on Automatic Control **AC-28**, 9 (1983), pp. 920-922

[CPR 74] Curtis, A.R.; Powell, M.J.D.; Reid, J.K.:
On the Estimation of Sparse Jacobian Matrices.
IMA Journal of Applied Mathematics **13** (1974), pp. 117-119

[CuRe 74] Curtis, A.R.; Reid, J.K.:
The Choice of Step Lengths When Using Differences to Approximate Jacobian Matrices.
IMA Journal of Applied Mathematics **13** (1974), pp. 121-126

[CVSB 01] Costa Jr., E.F.; Vieira R.C.; Secchi, A.R.; Biscaia Jr., E.C.:
Automatic Structural Characterization of DAE Systems.
in: *Computer-Aided Chemical Engineering **9***, Gani, R.; Jørgensen, S.B. (editors),
Proceedings of the European Symposium on Computer Aided Process Engineering (ESCAPE-11), Kolding, Denmark (May 27 – 30, 2001),
Elsevier Science, Amsterdam 2001, pp. 123-128

[DeBo 94] Deuflhard, P.; Bornemann, F.:
Numerische Mathematik II: Integration gewöhnlicher Differential-

gleichungen.
de Gruyter, Berlin 1994

[DeSc 96] Dennis, J.E.; Schnabel, R.B.:
Numerical Methods for Unconstrained Optimization and Nonlinear Equations.
SIAM Classics in Applied Mathematics **16**, SIAM, Philadelphia 1996

[Dieh 01] Diehl, M.:
Real-Time Optimization for Large Scale Processes.
Ph.D. thesis, University of Heidelberg, 2001
`http://www.iwr.uni-heidelberg.de/~Moritz.Diehl/-DISSERTATION/diehl_diss.ps.gz`

[DNR 87] Duff, I.S.; Nocedal, J.; Reid, J.K.:
The Use of Linear Programming for the Solution of Sparse Sets of Nonlinear Equations.
SIAM Journal on Scientific and Statistical Computing **8**, 2 (1987), pp. 99-108

[Dubb 81] Dubbel:
Taschenbuch für den Maschinenbau.
Beitz, W.; Küttner, K.-H. (editors),
Springer, Berlin 1981, 14^{th} edition

[Duff 81] Duff, I.S.:
On Algorithms for Obtaining a Maximum Transversal.
ACM Transactions on Mathematical Software **7**, 3 (1981), pp. 315-330

[Duff 99] Duff, I.S.:
Matrix Methods.
Proceedings of the meeting "Supercomputing, Collision Processes and Applications", Queen's University of Belfast (September 14 – 16, 1998), Bell, K.L.; Berrington, K.A.; Crother, D.S.F.; Hibbert, A.; Taylor, K.T. (editors),
Kluwer Academic/Plenum Publishers, New York 1999
Rutherford Appleton Laboratory Technical Report **RAL-TR-1998-076**
`http://www.numerical.rl.ac.uk/reports/reports.html`

[DUFS 01] Diehl, M.; Uslu, I.; Findeisen, R.; Schwarzkopf, S.; Allgöwer, F.; Bock, H.G.; Bürner, T.; Gilles, E.D.; Kienle, A.; Schlöder, J.P.; Stein, E.:
Real-Time Optimization for Large Scale Processes: Nonlinear Model Predictive Control of a High Purity Distillation Column.

in: *Online Optimization of Large Scale Systems*, Grötschel, M.; Krumke, S.O. (editors),
Springer, Heidelberg 2001, pp. 363-383
`http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/-Preprint-01-16.html`

[DuGe 86] Duff, I.S.; Gear, C.W.:
Computing the Structural Index.
SIAM Journal on Algebraic and Discrete Methods **7**, 4 (1986), pp. 594-603

[Dunk 84] Dunker, A.M.:
The Decoupled Direct Method for Calculating Sensitivity Coefficients in Chemical Kinetics.
Journal of Chemical Physics **81**, 5 (1984), pp. 2385-2393

[DuRe 96] Duff, I.S.; Reid, J.K.:
The Design of MA48*: A Code for the Direct Solution of Sparse Unsymmetric Linear Systems of Equations.*
ACM Transactions on Mathematical Software **22**, 2 (1996), pp. 187-226

[EaRa 92] Eaton, J.W.; Rawlings, J.B.:
Model-Predictive Control of Chemical Processes.
Chemical Engineering Science **47**, 4 (1992), pp. 705-720

[ECO 93] Elmqvist, H.; Cellier, F.E.; Otter, M.:
Object-Oriented Modeling of Hybrid Systems.
Proceedings of ESS'93, European Simulation Symposium, Delft, The Netherlands (October 25 – 28, 1993), pp. xxxi-xli
`http://www-er.df.op.dlr.de/publications/1993/-elmqvist_ess.ps.gz`

[Eich 92] Eich, E.:
Projizierende Mehrschrittverfahren zur numersichen Lösung von Bewegungsgleichungen technischer Mehrkörpersysteme mit Zwangsbedingungen und Unstetigkeiten.
Fortschritt-Berichte VDI, Reihe 18, Nr. **109** (1992)

[EKK 97] Engell, S.; Kowalewski, S.; Krogh, B.H.:
Discrete Events and Hybrid Systems in Process Control.
Proceedings of the Fifth International Conference of Chemical Process Control, Tahoe City, California (January 7 – 12, 1996), Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **316** (1997), pp. 165-176

[EKKS 99] Engl, G.; Kröner, A.; Kronseder, T.; von Stryk, O.:
Numerical Simulation and Optimal Control of Air Separation Plants.
in: *High Performance Scientific and Engineering Computing, Lecture Notes in Computational Science and Engineering Vol.* ***8***, Bungartz, H.-J.; Durst, F.; Zenger, Chr. (editors),
Proceedings of International FORTWIHR Conference on HPSEC, Munich (March 16 – 18, 1999),
Springer, New York 1999, pp. 221-231

[ELBK 97] Eich-Söllner, E.; Lory, P.; Burr, P.; Kröner, A.:
Stationary and Dynamic Flowsheeting in the Chemical Engineering Industry.
Surveys on Mathematics for Industry **7** (1997), pp. 1-28

[EMP 86] Economou, C.G.; Morari, M.; Palsson, B.O.:
Internal Model Control: 5. Extension to Nonlinear Systems.
Industrial & Engineering Chemistry Process Design and Development **25** (1986), pp. 403-411

[EnSc 97] Engl, G.; Schmidt, H.:
The optimization of natural gas liquefaction processes.
in: *Progress in Industrial Mathematics at ECMI 96*, Brøns, M.; Bensøe, M.P.; Sørensen, M.P. (editors),
B.G. Teubner, Stuttgart 1997, pp. 356-363

[ErAr 98] Ertel, S.; Arnold, M.:
Die Berechnung von Sensitivitätsmatrizen unter Berücksichtigung unstetiger Zustandsänderungen.
Manuskript, DLR, Oberpfaffenhofen (March 1998)

[ESBZ 97] Engell, S.; von Stryk, O.; Breitner, M.H.; Zimmermann, U.:
Definitionsversuch Echtzeitoptimierung.
working paper, April 1997

[EsLa 99] Estévez Schwarz, D.; Lamour, R.:
The Computation of Consistent Initial Values for Nonlinear Index-2 Differential-Algebraic Equations.
Institut für Mathematik, Humboldt Universität zu Berlin, Preprint **99-13**, Berlin (1999)
`http://taylor.mathematik.hu-berlin.de/publ/pre/1999/-p-list-99.html`

[Este 00] Estévez Schwarz, D.:
Consistent Initialization for Index-2 Differential-Algebraic Equations and its Application to Circuit Simulation.
Ph.D. thesis, Humboldt-Universität zu Berlin, Germany, May 30,

2000
`http://dochost.rz.hu-berlin.de/abstract.php3/-dissertationen/estevez-schwarz-diana-2000-07-13`

[Este 99a] Estévez Schwarz, D.:
Topological Analysis for Consistent Initialization in Circuit Simulation.
Institut für Mathematik, Humboldt Universität zu Berlin, Preprint **99-3**, Berlin (1999)
`http://taylor.mathematik.hu-berlin.de/publ/pre/1999/-p-list-99.html`

[Este 99b] Estévez Schwarz, D.:
Consistent Initial Values for DAE Systems in Circuit Simulation.
Institut für Mathematik, Humboldt Universität zu Berlin, Preprint **99-5**, Berlin (1999)
`http://taylor.mathematik.hu-berlin.de/publ/pre/1999/-p-list-99.html`

[FBB 95] Feehery, W.F.; Banga, J.R.; Barton, P.I.:
A Novel Approach to Dynamic Optimization of ODE and DAE Systems as High-Index Problems.
Proceedings of AICHE Annual Meeting, Miami (1995)

[FeBa 96] Feehery, W.F.; Barton, P.I.:
A Differentiation-Based Approach to Dynamic Simulation and Optimization with High-Index Differential-Algebraic Equations.
in: *Computational Differentiation – Techniques, Applications, and Tools*, Berz, M.; Bischof, C.; Corliss, G.; Griewank, A. (editors), SIAM, Philadelphia 1996, pp. 239-253
`http://yoric.mit.edu/cgi-bin/bartonpub`

[FeBa 98] Feehery, W.F.; Barton, P.I.:
Dynamic Optimization with State Variable Constraints.
Computers and Chemical Engineering **22**, 9 (1998), pp. 1241-1256

[Feeh 98] Feehery, W.F.:
Dynamic Optimization with Path Constraints.
Ph.D. thesis, Massachusetts Institute of Technology, March 5, 1998
`http://yoric.mit.edu/PIB/barton-rep.html`

[Fiac 83] Fiacco, A.V.:
Introduction to Sensitivity and Stability Analysis in Nonlinear Programming.
Mathematics in Science and Engineering, Vol. **165**, Academic Press, New York 1983

[FiAl 99] Findeisen, R.; Allgöwer, F.:
Nonlinear Model Predictive Control for Index-One DAE Systems.
IFA Technical Report **AUT-99-10**
`http://www.aut.ee.ethz.ch/research/publications/-publications.msql`

[Fish 91] Fisher, D.G.:
Process Control: An Overview and Personal Perspective.
Canadian Journal of Chemical Engineering **69** (1991), pp. 5-26

[Flet 87] Fletcher, R.:
Practical Methods of Optimization.
John Wiley & Sons, Chichester 1987, 2nd edition

[FTB 97] Feehery, W.F.; Tolsma, J.E.; Barton, P.I.:
Efficient Sensitivity Analysis of Large-Scale Differential-Algebraic Systems.
Applied Numerical Mathematics **25**, 1 (1997), pp. 41-54

[Führ 88] Führer, C.:
Differential-algebraische-Gleichungssysteme in mechanischen Mehrkörpersystemen: Theorie, numerische Ansätze und Anwendungen.
Ph.D. thesis, Technische Universität München, 1988

[FüLe 91] Führer, C.; Leimkuhler, B.J.:
Numerical Solution of Differential-Algebraic Equations for Constrained Mechanical Motion.
Numerische Mathematik **59** (1991), pp. 55-69

[GaBa 98] Galán, S.; Barton, P.I.:
Dynamic Optimization of Hybrid Systems.
Computers and Chemical Engineering **22S** (1998), pp. S183-S190

[GaCa 92] Gani, R.; Cameron, I.T.:
Modelling for Dynamic Simulation of Chemical Processes: the Index Problem.
Chemical Engineering Science **47**, 5 (1992), pp. 1311-1315

[GaMo 82] Garcia, C.E.; Morari, M.:
Internal Model Control: 1. A Unifying Review and Some New Results.
Industrial & Engineering Chemistry Process Design and Development **21** (1982), pp. 308-323

[Gear 71] Gear, C.W.:
Simultaneous Numerical Solution of Differential-Algebraic Equations.
IEEE Transactions on Circuit Theory **18**, 1 (1971), pp. 89-95

[Gear 88] Gear, C.W.:
Differential-Algebraic Equation Index Transformations.
SIAM Journal on Scientific and Statistical Computing **9**, 1 (1988), pp. 39-47

[Gear 90] Gear, C.W.:
Differential Algebraic Equations, Indices, and Integral Algebraic Equations.
SIAM Journal on Numerical Analysis **27**, 6 (1990), pp. 1527-1534

[GeBü 01] Gerdts, M.; Büskens, Ch.:
Computation of Consistent Initial Values for Optimal Control Problems with DAE Systems of Higher Index.
Zeitschrift für Angewandte Mathematik und Mechanik **81**, S2 (2001), pp. 249-250

[GeOs 84] Gear, C.W.; Østerby, O.:
Solving Ordinary Differential Equations with Discontinuities.
ACM Transactions on Mathematical Software **10**, 1 (1984), pp. 23-44

[GePe 84] Gear, C.W.; Petzold, L.R.:
ODE Methods for the Solution of Differential/Algebraic Systems.
SIAM Journal on Numerical Analysis **21**, 4 (1984), pp. 716-728

[Gerd 01] Gerdts, M.:
Numerische Methoden optimaler Steuerprozesse mit differential-algebraischen Gleichungssystemen höheren Indexes und ihre Anwendungen in der Kraftfahrzeugsimulation und Mechanik.
Bayreuther Mathematische Schriften Nr. **61**, Bayreuth, **2001**

[GFB 99] Galán, S.; Feehery, W.F.; Barton, P.I.:
Parametric Sensitivity Functions for Hybrid Discrete/Continuous Systems.
Applied Numerical Mathematics **31**, 1 (1999), pp. 17-48

[GHM 92] Griepentrog, E.; Hanke, M.; März, R.:
Toward a Better Understanding of Differential Algebraic Equations.
in: *Seminarbericht **92-1***, Griepentrog, E.; Hanke, M.; März, R. (editors),
Proceedings of the Berliner Seminar on Differential-Algebraic Equations, Berlin (January 1992), pp. 1-12
`http://www.mathematik.hu-berlin.de/publ/SB-92-1/intro.ps`

[GHMSW 86] Gill, P.E.; Hammarling, S.J.; Murray, W.; Saunders, M.A.; Wright, M.H.:
User's Guide for LSSOL (Version 1.0): A FORTRAN *Package for*

Constrained Least-Squares and Convex Quadratic Programming.
Stanford University, Department of Operations Research, Report **SOL 86-1**, Stanford (1986)
`http://www.tomlab.biz/docs/lssol1-0.pdf`

[GiMu 78] Gill, P.E.; Murray, W.:
Algorithms for the Solution of the Nonlinear Least-Squares Problem.
SIAM Journal on Numerical Analysis **15**, 5 (1978), pp. 977-992

[GJU 96] Griewank, A.; Juedes, D.; Utke, J:
ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++.
ACM Transactions on Mathematical Software **22**, 2 (1996), pp. 131-167

[GLG 85] Gear, C.W.; Leimkuhler, B.; Gupta, G.K.:
Automatic Integration of Euler-Lagrange Equations with Constraints.
Journal of Computational and Applied Mathematics **12 & 13** (1985), pp. 77-90

[GMS 97a] Gill, P.E.; Murray, W.; Saunders, M.A.:
SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization.
University of California, Department of Mathematics, Numerical Analysis Report **97-2**, San Diego (USA) (1997)
`http://www.scicomp.ucsd.edu/~peg/`

[GMS 97b] Gill, P.E.; Murray, W.; Saunders, M.A.:
User's Guide for SNOPT 5.3: A FORTRAN *Package for Large-Scale Nonlinear Programming.*
University of California, San Diego, Department of Mathematics, Report **NA 97-5**, California (USA) (1997)
`http://www.scicomp.ucsd.edu/~peg/`

[GMSW 83] Gill, P.E.; Murray, W.; Saunders, M.A.; Wright, M.H.:
Computing Forward-Difference Intervals for Numerical Optimization.
SIAM Journal on Scientific and Statistical Computing **4**, 2 (1983), pp. 310-321

[GMSW 98] Gill, P.E.; Murray, W.; Saunders, M.A.; Wright, M.H.:
User's Guide for NPSOL (Version 5.0): A FORTRAN *Package for Nonlinear Programming.*
University of California, Department of Mathematics, Numerical Analysis Report **98-2**, San Diego (USA) (1998)

[GMW 95] Gill, P.E.; Murray, W.; Wright, H.M.:
Practical Optimization.
Academic Press, London 1995, 10th printing

[GoBi 99] Gopal, V.; Biegler, L.T.:
A Successive Linear Programming Approach for Consistent Initialization of Differential Algebraic Equations and Reinitialization after Discontinuities.
SIAM Journal on Scientific Computing **20**, 2 (1999), pp. 447-467
`http://siam.org/journals/sisc/20-2/30772.html`

[Gold 63] Goldstein, H.:
Klassische Mechanik.
Akademische Verlagsgesellschaft, Frankfurt a. Main 1963

[GoTo 00] Gould, N.I.M; Toint, P.L.:
SQP Methods for Large-Scale Nonlinear Programming.
Proceedings of the 19th IFIP TC7 Conference on System Modelling and Optimization, Cambridge, England (July 12 – 16, 1999), Powell, M.J.D.; Scholtes, S. (editors),
Kluwer Academic Publishers, Boston 2000
Rutherford Appleton Laboratory Technical Report **RAL-TR-1999-055**
`http://www.numerical.rl.ac.uk/reports/reports.html`

[GPM 89] Garcia, C.E.; Prett, D.M.; Morari, M.:
Model Predictive Control: Theory and Practice — a Survey.
Automatica **25**, 3 (1989), pp. 335-348

[GPS 95] Gritsis, D.M.; Pantelides, C.C.; Sargent, R.W.H.:
Optimal Control of Systems Described by Index Two Differential-Algebraic Equations.
SIAM Journal on Scientific Computing **16**, 6 (1995), pp. 1349-1366

[Grit 90] Gritsis, D.:
The Dynamic Simulation and Optimal Control of Systems Described by Index-Two Differential-Algebraic Equations.
Ph.D. thesis, Imperial College of Science, Technology and Medicine, London, 1990

[GRP 91] Garcia, C.E.; Ramaker, B.L.; Pollard, J.F.:
Total Process Control – Beyond the Design of Model Predictive Controllers.
in: *Chemical Process Control CPC-IV*, Arkun, Y.; Ray, W.H. (editors),
Proceedings of the Fourth International Conference on Chemical

Process Control, Padre Island, Texas (February 17 – 22, 1991), pp. 335-360

[GrSh 96] Griewank, A.; Shiriaev, D.:
ADOL-F – Automatic Differentiation of Fortran Codes.
in: *Computational Differentiation – Techniques, Applications, and Tools*, Berz, M.; Bischof, C.; Corliss, G.; Griewank, A. (editors), SIAM, Philadelphia 1996, pp. 375-384

[Grup 96] Grupp, F.:
Parameteridentifizierung nichtlinearer mechanischer Deskriptorsysteme mit Anwendungen in der Rad-Schiene-Dynamik.
Fortschritt-Berichte VDI, Reihe 8, Nr. **550** (1996)

[GSB 87] Gladwell, I.; Shampine, L.F.; Brankin, R.W.:
Automatic Selection of the Initial Step Size for an ODE Solver.
Journal of Computational and Applied Mathematics **18** (1987), pp. 175-192

[GüFe 99a] Günther, M.; Feldmann, U.:
CAD-Based Electric-Circuit Modeling in Industry I: Mathematical Structure and Index of Network Equations.
Surveys on Mathematics for Industry **8**, 2 (1999), pp. 97-129

[GüFe 99b] Günther, M.; Feldmann, U.:
CAD-Based Electric-Circuit Modeling in Industry II: Impact of Circuit Configurations and Parameters.
Surveys on Mathematics for Industry **8**, 2 (1999), pp. 131-157

[Hage 84] Hager, W.W.:
Condition Estimates.
SIAM Journal on Scientific and Statistical Computing **5**, 2 (1984), pp. 311-316

[HaGr 80] Hay, J.L.; Griffin, A.W.J:
Simulation of Discontinuous Dynamical Systems.
in: *Simulation of Systems '79*, Dekker, L.; Sevastano, G.; Van Steenkiste, G.C. (editors),
Proceedings of the 9^{th} IMACS Congress, Sorrento, Italy (September 24 – 28, 1979),
North-Holland, Amsterdam 1980, pp. 79-87

[HaLa 01] Hanke, M.; Lamour, R.:
Consistent Initialization for Nonlinear Index-2 Differential-Algebraic Equation: Large Sparse Systems in MATLAB.
PSCI Report **2001:05**, Royal Institute of Technology, Stockholm,

2001
`http://www.nada.kth.se/~hanke/ps/consinin.1.ps`

[HAM 00] Helbig, A.; Abel, O.; Marquardt, W.:
Structural Concepts for Optimization Based Control of Transient Processes.
in: *Progress in Systems and Control Theory, Vol.* ***26***, Allgöwer, F.; Zheng, A. (editors),
Proceedings of the International Symposium on Nonlinear Model Predictive Control – Assessment and Future Directions, Ascona, Switzerland (June 3-5, 1998),
Birkhäuser, Basel 2000, pp. 295-311
`http://www.lfpt.rwth-aachen.de/Publication/Techreport/-1998/LPT-1998-20.html`

[Han 77] Han, S.P.:
A Globally Convergent Method for Nonlinear Programming.
Journal of Optimization Theory and Applications **22**, 3 (1977), pp. 297-309

[Hank 95] Hanke, M.:
Regularizations of Differential-Algebraic Equations Revisited.
Mathematische Nachrichten **174** (1995), pp. 159-183
`http://www.nada.kth.se/~hanke/ps/pitt.ps`

[Hans 92] Hansen, B.:
Computing Consistent Initial Values for Nonlinear Index-2 Differential-Algebraic Equations.
in: *Seminarbericht* ***92-1***, Griepentrog, E.; Hanke, M.; März, R. (editors),
Proceedings of the Berliner Seminar on Differential-Algebraic Equations, Berlin (January 1992), pp. 142-157
`http://www.mathematik.hu-berlin.de/publ/SB-92-1/-hansen1.ps`

[Heim 92] Heim, A.:
Parameteridentifizierung in differential-algebraischen Gleichungssystemen.
Diploma thesis, Mathematisches Institut der Technischen Universität München, 23.11.1992

[Hens 98] Henson, M.A.:
Nonlinear Model Predictive Control: Current Status and Future Directions.
Computers and Chemical Engineering **23** (1998), pp. 187-202

[HeSt 96] Heim, A.; von Stryk, O.:
Documentation of PAREST — A Multiple Shooting Code for Optimization Problems in Differential-Algebraic Equations.
Technische Universität München, Fakultät für Mathematik, Report **TUM M9616** (1996)

[High 88] Higham, N.J.:
FORTRAN *Codes for Estimating the One-Norm of a Real or Complex Matrix, with Applications to Condition Estimation.*
ACM Transactions on Mathematical Software **14**, 4 (1988), pp. 381-396

[High 92] Higham, N.J.:
Estimating the Matrix p-norm.
Numerische Mathematik **62** (1992), pp. 539-555

[HiHe 86] Hillestad, M.; Hertzberg, T.:
Dynamic Simulation of Chemical Engineering Systems by the Sequential Modular Approach.
Computers and Chemical Engineering **10**, 4 (1986), pp. 377-388

[Hins 97] Hinsberger, H.:
Ein direktes Mehrzielverfahren zur Lösung von Optimalsteuerungsproblemen mit großen, differential-algebraischen Gleichungssystemen und Anwendungen aus der Verfahrenstechnik.
Ph.D. thesis, Mathematisch-Naturwissenschaftliche Fakultät, Technische Universität Clausthal, 1997

[HiRa 71] Hicks, G.A.; Ray, W.H.:
Approximation Methods for Optimal Control Synthesis.
Canadian Journal of Chemical Engineering **49** (1971), pp. 522-528

[HiTi 00] Higham, N.J.; Tisseur, F.:
A Block Algorithm for Matrix 1-Norm Estimation, with an Application to 1-Norm Pseudospectra.
SIAM Journal on Matrix Analysis and Applications **4**, 21 (2000), pp. 1185-1201

[HMG 88] Holl, P.; Marquardt, W.; Gilles, E.D.:
DIVA – A Powerful Tool for Dynamic Process Simulation.
Computers and Chemical Engineering **12**, 5 (1988), pp. 421-426

[HNW 87] Hairer, E.; Nørsett, S.P.; Wanner, G.:
Solving Ordinary Differential Equations I: Nonstiff Problems.
Springer, Berlin 1987

[HoLi 83] Holland, C.D.; Liapis, A.I.:
Computer Methods for Solving Dynamic Separation Problems.
McGraw-Hill, New York 1983

[HRW 99] Hoschek, M.; Rentrop, P.; Wagner, Y.:
Network Approach and Differential-Algebraic Systems in Technical Applications.
Surveys on Mathematics for Industry **9** (1999), pp. 49-76

[HW 91] Hairer, E.; Wanner, G.:
Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.
Springer, Berlin 1991

[ISM 98] Iavernaro, F.; La Scala, M.; Mazzia, F.:
Boundary Values Methods for Time-Domain Simulation of Power System Dynamic Behavoir.
IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications **45**, 1 (1998), pp. 50-63

[Jonc 88] Jonckheere, E.:
Variational Calculus for Descriptor Problems.
IEEE Transactions on Automatic Control **AC-33**, 5 (1988), pp. 491-495

[Kels 95] Kelsey, T.W.:
The Formulation of LP and MILP Problems Which Model an Industrial Chemical Operation.
Master's thesis, University of Dundee, Department of Mathematics and Computer Science, October 1995
`http://www.dcs.st-and.ac.uk/rsch/publications/-Kel95.shtml`

[KFGE 97] Kreul, L.U.; Fernholz, G.; Górak, A.; Engell, S.:
Erfahrungen mit den dynamischen Simulatoren DIVA, gPROMS und ABACUSS.
Chemie - Ingenieur - Technik **65**, 5 (1997), pp. 650-653

[KHMG 90] Kröner, A.; Holl, P.; Marquardt, W.; Gilles, E.D.:
DIVA – An Open Architecture for Dynamic Simulation.
Computers and Chemical Engineering **14**, 11 (1990), pp. 1289-1295

[Kieh 99] Kiehl, M.:
Sensitivity Analysis of ODEs and DAEs — Theory and Implementation Guide.
Optimization Methods and Software **10**, 6 (1999), pp. 803-821

`http://www-lit.ma.tum.de/veroeff/html/980.65004.html`
(preprint)

[KKES 01] Kröner, A.; Kronseder, T.; Engl, G.; von Stryk, O.:
Dynamic Optimization for Air Separation Plants.
in: *Computer-Aided Chemical Engineering 9*, Gani, R.; Jørgensen, S.B. (editors),
Proceedings of the European Symposium on Computer Aided Process Engineering (ESCAPE-11), Kolding, Denmark (May 27 – 30, 2001),
Elsevier Science, Amsterdam 2001, pp. 433-438

[KMG 92] Kröner, A.; Marquardt, W.; Gilles, E.D.:
Computing Consistent Initial Conditions for Differential-Algebraic Equations.
Computers and Chemical Engineering **16S** (1992), pp. S131-S138

[KMG 97] Kröner, A.; Marquardt, W.; Gilles, E.D.:
Getting Around Consistent Initialization of DAE Systems?
Computers and Chemical Engineering **21**, 2 (1997), pp. 145-158

[Kraf 85] Kraft, D.:
On Converting Optimal Control Problems into Nonlinear Programming Problems.
in: *Computational Mathematical Programming, NATO ASI Series Vol. **F15***, Schittkowski, K. (editor),
Springer, Berlin 1985, pp. 261-280

[Kräm 85] Krämer-Eis, P.:
Ein Mehrzielverfahren zur numerischen Berechnung optimaler Feedback-Steuerungen bei beschränkten nichtlinearen Steuerungsproblemen.
Brieskorn, E.; Frehse, J.; Hildebrandt, St.; Hirzebruch, F.; Harder, G.; Klingenberg, W.; Leis, R.; Lieb, I.; Peschl, E.; Unger, H.; Vogel, W.; Werner, H. (editors),
Bonner Mathematische Schriften Nr. **166**, Bonn, 1985

[Krön 02] Kröner, A.:
Algorithmen zur Analyse und Lösung kontinuierlich-diskreter Differential-Algebra-Geichungen in der Simulationsumgebung DIVA.
Ph.D. thesis, Universität Stuttgart, 2002

[Kron 98] Kronseder, T.:
Optimal Control of Chemical Engineering Processes.
Diploma thesis, Mathematisches Institut der Technischen Universität München, 15.08.1998

[KSBK 01] Kronseder, T.; von Stryk, O.; Bulirsch, R.; Kröner, A.:
Towards Nonlinear Model-Based Predictive Optimal Control of Large-Scale Process Models with Application to Air Separation Plants.
in: *Online Optimization of Large Scale Systems*, Grötschel, M.; Krumke, S.O. (editors),
Springer, Heidelberg 2001, pp. 385-410

[KuDa 98] Kumar, A.; Daoutidis, P.:
Control of Nonlinear Differential Algebraic Equation Systems: An Overview.
in: *Nonlinear Model Based Process Control, NATO ASI Series Vol.* ***E353***, Berber, R.; Kravaris, C. (editors),
Kluwer Academic Publishers, Dordrecht 1998, pp. 311-344

[KuDa 99] Kumar, A.; Daoutidis, P.:
Control of Nonlinear Differential Algebraic Equations with Applications to Chemical Processes.
Chapman & Hall CRC, Boca Raton 1999

[KuMe 94] Kunkel, P.; Mehrmann, V.:
Canonical Forms for Linear Differential-Algebraic Equations with Variable Coefficients.
Journal of Computational and Applied Mathematics **56** (1994), pp. 225-251

[KuPe 90a] Kugelmann, B.; Pesch, H.J.:
New General Guidance Method in Constrained Optimal Control, Part 1: Numerical Method.
Journal of Optimization Theory and Applications **67**, 3 (1990), pp. 421-435

[KuPe 90b] Kugelmann, B.; Pesch, H.J.:
New General Guidance Method in Constrained Optimal Control, Part 2: Application to Space Shuttle Guidance.
Journal of Optimization Theory and Applications **67**, 3 (1990), pp. 437-446

[KuWe 99] Kugelmann, B.; Weber, W.:
A Parallel Method for Optimal Guidance in Air Traffic.
Schwerpunktprogramm der Deutschen Forschungsgemeinschaft "Echtzeitoptimierung Großer Systeme", Preprint **99-6**
`http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/-Preprint-99-6.html`

[Lamo 97] Lamour, R.:
A Shooting Method for Fully Implicit Index-2 Differential-Algebraic Equations.
SIAM Journal on Scientific Computing **18**, 1 (1997), pp. 94-114

[LBEM 90] Li, W.C.; Biegler, L.T.; Economou, C.G.; Morari, M.:
A Constrained Pseudo-Newton Control Strategy for Nonlinear Systems.
Computers and Chemical Engineering **14**, 4/5 (1990), pp. 451-486

[LBS 97] Leineweber, D.B.; Bock, H.G.; Schlöder, J.P.:
Fast Direct Methods for Real-Time Optimization of Chemical Processes.
Schwerpunktprogramm der Deutschen Forschungsgemeinschaft "Echtzeitoptimierung Großer Systeme", Preprint **97-3**
`http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/-Preprint-97-3.html`

[LeBr 89] Lee, A.Y.; Bryson, A.E.:
Neighbouring Extremals of Dynamic Optimization Problems with Parameter Variations.
Optimal Control, Applications & Methods **10** (1989), pp. 39-52

[LeCo 97] Lee, J.H.; Cooley, B.:
Recent Advances in Model Predictive Control and Other Related Areas.
Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **316**, 93 (1997), pp. 201-216

[Lee 00] Lee, J.H.:
Modeling and Identification for Nonlinear Model Predictive Control: Requirements, Current Status and Future Research Needs.
in: *Progress in Systems and Control Theory, Vol. **26***, Allgöwer, F.; Zheng, A. (editors),
Proceedings of the International Symposium on Nonlinear Model Predictive Control – Assessment and Future Directions, Ascona, Switzerland (June 3-5, 1998),
Birkhäuser, Basel 2000, pp. 269-293

[Leim 88] Leimkuhler, B.J.:
Approximation Methods for the Consistent Initialization of Differential-Algebraic Equations.
Ph.D. thesis, University of Illinois at Urbana-Champaign, Dept. of Computer Science, Report No. **UIUCDCS-R-88-1450**, August 1988

[LeKr 85] Leis, J.R.; Kramer, M.A.:
Sensitivity Analysis of Systems of Differential and Algebraic Equations.
Computers and Chemical Engineering **9**, 1 (1985), pp. 93-96

[LEL 92] Liebman, M.J.; Edgar, T.F.; Lasdon, L.S.:
Efficient Data Reconciliation and Estimation For Dynamic Processes Using Nonlinear Programming Techniques.
Computers and Chemical Engineering **16**, 10/11 (1992), pp. 963-986

[LeSr 93a] Lefkopoulos, A.; Stadtherr, M.A.:
Index Analysis of Unsteady-State Chemical Process Systems I: An Algorithm for Problem Formulation.
Computers and Chemical Engineering **17**, 4 (1993), pp. 399-413

[LeSr 93b] Lefkopoulos, A.; Stadtherr, M.A.:
Index Analysis of Unsteady-State Chemical Process Systems II: Strategies for Determining the Overall Flowsheet Index.
Computers and Chemical Engineering **17**, 4 (1993), pp. 415-430

[LiBi 88] Li, W.C.; Biegler, L.T.:
Process Control Strategies for Constrained Nonlinear Systems.
Industrial & Engineering Chemistry Research **27**, 8 (1988), pp. 1421-1433

[LiBi 89] Li, W.C.; Biegler, L.T.:
Multistep, Newton-type Control Strategies for Constrained, Nonlinear Processes.
Chemical Engineering Research and Design **67** (1989), pp. 562-577

[LiBi 90] Li, W.C.; Biegler, L.T.:
Newton-Type Controllers for Constrained Nonlinear Processes with Uncertainty.
Industrial & Engineering Chemistry Research **29**, 8 (1990), pp. 1647-1657

[Lind 98] Linde AG, Process Engineering and Contracting Division:
Linde builds world's biggest air separators in Mexico.
Linde Mail, Linde AG Wiesbaden, Nr. **1/1998**

[LiPe 99] Li, S.; Petzold, L.R.:
Design of New DASPK for Sensitivity Analysis.
Computer Science Technical Report **TRCS99-28**, Department of Computer Science, University of California, Santa Barbara, 1999
`http://www.cs.ucsb.edu/TRs/Html/TRCS99-28.html`

[LiYa 88] Lin, J.-Y.; Yang, Z.-H.:
Optimal Control Problems for Singular Systems.
International Journal of Control **47**, 6 (1988), pp. 1915-1924

[LöPe 86] Lötstedt, P.; Petzold, L.:
Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints I: Convergence Results for Backward Differentiation Formulas.
Mathematics of Computation **46**, 174 (1986), pp. 491-516

[LoPe 99a] Loeblein, C.; Perkins, J.D.:
Structural Design for On-Line Process Optimization: I. Dynamic Economics of MPC.
AIChE Journal **45**, 5 (1999), pp. 1018-1029

[LoPe 99b] Loeblein, C.; Perkins, J.D.:
Structural Design for On-Line Prorcess Optimization: II. Application to a Simulated FCC.
AIChE Journal **45**, 5 (1999), pp. 1030-1040

[LPG 91] Leimkuhler, B.; Petzold, L.R.; Gear, C.W.:
Approximation Methods for the Consistent Initialization of Differential-Algebraic Equations.
SIAM Journal on Numerical Analysis **28**, 1 (1991), pp. 205-226

[Luyb 01] Luyben, W.L.:
Comparison of Commercial Dynamic Simulators.
presented at AIChE Annual Meeting, Reno (Nevada), November 4-9, 2001
`http://www.aiche.org/conferences/techprogram/-paperdetail.asp?PaperID=267&DSN=annual01`

[MaAu 01] Maurer, H.; Augustin, D.:
Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Boundary Value Methods.
in: *Online Optimization of Large Scale Systems*, Grötschel, M.; Krumke, S.O. (editors),
Springer, Heidelberg 2001, pp. 17-55

[MaMi 90] Mayne, D.Q.; Michalska, H.:
Receding Horizon Control of Nonlinear Systems.
IEEE Transactions on Automatic Control **AC-35**, 7 (1990), pp. 814-824

[MaPe 96] Maly, T.; Petzold, L.R.:
Numerical Methods and Software for Sensitivity Analysis of

Differential-Algebraic Systems.
Applied Numerical Mechanics **20** (1996), pp. 57-79
`http://www.engineering.ucsb.edu/~cse/publica.html`

[MaPes 91] Maurer, H.; Pesch, H.J.:
Solution Differentiability for Nonlinear Parametric Control Problems.
Schwerpunktprogramm der Deutschen Forschungsgemeinschaft "Anwendungsbezogene Optimierung und Steuerung", Report No. **316** (1991)

[MaPes 93] Maurer, H.; Pesch, H.J.:
Solution Differentiability for Parametric Nonlinear Control Problems with Control-State Constraints.
Schwerpunktprogramm der Deutschen Forschungsgemeinschaft "Anwendungsbezogene Optimierung und Steuerung", Report No. **474** (1993)

[MaPes 94] Maurer, H.; Pesch, H.J.:
Solution Differentiability for Nonlinear Parametric Control Problems.
SIAM Journal on Control and Optimization **32**, 6 (1994), pp. 1542-1554

[MaPes 95] Maurer, H.; Pesch, H.J.:
Solution Differentiability for Parametric Nonlinear Control Problems with Control-State Constraints.
Journal of Optimization Theory and Applications **86**, 2 (1995), pp. 285-309

[MaPi 95] Maurer, H.; Pickenhain, S.:
Second-Order Sufficient Conditions for Control Problems with Mixed Control-State Constraints.
Journal of Optimization Theory and Applications **86**, 3 (1995), pp. 649-667

[Marq 91] Marquardt, W.:
Dynamic Process Simulation – Recent Progress and Future Challenges.
in: *Chemical Process Control CPC-IV*, Arkun, Y.; Ray, W.H. (editors),
Proceedings of the Fourth International Conference on Chemical Process Control, Padre Island, Texas (February 17 – 22, 1991), pp. 131-180

[Marq 99a] Marquardt, W.:
Von der Prozesssimulation zur Lebenszyklusmodellierung.
Chemie - Ingenieur - Technik **71** (1999), pp. 1119-1137
`http://www.lfpt.rwth-aachen.de/Publication/Techreport/-1999/LPT-1999-16.html`

[März 89] März, R.:
Index-2 Differential-Algebraic Equations.
Results in Mathematics **15** (1989), pp. 149-171

[März 90] März, R.:
Analysis and Numerical Treatment of Differential-Algebraic Equations.
in: *Mathematical Modelling and Simulation of Electrical Circuits and Semiconductor Devices*, Bank, R.E.; Bulirsch, R.; Merten, K. (editors),
Proceedings of a conference held at the Math. Forschungsinst., Oberwolfach (October 30 – November 5, 1988),
International Series of Numerical Mathematics **93** (1990), pp. 27-43

[März 92] März, R.:
Numerical Methods for Differential Algebraic Equations.
Acta Numerica (1992), pp. 141-198

[März 97] März, R.:
EXTRA-Ordinary Differential Equations. Attempts to an Analysis of Differential-Algebraic Systems.
Humboldt Universität zu Berlin, Naturwissenschaftliche Fakultät II, Institut für Mathematik, preprint **97-8**, 1997
`http://www.mathematik.hu-berlin.de/publ/pre/1997/-P-97-8.ps`

[MaSö 92] Mattsson, S.E.; Söderlind, G.:
A New Technique for Solving High-Index Differential-Algebraic Equations Using Dummy Dervatives.
Proceedings of the 1992 IEEE Symposium on Computer-Aided Control System Design, CADCS'92, Napa, California (March 17 – 19, 1992), pp. 218-224
`http://membres.tripod.fr/jldormoy/Simulation/DAEIndex/-cacsd92daeindex_ps.gz`

[MaSö 93] Mattsson, S.E.; Söderlind, G.:
Index Reduction in Differential-Algebraic Equations Using Dummy Dervatives.
SIAM Journal on Scientific Computing **14**, 3 (1993), pp. 677-692

[MäTi 94] März, R.; Tischendorf, C.:
Solving More General Index-2 Differential Algebraic Equations.
Computers & Mathematics with Applications **28**, 10-12 (1994), pp. 77-105

[MäTi 97] März, R.; Tischendorf, C.:
Recent Results in Solving Index-2 Differential-Algebraic Equations in Circuit Simulation.
SIAM Journal on Scientific Computing **18**, 1 (1997), pp. 139-159

[Matt 89] Mattsson, S.E.:
On Modelling and Differential/Algebraic Systems.
Simulation **52**, 1 (1989), pp. 24-32

[Mayn 00] Mayne, D.Q.:
Nonlinear Model Predictive Control: Challenges and Opportunities.
in: *Progress in Systems and Control Theory, Vol. **26***, Allgöwer, F.; Zheng, A. (editors),
Proceedings of the International Symposium on Nonlinear Model Predictive Control – Assessment and Future Directions, Ascona, Switzerland (June 3-5, 1998),
Birkhäuser, Basel 2000, pp. 23-44

[Mayn 97] Mayne, D.Q.:
Nonlinear Model Predictive Control: An Assessment.
Proceedings of the Fifth International Conference of Chemical Process Control, Tahoe City, California (January 7 – 12, 1996), Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **316**, 93 (1997), pp. 217-231

[MeRa 97] Meadows, E.S.; Rawlings, J.B.:
Model Predictive Control.
in: *Nonlinear Process Control*, Henson, M.A.; Seborg, D.E. (editors), Prentice-Hall, Upper Saddle River, New Jersey 1997, pp. 233-310

[Miha 99] Mihatsch, O:
Früherkennung kritischer Zustände in chemischen Reaktoren mit Hilfe neuronaler Netze.
Fortschritt-Berichte VDI, Reihe 3, Nr. **579** (1999)

[MiMa 93] Michalska, H.; Mayne, D.Q.:
Robust Receding Horizon Control of Constrained Nonlinear Systems.
IEEE Transactions on Automatic Control **AC-38**, 11 (1993), pp. 1623-1633

[MiMa 95] Michalska, H.; Mayne, D.Q.:
Moving Horizon Observers and Observer-Based Control.

IEEE Transactions on Automatic Control **AC-40**, 6 (1995), pp. 995-1006

[MMG 95] Majer, C.; Marquardt, W.; Gilles, E.D.:
Reinitialization of DAE's after Discontinuities.
Computers and Chemical Engineering **19S** (1995), pp. S507-S512

[Moe 95] Moe, H.I.:
Dynamic Process Simulation – Studies on Modeling and Index Reduction.
Ph.D. thesis, University of Trondheim, 1995

[MoLe 91] Morari, M.; Lee, J.H.:
Model Predictive Control: The Good, the Bad, and the Ugly.
in: *Chemical Process Control CPC-IV*, Arkun, Y.; Ray, W.H. (editors),
Proceedings of the Fourth International Conference on Chemical Process Control, Padre Island, Texas (February 17 – 22, 1991), pp. 419-444

[MoLe 99] Morari, M.; Lee, J.H.:
Model Predictive Control: Past, Present and Future.
Computers and Chemical Engineering **23**, 4/5 (1999), pp. 667-682

[MoSa 86] Morison, K.R.; Sargent, R.W.H.:
Optimization of Multistage Processes Described by Differential-Algebraic Equations.
in: *Lecture Notes in Mathematics **1230***, Dold, A.; Eckmann, B. (editors),
Springer, New York 1986, pp. 86-102

[MRRS 00] Mayne, D.Q.; Rawlings, J.B.; Rao, C.V.; Scokaert, P.O.M:
Constrained Model Predictive Control: Stability and Optimality.
Automatica **36**, 6 (2000), pp. 789-814

[MuEd 97] Muske, K.R.; Edgar, T.F.:
Nonlinear State Estimation.
in: *Nonlinear Process Control*, Henson, M.A.; Seborg, D.E. (editors),
Prentice-Hall, Upper Saddle River, New Jersey 1997, pp. 311-370

[MuRa 93] Muske, K.R.; Rawlings, J.B.:
Model Predictive Control with Linear Models.
AIChE Journal **39**, 2 (1993), pp. 262-287

[NAG 94a] NAG:
E04 - Minimizing or Maximizing a Function.

NAG Fortran Library Manual Mark **16**, Vol. **4**, Numerical Algorithms Group, Oxford 1994

[NAG 94b] NAG:
E04UCF - NAG Fortran Library Routine Document.
NAG Fortran Library Manual Mark **16**, Vol. **4**, Numerical Algorithms Group, Oxford 1994

[NAG 94c] NAG:
E04UPF - NAG Fortran Library Routine Document.
NAG Fortran Library Manual Mark **16**, Vol. **4**, Numerical Algorithms Group, Oxford 1994

[Nijs 96] Nijsing, J.A.R.:
Increasing the ASU Controllability by the Use of Dynamic Modeling & Simulation.
Proceedings of Munich Meeting of Air Separation Technology MUST'96, Linde AG, Process Engineering and Contracting Division, Höllriegelskreuth (1996), pp. 93-111

[NMS 00] De Nicolao, G.; Magni, L.; Scattolini, R.:
Stability and Robustness of Nonlinear Receding Horizon Control.
in: *Progress in Systems and Control Theory, Vol. **26***, Allgöwer, F.; Zheng, A. (editors),
Proceedings of the International Symposium on Nonlinear Model Predictive Control – Assessment and Future Directions, Ascona, Switzerland (June 3-5, 1998),
Birkhäuser, Basel 2000, pp. 3-22

[NoWe 91] Nowak, U.; Weimann, L.:
A Family of Newton Codes for Systems of Highly Nonlinear Equations.
Konrad-Zuse-Zentrum für Informationstechnik, Technical Report **TR-91-10**, Berlin (1991)
`ftp://ftp.zib.de/bibarch/TR-91-10.*`

[OgWr 97] Ogunnaike, B.; Wright, R.A.:
Industrial Applications of Nonlinear Control.
Proceedings of the Fifth International Conference of Chemical Process Control, Tahoe City, California (January 7 – 12, 1996), Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **316**, 93 (1997), pp. 46-59

[OhPa 96] Oh, M.; Pantelides, C.C.:
A Modelling and Simulation Language for Combined Lumped and

Distributed Parameter Systems.
Computers and Chemical Engineering **20**, 6/7 (1996), pp. 611-633

[OLea 80] O'Leary, D.P.:
Estimating Matrix Condition Numbers.
SIAM Journal on Scientific and Statistical Computing **1**, 2 (1980), pp. 205-209

[Oliv 96] De Oliveira, S.L.:
Model Predictive Control (MPC) for Constrained Nonlinear Systems.
Ph.D. thesis, California Institute of Technology, Pasadena, California, March 5, 1996,
IFA Technical Report **AUT-96-33**
`http://www.aut.ee.ethz.ch/research/publications/-publications.msql`

[OlMo 96a] De Oliveira, S.L.; Morari, M.:
Contractive Model Predictive Control with Local Linearization for Nonlinear Systems.
IFA Technical Report **AUT-96-14**
`http://www.aut.ee.ethz.ch/research/publications/-publications.msql`

[OlMo 96b] De Oliveira, S.L.; Morari, M.:
Contractive Model Predictive Control for Constrained Nonlinear Systems.
IFA Technical Report **AUT-96-13**
`http://www.aut.ee.ethz.ch/research/publications/-publications.msql`

[Otte 02] Otter, M.:
private communications, 2002

[Otte 95] Otter, M.:
Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter.
Fortschritt-Berichte VDI, Reihe 20, Nr. **147** (1995)
`http://www-er.df.op.dlr.de/publications/1995/-otter_diss.ps.gz`

[Otte 96] Otter, M.:
A Constructive Definition of the Perturbation Index.
presented at the COSY Workshop on Mathematical Modeling of Complex Systems, Lund, Sweden, September 5-7, 1996

`http://www-er.df.op.dlr.de/FF-DR-ER/publications/1996/-otter_cosy.html`

[PaBa 93] Pantelides, C.C; Barton, P.I.:
Equation-Oriented Dynamic Simulation: Current Status and Future Perspectives.
Computers and Chemical Engineering **17S** (1993), pp. S263-S285

[Pant 88a] Pantelides, C.C.:
The Consistent Initialization of Differential-Algebraic Systems.
SIAM Journal on Scientific and Statistical Computing **9**, 2 (1988), pp. 213-231

[Pant 88b] Pantelides, C.C.:
SPEEDUP – Recent Advances in Process Simulation.
Computers and Chemical Engineering **12**, 7 (1988), pp. 745-755

[PBa 96] Park, T.; Barton, P.I.:
State Event Location in Differential-Algebraic Models.
ACM Transactions on Modeling and Computer Simulation **6**, 2 (1996), pp. 137-165
`http://yoric.mit.edu/cgi-bin/bartonpub`

[PBGM 65] Pontryagin, L.S.; Boltyanskii, V.G.; Gamkrelidze, R.V.; Mishchenko, E.F.:
The Mathematical Theory of Optimal Processes.
John Wiley & Sons, New York 1965, 3rd printing

[PeBu 94] Pesch, H.J.; Bulirsch, R.:
The Maximum Principle, Bellman's Equation, and Carathéodory's Work.
Journal of Optimization Theory and Applications **80**, 2 (1994), pp. 199-225

[PeLö 86] Petzold, L.; Lötstedt, P.:
Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints II: Practical Implications.
SIAM Journal on Scientific and Statistical Computing **7**, 3 (1986), pp. 720-733

[Pesc 89a] Pesch, H.J.:
Real-Time Computation of Feedback Controls for Constrained Optimal Control Problems. Part 1: Neighbouring Extremals.
Optimal Control, Applications & Methods **10**, 2 (1989), pp. 129-145

[Pesc 89b] Pesch, H.J.:
Real-Time Computation of Feedback Controls for Constrained Optimal Control Problems. Part 2: A Correction Method Based on Multiple Shooting.
Optimal Control, Applications & Methods **10**, 2 (1989), pp. 147-171

[Pesc 94] Pesch, H.J.:
A Practical Guide to the Solution of Real-Life Optimal Control Problems.
Control and Cybernetics **23**, 1/2 (1994), pp. 7-60

[Petz 82] Petzold, L.:
Differential/Algebraic Equations are not ODE's.
SIAM Journal on Scientific and Statistical Computing **3**, 3 (1982), pp. 367-384

[PGMS 88] Pantelides, C.C.; Gritsis, D.; Morison K.R.; Sargent, R.W.H.:
The Mathematical Modelling of Transient Systems Using Differential-Algebraic Equations.
Computers and Chemical Engineering **12**, 5 (1988), pp. 449-454

[PiVi 97] de Pinho, M.do R.; Vinter, R.B.:
Necessary Conditions for Optimal Control Problems Involving Nonlinear Differential Algebraic Equations.
Journal of Mathematical Analysis and Applications **212** (1997), pp. 493-516

[PoFa 90] Pothen, A.; Fan, C.-J.:
Computing the Block Triangular Form of a Sparse Matrix.
ACM Transactions on Mathematical Software **16**, 4 (1990), pp. 303-324

[Powe 78] Powell, M.J.D.:
A Fast Algorithm for Nonlinearly Constrained Optimization Calculations.
in: *Numerical Analysis*, Watson, G.A. (editor),
Lecture Notes in Mathematics **630**, Springer, Berlin 1978, pp. 144-157

[PRE 90] Patwardhan, A.A.; Rawlings, J.B.; Edgar, T.F.:
Nonlinear Model Predictive Control.
Chemical Engineering Communications **87** (1990), pp. 123-141

[PRGJP 97] Petzold, L.; Rosen, J.B.; Gill, P.E.; Jay, L.O.; Park, K.:
Numerical Optimal Control of Parabolic PDEs Using DASOPT.
in: *Large-Scale Optimization with Applications, Part II*, Biegler,

L.T.; Coleman, T.F.; Conn, A.R.; Santosa, F.N. (editors),
Springer, Berlin 1997, pp. 271-299
`http://www.engineering.ucsb.edu/~cse/`

[PSLRC 01] Petzold, L.; Serban, R.; Li, S.; Raha, S.; Cao, Y.:
Sensitivity Analysis and Design Optimization of Differential-Algebraic Equation Systems.
Proceedings of the NATO Advanced Research Workshop on Computational Aspects of Nonlinear Structural Systems with Large Rigid Body Motion, Pultusk, Poland (July 2 – 7, 2000), Ambrosio, J.A.C.; Kleiber, M. (editors),
IOS Press, Amsterdam 2001, pp. 153-168
`http://www.engineering.ucsb.edu/~radu/papers.html`

[QiBa 00] Qin, S.J.; Badgwell, T.A.:
An Overview of Nonlinear Model Predictive Control Applications.
in: *Progress in Systems and Control Theory, Vol.* ***26***, Allgöwer, F.; Zheng, A. (editors),
Proceedings of the International Symposium on Nonlinear Model Predictive Control – Assessment and Future Directions, Ascona, Switzerland (June 3-5, 1998),
Birkhäuser, Basel 2000, pp. 369-392
`http://www.che.utexas.edu/~qin/publicat.html`

[QiBa 97] Qin, S.J.; Badgwell, T.A.:
An Overview of Industrial Model Predictive Control Technology.
Proceedings of the Fifth International Conference of Chemical Process Control, Tahoe City, California (January 7 – 12, 1996), Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **316**, 93 (1997), pp. 232-256
`http://www.che.utexas.edu/~qin/publicat.html`

[Qin 97] Qin, S.J.:
Neural Networks for Intelligent Sensors and Control — Practical Issues and Some Solutions.
in: *Neural Systems for Control*, Omidvar, O.; Elliot, D.L. (editors),
Academic Press, London 1997, pp. 213-234
`http://www.che.utexas.edu/~qin/publicat.html`

[RaMu 93] Rawlings, J.B.; Muske, K.R.:
The Stability of Constrained Receding Horizon Control.
IEEE Transactions on Automatic Control **AC-38**, 10 (1993), pp. 1512-1516

[RaRa 00] Rao, C.V.; Rawlings, J.B.:
Nonlinear Moving Horizon State Estimation.

in: *Progress in Systems and Control Theory, Vol.* ***26***, Allgöwer, F.; Zheng, A. (editors),
Proceedings of the International Symposium on Nonlinear Model Predictive Control – Assessment and Future Directions, Ascona, Switzerland (June 3-5, 1998),
Birkhäuser, Basel 2000, pp. 45-69

[RaRh 94] Rabier, P.J.; Rheinboldt, W.C.:
A Geometric Treatment of Implicit Differential-Algebraic Equations.
Journal of Differential Equations **109** (1994), pp. 110-146

[RaRh 94a] Rabier, P.J.; Rheinboldt, W.C.:
On Impasse Points of Quasilinear Differential-Algebraic Equations.
Journal of Mathematical Analysis and Applications **181** (1994), pp. 429-454

[RaRh 94b] Rabier, P.J.; Rheinboldt, W.C.:
On the Computation of Impasse Points of Quasi-linear Differential-Algebraic Equations.
Mathematics of Computation **62**, 205 (1994), pp. 133-154

[Rawl 99] Rawlings, J.B.:
Tutorial: Model Predictive Control Technology.
Proceedings of the American Control Conference, San Diego, California (June 1999), pp. 662-676

[Reic 90] Reich, S.:
On a Geometrical Interpretation of Differential-Algebraic Equations.
Circuits, Systems and Signal Processing **9**, 4 (1990), pp. 367-382

[Rhei 84] Rheinboldt, W.C.:
Differential-Algebraic Systems as Differential Equations on Manifolds.
Mathematics of Computation **43**, 168 (1984), pp. 473-482

[Rick 91] Ricker, N.L.:
Model Predictive Control: State of the Art.
in: *Chemical Process Control CPC-IV*, Arkun, Y.; Ray, W.H. (editors),
Proceedings of the Fourth International Conference on Chemical Process Control, Padre Island, Texas (February 17 – 22, 1991), pp. 271-296

[RMB 00] Reißig, G.; Martinson, W.S.; Barton, P.I.:
Differential-Algebraic Equations of Index 1 May Have an Arbitrarily High Structural Index.

SIAM Journal on Scientific Computing **21**, 6 (2000), pp. 1987-1990
`http://epubs.siam.org/sam-bin/dbq/article/35385`

[Rohd 94] Rohde, W.:
Production of Pure Argon from Air.
Linde Reports on Science and Technology **54** (1994), pp. 3-7

[RoLu 91] Rosen, O.; Luus, R.:
Evaluation of Gradients for Piecewiese Constant Optimal Control.
Computers and Chemical Engineering **15**, 4 (1991), pp. 273-281

[Roze 67] Rozenvasser, E.:
General Sensitivity Equations of Discontinuous Systems.
Automation and Remote Control **28**, 3 (1967), pp. 400-404

[Sarg 78] Sargent, R.W.H.:
The Decomposition of Systems of Procedures and Algebraic Equations.
Proceedings of Numerical Analysis: Biennial Conference, Dundee (June 28 – July 1, 1977), Watson, G.A. (editor),
Lecture Notes in Mathematics **630** (1978), pp. 158-178

[Sawy 92] Sawyer, P.:
Simulate to Succeed.
The Chemical Engineer **26** (March 1992), pp. 28-30

[SBR 00] Seidel, U.; Buchta, H.; Reinschke, K.J.:
Variablen- und Parameterschätzung für Regelstreckenmodelle in Deskriptorform.
at – Automatisierungstechnik **48**, 12 (2000), pp. 602-610

[SBS 98] Schulz, V.H.; Bock, H.G.; Steinbach, M.C.:
Exploiting Invariants in the Numerical Solution of Multipoint Boundary Value Problems for DAEs.
SIAM Journal on Scientific Computing **19**, 2 (1998), pp. 440-467

[Schi 81a] Schittkowski, K.:
The Nonlinear Programming Method of Wilson, Han, and Powell with an Augmented Lagrangian Type Line Search Function; Part 1: Convergence Analysis.
Numerische Mathematik **38** (1981), pp. 83-114

[Schi 81b] Schittkowski, K.:
The Nonlinear Programming Method of Wilson, Han, and Powell with an Augmented Lagrangian Type Line Search Function; Part 2: An Efficient Implementation with Linear Least Squares Subproblems.
Numerische Mathematik **38** (1981), pp. 115-127

[Schl 88] Schlöder, J.P.:
Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung.
Alt, H.W.; Brieskorn, E.; Frehse, J.; Harder, G.; Hildebrandt, St.; Hirzebruch, F.; Klingenberg, W.; Leis, R.; Lieb, I.; Unger, H.; Vogel, W. (editors),
Bonner Mathematische Schriften Nr. **187**, Bonn, 1988

[Schw 99] von Schwerin, R.:
MultiBody System SIMulation: Numerical Methods, Algorithms and Software.
Lecture Notes in Computational Science and Engineering **7**,
Springer, Berlin 1999

[ScRa 97] Scokaert, P.O.M.; Rawlings, J.B.:
On Infeasibilities in Model Predictive Control.
Proceedings of the Fifth International Conference of Chemical Process Control, Tahoe City, California (January 7 – 12, 1996), Kantor, J.C.; Garcia, C.E.; Carnahan, B. (editors),
AIChE Symposium Series **316**, 93 (1997), pp. 331-334

[ScSc 99] van der Schaft, A.J.; Schumacher, J.M.:
An Introduction to Hybrid Dynamical Systems.
Lecture Notes in Control and Information Sciences **251**, Springer, London 1999

[ScWi 94] von Schwerin, R.; Winckler, M.:
A Guide to the Integrator Library MBSSIM – Version 1.00.
IWR-Preprint **94-75**, University of Heidelberg, 1994
`http://www.iwr.uni-heidelberg.de/~Michael.Winckler/`

[Sebo 99] Seborg, D.E.:
A Perspective on Advanced Strategies for Process Control.
atp – Automatisierungstechnische Praxis **41**, 11 (1999), pp. 13-31

[SEYE 81] Sincovec, R.F.; Erisman, A.M.; Yip, E.L.; Epton, M.A.:
Analysis of Descriptor Systems Using Numerical Algorithms.
IEEE Transactions on Automatic Control **AC-26**, 1 (1981), pp. 139-147

[Sham 87] Shampine, L.F.:
Starting Variable-Order ADAMS and BDF Codes.
Applied Numerical Mechanics **3** (1987), pp. 331-337

[Sörg 97] Sörgel, S.:
Application of a Modern Plant Optimization Package in Olefin

Plants.
Linde Reports on Science and Technology **59** (1997), pp. 3-11

[Soro 98] Soroush, M.:
State and Parameter Estimations and their Applications in Process Control.
Computers and Chemical Engineering **23** (1998), pp. 229-245

[StBu 90] Stoer, J.; Bulirsch, R.:
Numerische Mathematik 2.
Springer, Berlin 1990, 3rd edition

[StGl 01] von Stryk, O.; Glocker, M.:
Numerical Mixed-Integer Optimal Control and Motorized Traveling Salesmen Problems.
Journal européen des systèmes automatisés – European Journal of Control **35**, 4 (2001), pp. 519-533

[Stöc 93] Stöcker, H.:
Taschenbuch der Physik.
Verlag Harry Deutsch, Frankfurt am Main 1993

[Stoe 94] Stoer, J.:
Numerische Mathematik 1.
Springer, Berlin 1994, 7th edition

[StrBu 92] von Stryk, O.; Bulirsch, R.:
Direct and Indirect Methods for Trajectory Optimization.
Annals of Operations Research **37** (1992), pp. 357-373

[Stry 00] von Stryk, O.:
Numerical Hybrid Optimal Control and Related Topics.
Master's thesis, Technische Universität München, 2000

[Stry 94] von Stryk, O.:
Numerische Lösung optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung der adjungierten Variablen.
Ph.D. thesis, Technische Universität München, 1994

[SWS 95] Schwerin, R. von; Winckler, M.; Schulz, V.H.:
Parameter Estimation in Discontinuous Descriptor Models.
Proceedings of IUTAM Symposium on Optimization of Mechanical Systems, Stuttgart, Germany (March 26 – 31, 1995), Bestle, D.; Schiehlen, W. (editors),
Kluwer Academic Publishers, Dordrecht 1995, pp. 269-276

[Tarj 72] Tarjan, R.:
Depth-First Search and Linear Graph Algorithms.
SIAM Journal on Scientific Computing **1**, 2 (1972), pp. 146-160

[TCB 01] Tolsma, J.E.; Clabaugh, J.A.; Barton, P.I.:
Symbolic Incorporation of External Procedures into Process Modeling Environments.
preprint, September 24, 2001
`http://yoric.mit.edu/abacuss2/TCBIEC.pdf`

[Tisc 95] Tischendorf, C.:
Feasibility and Stability Behaviour of the BDF Applied to Index-2 Differential Algebraic Equations.
Zeitschrift für Angewandte Mathematik und Mechanik **75**, 12 (1995), pp. 927-946

[ToBa 02] Tolsma, J.E.; Barton, P.I.:
Hidden Discontinuities and Parametric Sensitivity Calculations.
SIAM Journal on Scientific Computing **23**, 6 (2002), pp. 1862-1875
`http://yoric.mit.edu/download/Papers/hybrid.pdf`

[UKM 95] Unger, J.; Kröner, A.; Marquardt, W.:
Structural Analysis of Differential-Algebraic Equations — Theory and Applications.
Computers and Chemical Engineering **19**, 8 (1995), pp. 867-882

[Unge 90] Unger, J.:
Structural Analysis of Differential-Algebraic Equation Systems.
Master's thesis, University of Wisconsin-Madison, December 1990

[UnMa 91] Unger, J.; Marquardt, W.:
Structural Analysis of Differential-Algebraic Equation Systems.
Proceedings of the European Symposium on Computer Applications in Chemical Engineering: Computer Oriented Process Engineering COPE-91, Barcelona, Spain (October 14 – 16, 1991), Puigjaner, L.; Espuña, A. (editors),
Elsevier Science, Amsterdam 1991

[VBB 99] Vassiliadis, V.S.; Balsa Canto, E.; Banga, J.R.:
Second-Order Sensitivities of General Dynamic Systems with Application to Optimal Control Problems.
Chemical Engineering Science **54**, 17 (1999), pp. 3851-3860

[ViBi 01] Vieira, R.C.; Biscaia Jr., E.C.:
Direct Methods for Consistent Initialization of DAE systems.
Computers and Chemical Engineering **25**, 9-10 (2001), pp. 1299-1311

[Voit 94] Voit, J.:
The Production of Computer-based Plant Optimization Systems with the OPTISIM Process Simulator: Experience with a Linde Air Separation Plant.
Linde Reports on Science and Technology **54** (1994), pp. 19-25

[VSP 94a] Vassiliadis, V.S.; Sargent, R.W.H.; Pantelides C.C.:
Solution of a Class of Multistage Dynamic Optimization Problems: 1. Problems without Path Constraints.
Industrial & Engineering Chemistry Research **33**, 9 (1994), pp. 2111-2122

[VSP 94b] Vassiliadis, V.S.; Sargent, R.W.H.; Pantelides C.C.:
Solution of a Class of Multistage Dynamic Optimization Problems: 2. Problems with Path Constraints.
Industrial & Engineering Chemistry Research **33**, 9 (1994), pp. 2123-2133

[Watt 83] Watts, H.A.:
Starting Step Size for an ODE Solver.
Journal of Computational and Applied Mathematics **9** (1983), pp. 177-191

[Wrig 02] Wright, S.J.:
Optimization Software Packages.
in: *Handbook of Applied Optimization*, Resende, M.; Pardalos, P. (editors),
Oxford University Press, Oxford 2002
`http://www.cs.wisc.edu/~swright/papers/`

[WuWh 01] Wu, B.; White, R.E.:
An Initialization Subroutine for DAEs Solvers: DAEIS.
Computers and Chemical Engineering **25**, 2-3 (2001), pp. 301-311

[YaPo 93] Yang, T.H.; Polak, E.:
Moving Horizon Control of Nonlinear Systems with Input Saturation, Disturbances and Plant Uncertainty.
International Journal of Control **58**, 4 (1993), pp. 875-903

[YiSi 81] Yip, E.L.; Sincovec, R.F.:
Solvability, Controllability, and Observability of Continuous Descriptor Systems.
IEEE Transactions on Automatic Control **AC-26**, 3 (1981), pp. 702-707

[Zapp 94] Zapp, G.:
Dynamic Simulation of Air Separation Plants and the Use of "Relative Gain Analysis" for Design of Control Systems.
Linde Reports on Science and Technology **54** (1994), pp. 13-18

Remark 8.1:
In some bibliographic entries a reference to the location of the respective document in the Internet (URL) is given. We provide these URLs according to our present knowledge. Please note that the URLs may be subject to unnotified change; it as also possible that the access to the document may be entirely denied in the future. ◊

Weiner's Law of Libraries:
There are no answers, only cross references.

FORTUNE cookie