A Survey on Numerical Methods for the Simulation of Initial Value Problems with sDAEs

Michael Burger and Matthias Gerdts

Abstract This paper provides an overview on numerical aspects in the simulation of differential-algebraic equations (DAEs). Amongst others we discuss the basic construction principles of frequently used discretization schemes, such as BDF methods, Runge–Kutta methods, and ROW methods, as well as their adaption to DAEs. Moreover, topics like consistent initialization, stabilization, parametric sensitivity analysis, co-simulation techniques, aspects of real-time simulation, and contact problems are covered. Finally, some illustrative numerical examples are presented.

Keywords BDF methods • Consistent initialization • Contact problems • Co-simulation • Differential-algebraic equations • Real-time simulation • ROW methods • Runge–Kutta methods • Sensitivity analysis • Stabilization

Subject Classifications: 65L80, 65L05, 65L06

1 Introduction

Simulation is a well-established and indispensable tool in scientific research as well as in industrial development processes. Efficient tools are needed that are capable of simulating complex processes in, e.g., mechanical engineering, process engineering, or electrical engineering. Many of such processes (where appropriate after a spatial discretization of a partial differential equation) can be modeled as

M. Burger (✉)
Department Mathematical Methods in Dynamics and Durability MDF, Fraunhofer Institute for Industrial Mathematics ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
e-mail: Michael.Burger@itwm.fraunhofer.de

M. Gerdts
Department of Aerospace Engineering, Institute of Mathematics and Applied Computing, Universität der Bundeswehr München, Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
e-mail: matthias.gerdts@unibw.de

differential-algebraic equations (DAEs), which are implicit differential equations that typically consist of ordinary differential equations as well as algebraic equations. Often, DAEs are formulated automatically by software packages such as MODELICA or SIMPACK. In its general form, the initial value problem for a DAE on the compact interval $I = [a, b]$ reads as

$$F(t, z(t), z'(t)) = 0, \qquad z(a) = z_a, \tag{1.1}$$

where $F : I \times \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is a given function and $z_a \in \mathbb{R}^n$ is an appropriate initial value at $t = a$. The task is to find a solution $z : I \longrightarrow \mathbb{R}^n$ of (1.1). Throughout it is assumed that F is sufficiently smooth, i.e., it possesses all the continuous partial derivatives up to a requested order.

Please note that (1.1) is not just an ordinary differential equation in implicit notation, since we permit the Jacobian of F with respect to z', i.e., $F'_{z'}$, to be *singular* along a solution. In such a situation, (1.1) cannot be solved directly for z'. Particular examples with singular Jacobian are semi-explicit DAEs of type

$$F(t, z, z') = \begin{pmatrix} M(t, x)x' - f(t, x, y) \\ g(t, x, y) \end{pmatrix}, \qquad z := (x, y)^\top, \tag{1.2}$$

with a non-singular matrix M and the so-called differential state vector x and the algebraic state vector y. Such systems occur, e.g., in process engineering and mechanical multi-body systems. More generally, quasi-linear DAEs of type

$$F(t, z, z') = Q(t, z)z' - f(t, z)$$

with a possibly singular matrix function Q frequently occur in electrical engineering.

The potential singularity of the Jacobian $F'_{z'}$ has implications with regard to theoretical properties (existence and uniqueness of solutions, smoothness properties, structural properties, …) and with regard to the design of numerical methods (consistent initial values, order of convergence, stability, …). A survey on the solution theory for linear DAEs can be found in the recent survey paper [141]. A comprehensive structural analysis of linear and nonlinear DAEs can be found in the monographs [90] and [92]. While explicit ordinary differential equations (ODEs) can be viewed as well-behaved systems, DAEs are inherently ill-conditioned and the degree of ill-conditioning increases with the so-called (perturbation) index, compare [75, Definition 1.1]. As such, DAEs require suitable techniques for its numerical treatment.

To this end, the paper aims to provide an overview on the numerical treatment of the initial value problem. The intention is to cover the main ideas without too many technical details, which, if required, can be found in full detail in a huge number of publications and excellent textbooks. Naturally not all developments can be covered, so we focus on a choice of methods and concepts that are relevant in industrial simulation environments for coupled systems of potentially large size.

These concepts enhance basic integration schemes by adding features like sensitivity analysis (needed, e.g., in optimization procedures), contact dynamics, real-time schemes, or co-simulation techniques. Still, the core challenges with DAEs, that is ill-conditioning, consistent initial values, index reduction, will be covered as well.

The outline of this paper is as follows. Section 2 introduces index concepts and summarizes stabilization techniques for certain classes of DAEs. Section 3 deals with the computation of the so-called consistent initial values for DAEs and their influence on parameters. Note in this respect that DAEs, in contrast to ODEs, do not permit solutions for arbitrary initial values and thus techniques are required to find suitable initial values. The basics of the most commonly used numerical discretization schemes are discussed in Sect. 4, amongst them are BDF methods, Runge–Kutta methods, and ROW methods. Co-simulation techniques for the interaction of different subsystems are presented in Sect. 5. Herein, the stability and convergence of the overall scheme are of particular importance. Section 6 discusses approaches for the simulation of time crucial systems in real-time. The influence of parameters on the (discrete and continuous) solution of an initial value problem is studied in Sect. 7. Hybrid systems and mechanical contact problems are discussed in Sect. 8.

Notation

We use the following notation. The derivative w.r.t. time of a function $z(t)$ is denoted by $z'(t)$. The partial derivative of a function f with respect to a variable x will be denoted by $f'_x = \partial f / \partial x$. As an abbreviation of a function of type $f(t, x(t))$ we use the notation $f[t]$.

2 Error Influence and Stabilization Techniques

DAEs are frequently characterized and classified according to its index. Various index definitions exist, for instance the differentiation index [62], the structural index [45], the strangeness index [90], the tractability index [92], and the perturbation index [75]. These index definitions are not equivalent for general DAEs (1.1), but they coincide for certain subclasses thereof, for instance semi-explicit DAEs in Hessenberg form. For our purposes we will focus on the differentiation index and the perturbation index only.

The *differentiation index* is one of the earliest index definitions for (1.1) and is based on a structural investigation of the DAE. It aims to identify the so-called underlying ordinary differential equation. To this end let the functions $F^{(j)} : [t_0, t_f] \times R^{(j+2)n} \longrightarrow \mathbb{R}^n$ for the variables $z, z', \ldots, z^{(j+1)} \in \mathbb{R}^n$ for $j = 0, 1, 2, \ldots$ be defined by the recursion

$$F^{(0)}(t,z,z') := F(t,z,z'), \tag{2.1}$$

$$F^{(j)}(t,z,z',\ldots,z^{(j+1)}) := \frac{\partial F^{(j-1)}}{\partial t}(t,z,z',\ldots,z^{(j)}) \tag{2.2}$$

$$+\sum_{\ell=0}^{j}\frac{\partial F^{(j-1)}}{\partial z^{(\ell)}}(t,z,z',\ldots,z^{(j)})z^{(\ell+1)}, \quad j=1,2,\ldots. \tag{2.3}$$

Herein, F is supposed to be sufficiently smooth such that the functions $F^{(j)}$ are well defined.

The differentiation index is defined as follows:

Definition 2.1 (Differentiation Index, Compare [62]) The DAE (1.1) has *differentiation index* $d \in \mathbb{N}_0$, if d is the smallest number in $\mathbb{N}_0$ such that the so-called derivative array

$$F^{(j)}(t,z,z',\ldots,z^{(j+1)}) = 0, \qquad j=0,1,\ldots,d, \tag{2.4}$$

allows to deduce a relation of type $z' = f(t,z)$ by algebraic manipulations.

If such a relation exists, then the corresponding ordinary differential equation (ODE) $z'(t) = f(t,z(t))$ is called the *underlying ODE* of the DAE (1.1).

The definition leaves some space for interpretation as it is not entirely clear what is meant by "algebraic manipulations." However, for semi-explicit DAEs it provides a guideline to determine the differentiation index. Note that the special structure of semi-explicit DAEs is often exploited in the design of numerical schemes and stabilization techniques.

Definition 2.2 (Semi-Explicit DAE) A DAE of type

$$x'(t) = f(t,x(t),y(t)), \tag{2.5}$$

$$0 = g(t,x(t),y(t)), \tag{2.6}$$

is called *semi-explicit DAE*. Herein, $x(\cdot)$ is referred to as *differential variable* and $y(\cdot)$ is called *algebraic variable*. Correspondingly, (2.5) is called *differential equation* and (2.6) *algebraic equation*.

For semi-explicit DAEs the common approach is to differentiate the algebraic equation w.r.t. time and to substitute the occurring derivatives of x by the right-hand side of the differential equation. This procedure is repeated until the resulting equation can be solved for y'.

Example 2.1 (Semi-Explicit DAE with Differentiation Index One) Consider (2.5)–(2.6). Differentiation of the algebraic equation w.r.t. time yields

$$\begin{aligned} 0 &= g_t'[t] + g_x'[t]x'(t) + g_y'[t]y'(t) \\ &= g_t'[t] + g_x'[t]f[t] + g_y'[t]y'(t). \end{aligned}$$

Herein, we used the abbreviation $f[t]$ for $f(t, x(t), y(t))$ and likewise for the partial derivatives of g.

Now, if the Jacobian matrix $g_y'[t]$ is non-singular with a bounded inverse along a solution of the DAE, then the above equation can be solved for y' by the implicit function theorem and together with the differential equation (2.5) we obtain the underlying ODE

$$\begin{aligned} x'(t) &= f(t, x(t), y(t)), \\ y'(t) &= -g_y'[t]^{-1}\left(g_t'[t] + g_x'[t]f[t]\right), \end{aligned}$$

and the differentiation index is $d = 1$.

In the above example, the situation becomes more involved, if the Jacobian matrix $g_y'[t]$ is singular. If it actually vanishes, then one can proceed as in the following example.

Example 2.2 (Semi-Explicit DAE with Differentiation Index Two) Consider (2.5)–(2.6). Suppose g does not depend on y and thus $g_y'[t] \equiv 0$. By differentiation of the algebraic equation we obtain

$$0 = g_t'[t] + g_x'[t]x'(t) = g_t'[t] + g_x'[t]f[t] =: g^{(1)}(t, x(t), y(t))$$

A further differentiation w.r.t. time yields

$$0 = (g^{(1)})_t'[t] + (g^{(1)})_x'[t]f[t] + (g^{(1)})_y'[t]y'(t)$$

with $(g^{(1)})_y'[t] = g_x'[t]f_y'[t]$. Now, if the matrix $g_x'[t]f_y'[t]$ is non-singular with a bounded inverse along a solution of the DAE, then the above equation can be solved for y' by the implicit function theorem and together with the differential equation (2.5) we obtain the underlying ODE

$$\begin{aligned} x'(t) &= f(t, x(t), y(t)), \\ y'(t) &= -(g_x'[t]f_y'[t])^{-1}\left((g^{(1)})_t'[t] + (g^{(1)})_x'[t]f[t]\right), \end{aligned}$$

and the differentiation index is $d = 2$.

The procedure of the preceding examples works for semi-explicit Hessenberg DAEs, which are defined as follows:

Definition 2.3 (Hessenberg DAE)

(a) For a given $k \geq 2$ the DAE

$$\begin{aligned}
x_1'(t) &= f_1(t,\ y(t),\ x_1(t),\ x_2(t),\ \dots,\ x_{k-2}(t),\ x_{k-1}(t)),\\
x_2'(t) &= f_2(t,\ \ x_1(t),\ x_2(t),\ \dots,\ x_{k-2}(t),\ x_{k-1}(t)),\\
&\ \ \vdots \qquad\qquad\qquad \ddots\\
x_{k-1}'(t) &= f_{k-1}(t,\ x_{k-2}(t),\ x_{k-1}(t)),\\
0 &= g(t,\ x_{k-1}(t))
\end{aligned} \tag{2.7}$$

is called *Hessenberg DAE of order k*, if the matrix

$$R(t) := g_{x_{k-1}}'[t] \cdot f_{k-1,x_{k-2}}'[t] \cdots f_{2,x_1}'[t] \cdot f_{1,y}'[t] \tag{2.8}$$

is non-singular for all $t \in [t_0, t_f]$ with a uniformly bounded inverse $\|R^{-1}(t)\| \leq C$ in $[t_0, t_f]$, where C is a constant independent of t.

(b) The DAE

$$\begin{aligned}
x'(t) &= f(t, x(t), y(t)),\\
0 &= g(t, x(t), y(t))
\end{aligned} \tag{2.9}$$

is called *Hessenberg DAE of order* 1, if the matrix $g_y'[t]$ is non-singular with $\|g_y'[t]^{-1}\| \leq C$ for all $t \in [t_0, t_f]$ and some constant C independent of t.

Herein, y is called algebraic variable and $x = (x_1, \dots, x_{k-1})^\top$ in (a) and x in (b), respectively, is called differential variable.

By repeated differentiation of the algebraic constraint $0 = g(t, x_{k-1}(t))$ w.r.t. to time and simultaneous substitution of the derivatives of the differential variable by the corresponding differential equations, it is straightforward to show that the differentiation index of a Hessenberg DAE of order k is equal to k, provided the functions g and f_j, $j = 1, \dots, k-1$, are sufficiently smooth. In order to formalize this procedure, define

$$g^{(0)}(t, x_{k-1}(t)) := g(t, x_{k-1}(t)). \tag{2.10}$$

Differentiation of $g^{(0)}$ with respect to time and substitution of

$$x_{k-1}'(t) = f_{k-1}(t, x_{k-2}(t), x_{k-1}(t))$$

leads to the equation

$$\begin{aligned} 0 \;&=\; g'_t(t,x_{k-1}(t)) + g'_{x_{k-1}}(t,x_{k-1}(t)) \cdot f_{k-1}(t,x_{k-2}(t),x_{k-1}(t)) \\ &=: g^{(1)}(t,x_{k-2}(t),x_{k-1}(t)), \end{aligned}$$

which is satisfied implicitly as well. Recursive application of this differentiation and substitution process leads to the algebraic equations

$$0 = g^{(j)}(t,x_{k-1-j}(t),\ldots,x_{k-1}(t)), \qquad j = 1,2,\ldots,k-2, \tag{2.11}$$

and

$$0 = g^{(k-1)}(t,y(t),x_1(t),\ldots,x_{k-1}(t)). \tag{2.12}$$

Since Eqs. (2.11)–(2.12) do not occur explicitly in the original system (2.7), these equations are called *hidden constraints* of the Hessenberg DAE. Note that the matrix R in (2.8) is given by $\partial g^{(k-1)}/\partial y$.

A practically important subclass of Hessenberg DAEs are mechanical multibody systems in descriptor form given by

$$\begin{aligned} q'(t) &= v(t), \\ M(t,q(t))v'(t) &= f(t,q(t),v(t)) - g'_q(t,q(t))^\top \lambda(t), \\ 0 &= g(t,q(t)), \end{aligned} \tag{2.13}$$

where $q(\cdot) \in \mathbb{R}^n$ denotes the vector of generalized positions, $v(\cdot) \in \mathbb{R}^n$ the vector of generalized velocities, and $\lambda(\cdot) \in \mathbb{R}^m$ are Lagrange multipliers. The mass matrix M is supposed to be symmetric and positive definite with a bounded inverse M^{-1} and thus, the second equation in (2.13) can be multiplied by $M(t,q(t))^{-1}$. The vector f denotes the generalized forces and torques. The term $g'_q(t,q)^\top\lambda$ can be interpreted as a force that keeps the system on the algebraic constraint $g(t,q) = 0$.

The constraint $g(t,q(t)) = 0$ is called *constraint on position level*. Differentiation with respect to time of this algebraic constraint yields the *constraint on velocity level*

$$g'_t(t,q(t)) + g'_q(t,q(t)) \cdot v(t) = 0$$

and the *constraint on acceleration level*

$$g''_{tt}(t,q(t)) + g''_{tq}(t,q(t)) \cdot v(t) + g'_q(t,q(t)) \cdot v'(t) + g''_{qq}(t,q(t))(v(t),v(t)) = 0.$$

Replacing v' by

$$v'(t) = M(t,q(t))^{-1}\left(f(q(t),v(t)) - g'_q(t,q(t))^\top \lambda(t)\right)$$

yields

$$\begin{aligned}0 = {} & g''_{tt}(t,q(t)) + g''_{tq}(t,q(t)) \cdot v(t) \\ & + g'_q(t,q(t)) \cdot M(t,q(t))^{-1} \left(f(q(t),v(t)) - g'_q(t,q(t))^\top \lambda(t)\right) \\ & + g''_{qq}(t,q(t))(v(t),v(t)).\end{aligned}$$

If $g'_q(t,q))$ has full rank, then the matrix $g'_q(t,q)M(t,q)^{-1}g'_q(t,q)^\top$ is non-singular and the latter equation can be solved for the algebraic variable λ. Thus, the differentiation index is three.

Remark 2.1 Note that semi-explicit DAEs are more general than Hessenberg DAEs since no regularity assumptions are imposed in Definition 2.2. In fact, without additional regularity assumptions, the class of semi-explicit DAEs is essentially as large as the class of general DAEs (1.1), since the settings $z'(t) = y(t)$ and $F(t, y(t), z(t)) = 0$ transform the DAE (1.1) into a semi-explicit DAE (some care has to be taken with regard to the smoothness of solutions, though).

2.1 Error Influence and Perturbation Index

The differentiation index is based on a structural analysis of the DAE, but it does not indicate how perturbations influence the solution. In contrast, the perturbation index addresses the influence of perturbations on the solution and thus it is concerned with the *stability of DAEs*. Note that perturbations frequently occur, for instance they are introduced by numerical discretization schemes.

Definition 2.4 (Perturbation Index, See [75]) The DAE (1.1) has *perturbation index* $p \in \mathbb{N}$ along a solution z on $[t_0, t_f]$, if $p \in \mathbb{N}$ is the smallest number such that for all functions $\tilde{z}$ satisfying the perturbed DAE

$$F(t, \tilde{z}(t), \tilde{z}'(t)) = \delta(t), \tag{2.14}$$

there exists a constant S depending on F and $t_f - t_0$ with

$$\|z(t) - \tilde{z}(t)\| \leq S \left(\|z(t_0) - \tilde{z}(t_0)\| + \max_{t_0 \leq \tau \leq t} \|\delta(\tau)\| + \ldots + \max_{0 \leq \tau \leq t} \|\delta^{(p-1)}(\tau)\| \right) \tag{2.15}$$

for all $t \in [t_0, t_f]$, whenever the expression on the right is less than or equal to a given bound.

The *perturbation index* is $p = 0$, if the estimate

$$\|z(t) - \tilde{z}(t)\| \leq S \left(\|z(t_0) - \tilde{z}(t_0)\| + \max_{t_0 \leq \tau \leq t_f} \left\| \int_{t_0}^{\tau} \delta(s) ds \right\| \right) \tag{2.16}$$

holds. The DAE is said to be of *higher index*, if $p \geq 2$.

According to the definition of the perturbation index, higher index DAEs are ill-conditioned in the sense that small perturbations with high frequencies, i.e., with large derivatives, can have a considerable influence on the solution of a higher index DAE as it can be seen in (2.15). For some time it was believed that the difference between perturbation index and differentiation index is at most one, until it was shown in [34] that the difference between perturbation index and differentiation index can be arbitrarily large. However, for the subclass of Hessenberg DAEs as defined in Definition 2.3 both index concepts (and actually all other relevant index concepts) coincide.

The definition of the perturbation index shows that the degree of ill-conditioning increases with the perturbation index. Hence, in order to make a higher index DAE accessible to numerical methods it is advisable and common practice to reduce the perturbation index of a DAE. A straightforward idea is to replace the original DAE by a mathematically equivalent DAE with lower perturbation index. The index reduction process itself is nontrivial for general DAEs, since one has to ensure that it is actually the perturbation index, which is being reduced (and not some other index like the differentiation index).

For Hessenberg DAEs, however, the index reduction process is straightforward as perturbation index and differentiation index coincide. Consider a Hessenberg DAE of order k as in (2.7). Then, by replacing the algebraic constraint $0 = g(t, x_{k-1}(t))$ by one of the hidden constraints $g^{(j)}, j \in \{1, \ldots, k-1\}$, defined in (2.11) or (2.12) we obtain the Hessenberg DAE

$$\begin{aligned}
x_1'(t) &= f_1(t, y(t), x_1(t), x_2(t), \ldots, x_{k-2}(t), x_{k-1}(t)), \\
x_2'(t) &= f_2(t, x_1(t), x_2(t), \ldots, x_{k-2}(t), x_{k-1}(t)), \\
&\vdots \qquad \ddots \\
x_{k-1}'(t) &= f_{k-1}(t, x_{k-2}(t), x_{k-1}(t)), \\
0 &= g^{(j)}(t, x_{k-1-j}(t), \ldots, x_{k-1}(t)),
\end{aligned} \tag{2.17}$$

where we use the setting $x_0 := y$ for notational convenience. The Hessenberg DAE in (2.17) has perturbation index $k - j$. Hence, this simple index reduction strategy actually reduces the perturbation index, and it leads to a mathematically equivalent DAE with the same solution as the original DAE, if the initial values $x(t_0)$ and $y(t_0)$ satisfy the algebraic constraints $g^{(\ell)}(t_0, x_{k-1-\ell}(t_0), \ldots, x_{k-1}(t_0)) = 0$ for all $\ell = 0, \ldots, k-1$.

On the other hand, the index reduced DAE (2.17) in general permits additional solutions for those initial values $x(t_0)$ and $y(t_0)$, which merely satisfy the algebraic constraints $g^{(\ell)}(t_0, x_{k-1-\ell}(t_0), \ldots, x_{k-1}(t_0)) = 0$ for all $\ell = j, \ldots, k-1$, but not the neglected algebraic constraints with index $\ell = 0, \ldots, j-1$. In the most extreme case $j = k-1$ (the reduced DAE has index-one) $x(t_0)$ can be chosen arbitrarily (assuming that the remaining algebraic constraint can be solved for $y(t_0)$ given the value of $x(t_0)$). The following theorem shows that the use of inconsistent initial values leads to a polynomial drift off the neglected algebraic constraints in time, compare [73, Sect. VII.2].

Theorem 2.1 *Consider the Hessenberg DAE of order k in (2.7) and the index reduced DAE in (2.17) with $j \in \{1, \ldots, k-1\}$. Let $x(t)$ and $y(t)$ be a solution of (2.17) such that the initial values $x(t_0)$ and $y(t_0)$ satisfy the algebraic constraints $g^{(\ell)}(t_0, x_{k-1-\ell}(t_0), \ldots, x_{k-1}(t_0)) = 0$ for all $\ell = j, \ldots, k-1$. Then for $\ell = 1, \ldots, j$ and $t \geq t_0$ we have*

$$g^{(j-\ell)}(t, x_{k-1-(j-\ell)}(t), \ldots, x_{k-1}(t)) = \sum_{\nu=0}^{\ell-1} \frac{1}{\nu!}(t-t_0)^{\nu} g^{(j-\ell+\nu)}[t_0]. \tag{2.18}$$

with $g^{(j-\ell+\nu)}[t_0] := g^{(j-\ell+\nu)}(t_0, x_{k-1-(j-\ell+\nu)}(t_0), \ldots, x_{k-1}(t_0))$.

Proof We use the abbreviation $g^{(\ell)}[t]$ for $g^{(\ell)}(t, x_{k-1-\ell}(t), x_{k-1}(t))$ for notational convenience. Observe that

$$g^{(j-\ell+1)}[t] = \frac{d}{dt} g^{(j-\ell)}[t], \qquad \ell = 1, \ldots, j,$$

and thus

$$g^{(j-\ell)}[t] = g^{(j-\ell)}[t_0] + \int_{t_0}^{t} g^{(j-\ell+1)}[\tau] d\tau.$$

We have $g^{(j)}[t] = 0$ and thus for $\ell = 1$:

$$g^{(j-1)}[t] = g^{(j-1)}[t_0] + \int_{t_0}^{t} g^{(j)}[\tau] d\tau = g^{(j-1)}[t_0].$$

This proves (2.18) for $\ell = 1$. Inductively we obtain

$$\begin{aligned} g^{(j-(\ell+1))}[t] &= g^{(j-(\ell+1))}[t_0] + \int_{t_0}^{t} g^{(j-\ell)}[\tau] d\tau \\ &= g^{(j-(\ell+1))}[t_0] + \int_{t_0}^{t} \sum_{\nu=0}^{\ell-1} \frac{1}{\nu!}(\tau - t_0)^{\nu} g^{(j-\ell+\nu)}[t_0] d\tau \end{aligned}$$

$$= g^{(j-(\ell+1))}[t_0] + \sum_{\nu=0}^{\ell-1} \frac{1}{(\nu+1)!}(t-t_0)^{\nu+1} g^{(j-\ell+\nu)}[t_0]$$

$$= g^{(j-(\ell+1))}[t_0] + \sum_{\nu=1}^{\ell} \frac{1}{\nu!}(t-t_0)^{\nu} g^{(j-\ell+\nu-1)}[t_0]$$

$$= \sum_{\nu=0}^{\ell} \frac{1}{\nu!}(t-t_0)^{\nu} g^{(j-(\ell+1)+\nu)}[t_0],$$

which proves the assertion. □

We investigate the practically relevant index-three case in more detail and consider the reduction to index one (i.e., $k = 3$ and $j = 2$). In this case Theorem 2.1 yields

$$g^{(0)}(t, x_2(t)) = g^{(0)}[t_0] + (t-t_0)g^{(1)}[t_0], \tag{2.19}$$

$$g^{(1)}(t, x_1(t), x_2(t)) = g^{(1)}[t_0]. \tag{2.20}$$

The drift-off property of the index reduced DAE causes difficulties for numerical discretization methods as the subsequent result shows, compare [73, Sect. VII.2].

Theorem 2.2 *Consider the DAE (2.7) with $k = 3$ and the index reduced problem (2.17) with $j = 2$. Let $z(t; t_m, z_m)$ denote the solution of the latter at time t with initial value z_m at t_m, where $z = (x_1, x_2, y)^\top$ denotes the vector of differential and algebraic states. Suppose the initial value z_0 at t_0 satisfies $g^{(0)}[t_0] = 0$ and $g^{(1)}[t_0] = 0$.*

Let a numerical method generate approximations $z_n = (x_{1,n}, x_{2,n}, y_n)^\top$ of $z(t_n; t_0, z_0)$ at time points $t_n = t_0 + nh$, $n \in \mathbb{N}_0$, with stepsize $h > 0$. Suppose the numerical method is of order $p \in \mathbb{N}$, i.e., the local error satisfies

$$\|z_{n+1} - z(t_{n+1}; t_n, z_n)\| = \mathcal{O}(h^{p+1}), \qquad n \in \mathbb{N}_0.$$

Then, for $n \in \mathbb{N}$ the algebraic constraint $g^{(0)} = g$ satisfies the estimate

$$\|g(t_n, x_{2,n})\| \le Ch^p \left(L_0(t_n - t_0) + \frac{L_1}{2}(t_n - t_0)^2 \right) \tag{2.21}$$

with constants C, L_0, and L_1.

Proof Since z_0 satisfies $g^{(0)}[t_0] = 0$ and $g^{(1)}[t_0] = 0$, the solution $z(t; t_0, z_0)$ satisfies these constraints for every t. For notational convenience we use the notion $g^{(0)}(t, z(t))$ instead of $g^{(0)}(t, x_2(t))$ and likewise for $g^{(1)}$. To this end, for a given t_n

we have

$$\begin{aligned}
\|g^{(0)}(t_n,z_n)\| &= \|g^{(0)}(t_n,z_n) - g^{(0)}(t_n,z(t_n;t_0,z_0))\| \\
&= \|\sum_{m=0}^{n-1}\left(g^{(0)}(t_n,z(t_n;t_{m+1},z_{m+1})) - g^{(0)}(t_n,z(t_n;t_m,z_m))\right)\| \\
&\le \sum_{m=0}^{n-1}\|g^{(0)}(t_n,z(t_n;t_{m+1},z_{m+1})) - g^{(0)}(t_n,z(t_n;t_m,z_m))\|. \qquad (2.22)
\end{aligned}$$

Exploitation of (2.19)–(2.20) with t_0 replaced by t_m and t_{m+1}, respectively, yields

$$\begin{aligned}
&\|g^{(0)}(t_n,z(t_n;t_{m+1},z_{m+1})) - g^{(0)}(t_n,z(t_n;t_m,z_m))\| \\
&= \|g^{(0)}(t_{m+1},z_{m+1}) + (t_n-t_{m+1})g^{(1)}(t_{m+1},z_{m+1}) \\
&\qquad - g^{(0)}(t_m,z_m) - (t_n-t_m)g^{(1)}(t_m,z_m)\| \\
&= \|g^{(0)}(t_{m+1},z_{m+1}) + (t_n-t_{m+1})g^{(1)}(t_{m+1},z_{m+1}) - g^{(0)}(t_{m+1},z(t_{m+1};t_m,z_m)) \\
&\qquad + g^{(0)}(t_{m+1},z(t_{m+1};t_m,z_m)) - g^{(0)}(t_m,z_m) - (t_n-t_m)g^{(1)}(t_m,z_m)\| \\
&= \|g^{(0)}(t_{m+1},z_{m+1}) + (t_n-t_{m+1})g^{(1)}(t_{m+1},z_{m+1}) - g^{(0)}(t_{m+1},z(t_{m+1};t_m,z_m)) \\
&\qquad + g^{(0)}(t_m,z_m) + (t_{m+1}-t_m)g^{(1)}(t_m,z_m) - g^{(0)}(t_m,z_m) - (t_n-t_m)g^{(1)}(t_m,z_m)\| \\
&= \|g^{(0)}(t_{m+1},z_{m+1}) - g^{(0)}(t_{m+1},z(t_{m+1};t_m,z_m)) \\
&\qquad + (t_n-t_{m+1})\left(g^{(1)}(t_{m+1},z_{m+1}) - g^{(1)}(t_m,z_m)\right)\| \\
&\le L_0Ch^{p+1} + (t_n-t_{m+1})\|g^{(1)}(t_{m+1},z_{m+1}) - g^{(1)}(t_{m+1},z(t_{m+1};t_m,z_m))\| \\
&\qquad + (t_n-t_{m+1})\|g^{(1)}(t_{m+1},z(t_{m+1};t_m,z_m)) - g^{(1)}(t_m,z_m)\| \\
&\le Ch^{p+1}\left(L_0 + L_1(t_n-t_{m+1})\right) + (t_n-t_{m+1})\|g^{(1)}(t_m,z_m) - g^{(1)}(t_m,z_m)\| \\
&= Ch^{p+1}\left(L_0 + L_1(t_n-t_{m+1})\right),
\end{aligned}$$

where L_0 and L_1 are Lipschitz constants of $g^{(0)}$ and $g^{(1)}$. Together with (2.22) we thus proved the estimate

$$\begin{aligned}
\|g^{(0)}(t_n,z_n)\| &\le \sum_{m=0}^{n-1} Ch^{p+1}\left(L_0 + L_1(t_n-t_{m+1})\right) \\
&\le Ch^p\left(L_0(t_n-t_0) + \frac{L_1}{2}(t_n-t_0)^2\right).
\end{aligned}$$

□

The estimate (2.21) shows that the numerical solution may violate the algebraic constraint with a quadratic drift term in t_n for the setting in Theorem 2.2. This drift-off effect may lead to useless numerical simulation results, especially on long time horizons. For DAEs with even higher index, the situation becomes worse as the degree of the polynomial drift term depends on the j in (2.17), i.e., on the number of differentiations used in the index reduction.

2.2 Stabilization Techniques

The basic index reduction approach in the previous section may lead to unsatisfactory numerical results. One possibility to avoid the drift-off on numerical level is to perform a projection step onto the neglected algebraic constraints after each successful integration step for the index reduced system, see [18, 47].

Another idea is to use stabilization techniques to stabilize the index reduced DAE itself. The common approaches are Baumgarte stabilization, Gear–Gupta–Leimkuhler stabilization, and the use of overdetermined DAEs.

2.2.1 Baumgarte Stabilization

The Baumgarte stabilization [22] was originally introduced for mechanical multibody systems (2.13). It can be extended to Hessenberg DAEs in a formal way. The idea is to replace the algebraic constraint in (2.7) by a linear combination of original and hidden algebraic constraints $g^{(\ell)}$, $\ell \in \{0, 1, \ldots, k-1\}$. With the setting $x_0 := y$, the resulting DAE reads as follows:

$$\begin{aligned}
x_1'(t) &= f_1(t, y(t), x_1(t), x_2(t), \ldots, x_{k-2}(t), x_{k-1}(t)),\\
x_2'(t) &= f_2(t, x_1(t), x_2(t), \ldots, x_{k-2}(t), x_{k-1}(t)),\\
&\vdots \qquad\qquad \ddots\\
x_{k-1}'(t) &= f_{k-1}(t, x_{k-2}(t), x_{k-1}(t)),\\
0 &= \sum_{\ell=0}^{k-1} \alpha_\ell g^{(\ell)}(t, x_{k-1-\ell}(t), \ldots, x_{k-1}(t)).
\end{aligned} \tag{2.23}$$

The DAE (2.23) has index one. The weights α_ℓ, $\ell = 0, 1, \ldots, k-1$, with $\alpha_{k-1} = 1$ have to be chosen such that the associated differential equation

$$0 = \sum_{\ell=0}^{k-1} \alpha_\ell \eta^{(\ell)}(t)$$

is asymptotically stable with $\|\eta^{(\ell)}(t)\| \longrightarrow 0$ for $\ell \in \{0, \ldots, k-2\}$ as $t \longrightarrow \infty$, compare [73, Sect. VII.2]. A proper choice of the weights is crucial since a balance between quick damping and low degree of stiffness has to be found.

The Baumgarte stabilization was used for real-time simulations in [14, 31], but on the index-two level and not on the index-one level.

2.2.2 Gear–Gupta–Leimkuhler Stabilization

The Gear–Gupta–Leimkuhler (GGL) stabilization [64] does not neglect algebraic constraints but couples them to the index reduced DAE using an additional multiplier. Consider the mechanical multibody system (2.13). The GGL stabilization reads as follows:

$$\begin{aligned} q'(t) &= v(t) - g'_q(t,q(t))^\top \mu(t), \\ M(t,q(t))v'(t) &= f(t,q(t),v(t)) - g'_q(t,q(t))^\top \lambda(t), \\ 0 &= g(t,q(t)), \\ 0 &= g'_t(t,q(t)) + g'_q(t,q(t)) \cdot v(t) \end{aligned} \tag{2.24}$$

The DAE is of Hessenberg type (if multiplied by M^{-1}) and it has index two, if M is symmetric and positive definite and g'_q has full rank. Differentiation of the first algebraic equation yields

$$0 = g'_t(t,q(t)) + g'_q(t,q(t)) \cdot \big(v(t) - g'_q(t,q(t))^\top \mu(t)\big) = -g'_q(t,q(t)) g'_q(t,q(t))^\top \mu(t).$$

Since g'_q is supposed to be of full rank, the matrix $g'_q[t]g'_q[t]^\top$ is non-singular and the equation implies $\mu \equiv 0$.

The idea of the GGL stabilization can be extended to Hessenberg DAEs. To this end consider (2.7) and the index reduced DAE (2.17) with $j \in \{1,\ldots,k-1\}$ fixed. Define

$$G(t,x_1,\ldots,x_{k-1}) := \begin{pmatrix} g^{(0)}(t,x_{k-1}) \\ g^{(1)}(t,x_{k-2},x_{k-1}) \\ \vdots \\ g^{(j-1)}(t,x_{k-j},\ldots,x_{k-1}) \end{pmatrix}$$

and suppose the Jacobian

$$G'_{(x_1,\ldots,x_{k-1})} = \begin{pmatrix} 0\cdots 0 & 0 & \cdots & 0 & (g^{(0)})'_{x_{k-1}} \\ 0\cdots 0 & \vdots & \iddots & (g^{(1)})'_{x_{k-2}} & (g^{(1)})'_{x_{k-1}} \\ 0\cdots 0 & 0 & \iddots & \vdots & \vdots \\ 0\cdots 0 & (g^{(j-1)})'_{x_{k-j}} & \cdots & (g^{(j-1)})'_{x_{k-2}} & (g^{(j-1)})'_{x_{k-1}} \end{pmatrix}$$

has full rank. A stabilized version of (2.17) is given by

$$\begin{aligned} x'(t) &= f(t, x(t), y(t)) - G_x'(t, x(t))^\top \mu(t), \\ 0 &= G(t, x(t)), \\ 0 &= g^{(j)}(t, x_{k-j-1}(t), \dots, x_{k-1}(t)), \end{aligned} \tag{2.25}$$

where μ is an additional algebraic variable, $x = (x_1, \dots, x_{k-1})^\top$, and $f = (f_1, \dots, f_{k-1})^\top$. The stabilized DAE has index $\max\{2, k-j\}$. Note that

$$\begin{aligned} G_t'[t] + G_x'[t]f[t] &= \begin{pmatrix} (g^{(0)})_t'[t] + (g^{(0)})_{x_{k-1}}'[t]f_{k-1}[t] \\ \vdots \\ (g^{(j-1)})_t'[t] + \sum_{\ell=1}^{j} (g^{(j-1)})_{x_{k-\ell}}'[t]f_{k-\ell}[t] \end{pmatrix} \\ &= \begin{pmatrix} g^{(1)}[t] \\ \vdots \\ g^{(j)}[t] \end{pmatrix} = 0. \end{aligned}$$

Moreover,

$$\begin{aligned} 0 &= \frac{d}{dt} G(t, x_1(t), \dots, x_{k-1}(t)) \\ &= G_t'[t] + G_x'[t]\left(f[t] - G_x'[t]^\top \mu(t)\right) \\ &= G_t'[t] + G_x'[t]f[t] - G_x'[t]G_x'[t]^\top \mu(t) \\ &= -G_x'[t]G_x'[t]^\top \mu(t) \end{aligned}$$

and thus $\mu \equiv 0$ since G_x' was supposed to have full rank.

2.2.3 Stabilization by Over-Determination

The GGL stabilization approaches for the mechanical multibody system in (2.24) and the Hessenberg DAE in (2.25) are mathematically equivalent to the *overdetermined DAEs*

$$\begin{aligned} q'(t) &= v(t), \\ M(t, q(t))v'(t) &= f(t, q(t), v(t)) - g_q'(t, q(t))^\top \lambda(t), \\ 0 &= g(t, q(t)), \\ 0 &= g_t'(t, q(t)) + g_q'(t, q(t)) \cdot v(t) \end{aligned}$$

and

$$\begin{aligned} x'(t) &= f(t,x(t),y(t)), \\ 0 &= G(t,x(t)), \\ 0 &= g^{(j)}(t,x_{k-j-1}(t),\dots,x_{k-1}(t)), \end{aligned}$$

respectively, because the additional algebraic variable μ vanishes in either case. Hence, from an analytical point of view there is no difference between the respective systems. A different treatment is necessary from the numerical point of view, though. The GGL stabilized DAEs in (2.24) and (2.25) can be solved by standard discretization schemes, like BDF methods or methods of Runge–Kutta type, provided those are suitable for higher index DAEs. In contrast, the overdetermined DAEs require tailored numerical methods that are capable of dealing with overdetermined linear equations, which arise internally in each integration step. Typically, such overdetermined equations are solved in a least-squares sense, compare [56, 57] for details.

3 Consistent Initialization and Influence of Parameters

One of the crucial issues when dealing with DAEs is that a DAE in general only permits a solution for properly defined initial values, the so-called *consistent initial values*. The initial values not only have to satisfy those algebraic constraints that are explicitly present in the DAE, but hidden constraints have to be satisfied as well.

3.1 Consistent Initial Values

For the Hessenberg DAE (2.7) consistency is defined as follows.

Definition 3.1 (Consistent Initial Value for Hessenberg DAEs) The initial values $x(t_0) = (x_1(t_0),\dots,x_{k-1}(t_0))^\top$ and $y(t_0)$ are *consistent with (2.7)*, if the equations

$$0 = g^{(j)}(t_0,x_{k-1-j}(t_0),\dots,x_{k-1}(t_0)), \qquad j = 1,2,\dots,k-2, \tag{3.1}$$

$$0 = g^{(k-1)}(t_0,y(t_0),x_1(t_0),\dots,x_{k-1}(t_0)) \tag{3.2}$$

hold.

Finding a consistent initial value for a Hessenberg DAE typically consists of two steps. Firstly, a suitable $x(t_0)$ subject to the constraints (3.1) has to be determined. Secondly, given $x(t_0)$ with (3.1), Eq. (3.2) can be solved for $y_0 = y(t_0)$ by Newton's method, if the matrix $R_0 = \partial g^{(k-1)}/\partial y$ is non-singular in a solution (assuming that a solution exists). For mechanical multibody systems even a linear equation arises in the second step.

Example 3.1 Consider the mechanical multibody system (2.13). A consistent initial value (q_0, v_0, λ_0) at t_0 must satisfy

$$\begin{aligned}
0 &= g(t_0, q_0),\\
0 &= g_t'(t_0, q_0) + g_q'(t_0, q_0) \cdot v_0,\\
0 &= g_{tt}''(t_0, q_0) + g_{tq}''(t_0, q_0) \cdot v_0 + g_q'(t_0, q_0) \cdot v_0' + g_{qq}''(t_0, q_0)(v_0, v_0)
\end{aligned}$$

with

$$M(t_0, q_0)v_0' = f(t_0, q_0, v_0) - g_q'(t_0, q_0)^\top \lambda_0.$$

The latter two equations yield a linear equation for v_0' and λ_0:

$$\begin{pmatrix} M(t_0, q_0) & g_q'(t_0, q_0)^\top \\ g_q'(t_0, q_0) & 0 \end{pmatrix} \begin{pmatrix} v_0' \\ \lambda_0 \end{pmatrix} = \begin{pmatrix} f(t_0, q_0, v_0) \\ -g_{tt}''(t_0, q_0) - g_{tq}''(t_0, q_0) \cdot v_0 - g_{qq}''(t_0, q_0)(v_0, v_0) \end{pmatrix}.$$

The matrix on the left-hand side is non-singular, if M is symmetric and positive definite and $g_q'(t_0, q_0)$ is of full rank.

Definition 3.2 (Consistent Initial Value for General DAEs, Compare [29, Sect. 5.3.4]) For a general DAE (1.1) with differentiation index d the initial value $z_0 = z(t_0)$ is said to be consistent at t_0, if the derivative array

$$F^{(j)}(t_0, z_0, z_0', \ldots, z_0^{(j+1)}) = 0, \qquad j = 0, 1, \ldots, d, \tag{3.3}$$

in (2.4) has a solution $(z_0, z_0', \ldots, z_0^{(d+1)})$.

Note that the system of nonlinear equations (3.3) in general has many solutions and additional conditions are required to obtain a particular consistent initial value, which might be relevant for a particular application. This can be achieved for instance by imposing additional constraints

$$G(t_0, z_0, z_0') = 0, \tag{3.4}$$

which are known to hold for a specific application, compare [29, Sect. 5.3.4]. Of course, such additional constraints must not contradict the equations in (3.3).

If the user is not able to formulate relations in (3.4) such that the combined system of equations (3.3) and (3.4) returns a unique solution, then a least-squares approach could be used to find a consistent initial value closest to a ‘desired’ initial value, compare [33]:

$$\begin{array}{lll} \text{Minimize} & \frac{1}{2}\|G(t_0, z_0, z_0')\|^2 + \frac{1}{2}\sum_{\ell=2}^{d+1} \|z_0^{(\ell)}\|^2 & \\ \text{w.r.t.} & (z_0, z_0', \ldots, z_0^{(d+1)})^\top & \\ \text{s.t.} & F^{(j)}(t_0, z_0, z_0', \ldots, z_0^{(j+1)}) = 0, & j = 0, 1, \ldots, d. \end{array}$$

In practical computations the major challenge for higher index DAEs is to obtain analytical expressions or numerical approximations of the derivatives in $F^{(j)}$, $j = 1, \ldots, d$. For this purpose computer algebra packages like MAPLE, MATHEMATICA, or the symbolic toolbox of MATLAB can be used. Algorithmic differentiation tools are suitable as well, compare [72] for an overview. A potential issue is redundancy in the constraints (3.3) and the identification of the relevant equations in the derivative array. Approaches for the consistent initialization of general DAEs can be found in [1, 30, 35, 51, 71, 78, 93, 108]. A different approach is used in [127] in the context of shooting methods for parameter identification problems or optimal control problems. Herein, the algebraic constraints of the DAE are relaxed such that they are satisfied for any initial value. Then, the relaxation terms are driven to zero in the superordinate optimization problem in order to ensure consistency with the original DAE.

3.2 Dependence on Parameters

Initial values may depend on parameters that are present in the DAE. To this end the recomputation of consistent initial values for perturbed parameters becomes necessary or a parametric sensitivity analysis has to be performed, compare [66, 69]. Such issues frequently arise in the context of optimal control problems or parameter identification problems subject to DAEs, compare [68].

Example 3.2 Consider the equations of motion of a pendulum of mass m and length ℓ in the plane:

$$\begin{aligned} q_1'(t) &= v_1(t), \\ q_2'(t) &= v_2(t), \\ mv_1'(t) &= \qquad - 2q_1(t)\lambda(t), \\ mv_2'(t) &= -mg - 2q_2(t)\lambda(t), \\ 0 &= q_1(t)^2 + q_2(t)^2 - \ell^2. \end{aligned}$$

Herein, (q_1, q_2) denotes the pendulum's position, (v_1, v_2) its velocity, and λ the stress in the bar. A consistent initial value $(q_{1,0}, q_{2,0}, v_{1,0}, v_{2,0}, \lambda_0)$ has to satisfy the equations

$$0 = q_{1,0}^2 + q_{2,0}^2 - \ell^2, \tag{3.5}$$

$$0 = q_{1,0}v_{1,0} + q_{2,0}v_{2,0}, \tag{3.6}$$

$$0 = -\frac{\ell^2}{m}\lambda_0 + (v_{1,0}^2 + v_{2,0}^2) - q_{2,0}g. \tag{3.7}$$

Apparently, the algebraic component λ_0 depends on the parameter $p = (m, \ell, g)^\top$ according to

$$\lambda_0 = \lambda_0(p) = \frac{m}{\ell^2}\left(v_{1,0}^2 + v_{2,0}^2 - q_{2,0}g\right).$$

But in addition, the positions $q_{1,0}$ and $q_{2,0}$ depend on ℓ through the relation (3.5). So, if ℓ changes, then $q_{1,0}$ and/or $q_{2,0}$ have to change as well subject to (3.5) and (3.6). However, those equations in general do not uniquely define $q_{1,0}, q_{2,0}, v_{1,0}, v_{2,0}$ and the question arises, which set of values one should choose?

Firstly, we focus on the recomputation of an initial value for perturbed parameters. As the previous example shows, there is not a unique way to determine such a consistent initial value. A common approach is to use a projection technique, compare, e.g., [69] for a class of index-two DAEs, [68, Sect. 4.5.1] for Hessenberg DAEs, or [33] for general DAEs.

Consider the general parametric DAE

$$F(t, z(t), z'(t), p) = 0 \tag{3.8}$$

and the corresponding derivative array

$$F^{(j)}(t, z, z', \ldots, z^{(j+1)}, p) = 0, \qquad j = 0, 1, \ldots, d.$$

Remark 3.1 Please note that the differentiation index d of the general parametric DAE (3.8) may depend on p. For simplicity, we assume throughout that this is not the case (at least locally around a fixed nominal parameter).

Let $\tilde{p}$ be a given parameter. Suppose $\tilde{z}_0 = z_0(\tilde{p})$ with $\tilde{z}_0' = z_0'(\tilde{p}), \ldots, \tilde{z}_0^{(d+1)} = z_0^{(d+1)}(\tilde{p})$ is consistent. In order to find a consistent initial value for p, which is supposed to be close to $\tilde{p}$, solve the following parametric constrained least-squares problem:

LSQ(p): Minimize

$$\frac{1}{2}\|(\xi_0, \xi_0', \ldots, \xi_0^{(d+1)})^\top - (\tilde{z}_0, \tilde{z}_0', \ldots, \tilde{z}_0^{(d+1)})^\top\|^2$$

with respect to $(\xi_0, \xi_0', \dots, \xi_0^{(d+1)})^\top$ *subject to the constraints*

$$F^{(j)}(t_0, \xi_0, \xi_0', \dots, \xi_0^{(j+1)}, p) = 0, \qquad j = 0, 1, \dots, d.$$

Remark 3.2 In case of a parametric Hessenberg DAE it would be sufficient to consider the hidden constraints up to order $k-2$ as the constraints in LSQ(p) and to compute a consistent algebraic component afterwards. Moreover, the quantities t_0 and $\tilde{z}_0^{(j)}, j = 0, \dots, d+1$, could be considered as parameters of LSQ(p) as well, but here we are only interested in p.

The least-squares problem LSQ(p) is a parametric nonlinear optimization problem and allows for a sensitivity analysis in the spirit of [54] in order to investigate the sensitivity of a solution of LSQ(p) for p close to some nominal value $\hat{p}$. Let

$$L(\xi, \mu, p) = \frac{1}{2}\|\xi - \tilde{z}\|^2 + \mu^\top G(\xi, p) \tag{3.9}$$

with $\xi = (\xi_0, \xi_0', \dots, \xi_0^{(d+1)})^\top$, $\tilde{z} = (\tilde{z}_0, \tilde{z}_0', \dots, \tilde{z}_0^{(d+1)})^\top$, and

$$G(\xi, p) = \left(F^{(j)}(t_0, \xi_0, \xi_0', \dots, \xi_0^{(j+1)}, p) \right)_{j=0,1,\dots,d} \tag{3.10}$$

denote the Lagrange function of LSQ(p).

Theorem 3.1 (Sensitivity Theorem, Compare [54]) *Let G in (3.10) be twice continuously differentiable and* $\hat{p}$ *a nominal parameter. Let* $\hat{\xi}$ *be a local minimum of LSQ(*$\hat{p}$*) with Lagrange multiplier* $\hat{\mu}$ *such that the following assumptions hold:*

(a) Linear independence constraint qualification: $G_\xi'(\hat{\xi}, \hat{p})$ *has full rank.*
(b) KKT conditions: $0 = \nabla_\xi L(\hat{\xi}, \hat{\mu}, \hat{p})$ *with L from (3.9)*
(c) Second-order sufficient condition:

$$L_{\xi\xi}''(\hat{\xi}, \hat{\mu}, \hat{p})(h, h) > 0 \qquad \forall h \neq 0 \; : \; G_\xi'(\hat{\xi}, \hat{p})h = 0.$$

Then there exist neighborhoods $B_\epsilon(\hat{p})$ *and* $B_\delta(\hat{\xi}, \hat{\mu})$*, such that LSQ(p) has a unique local minimum*

$$(\xi(p), \mu(p)) \in B_\delta(\hat{\xi}, \hat{\mu})$$

for each $p \in B_\epsilon(\hat{p})$*. In addition,* $(\xi(p), \mu(p))$ *is continuously differentiable with respect to p with*

$$\begin{pmatrix} L_{\xi\xi}''(\hat{\xi}, \hat{\mu}, \hat{p}) & G_\xi'(\hat{\xi}, \hat{p})^\top \\ G_\xi'(\hat{\xi}, \hat{p}) & 0 \end{pmatrix} \begin{pmatrix} \xi'(\hat{p}) \\ \mu'(\hat{p}) \end{pmatrix} = - \begin{pmatrix} L_{\xi p}''(\hat{\xi}, \hat{\mu}, \hat{p}) \\ G_p'(\hat{\xi}, \hat{p}) \end{pmatrix}. \tag{3.11}$$

The second equation in (3.11) reads

$$G'_\xi(\hat{\xi},\hat{p})\xi'(\hat{p}) + G'_p(\hat{\xi},\hat{p}) = 0$$

and in more detail using (3.10),

$$\sum_{\ell=0}^{j+1} \frac{\partial F^{(j)}}{\partial z^{(\ell)}}(t_0,\hat{\xi}_0,\dots,\hat{\xi}_0^{(j+1)},\hat{p}) \cdot (\xi_0^{(\ell)})'(\hat{p}) + \frac{\partial F^{(j)}}{\partial p}(t_0,\hat{\xi}_0,\dots,\hat{\xi}_0^{(j+1)},\hat{p}) = 0$$

for $j = 0, 1, \dots, d$. Let us define $S_0^{(\ell)} := (\xi_0^{(\ell)})'(\hat{p})$ for $\ell = 0, \dots, d+1$. Then, we obtain

$$\sum_{\ell=0}^{j+1} \frac{\partial F^{(j)}}{\partial z^{(\ell)}}(t_0,\hat{\xi}_0,\dots,\hat{\xi}_0^{(j+1)},\hat{p}) \cdot S_0^{(\ell)} + \frac{\partial F^{(j)}}{\partial p}(t_0,\hat{\xi}_0,\dots,\hat{\xi}_0^{(j+1)},\hat{p}) = 0 \qquad (3.12)$$

and in particular for $j = 0$,

$$\frac{\partial F}{\partial z}(t_0,\hat{\xi}_0,\hat{\xi}'_0,\hat{p}) \cdot S_0 + \frac{\partial F}{\partial z'}(t_0,\hat{\xi}_0,\hat{\xi}'_0,\hat{p}) \cdot S'_0 + \frac{\partial F}{\partial p}(t_0,\hat{\xi}_0,\hat{\xi}'_0,\hat{p}) = 0,$$

which is the linearization of (3.8) around $(t_0, \xi_0, \xi'_0, \hat{p})$ with respect to p. Taking into account the definition of the further components $F^{(j)}$, $j = 1, \dots, d+1$, of the derivative array, compare (2.3), we recognize that (3.12) provides a linearization of (2.3) with respect to p. Hence, the settings

$$S(t_0) = S_0, \quad S'(t_0) = S'_0, \dots, S^{(d+1)}(t_0) = S_0^{(d+1)}$$

provide consistent initial values for the sensitivity DAE

$$F'_z(t,z(t),z'(t),p)S(t) + F'_{z'}(t,z(t),z'(t),p)S'(t) + F'_p(t,z(t),z'(t),p) = 0,$$

where $S(t) := \partial z(t;p)/\partial p$ denotes the sensitivity of the solution of (3.8) with respect to the parameter p, compare Sect. 7. Herein, it is assumed that F is sufficiently smooth with respect to all arguments.

In summary, the benefits of the projection approach using LSQ(p) are twofold: Firstly, it allows to compute consistent initial values for the DAE itself. Secondly, the sensitivity analysis provides consistent initial values for the sensitivity DAE. Finally, the sensitivity analysis can be used to predict consistent initial values under perturbations through the Taylor expansion

$$\xi(p) = \xi(\hat{p}) + \xi'(\hat{p})(p-\hat{p}) + o(\|p-\hat{p}\|)$$

for $p \in B_\varepsilon(\hat{p})$.

4 Integration Methods

A vast number of numerical discretizations schemes exist for DAEs, most of them are originally designed for ODEs, such as BDF methods or Runge–Kutta methods. The methods for DAEs are typically (at least in part) implicit methods owing to the presence of algebraic equations. It is beyond the scope of the paper to provide a comprehensive overview on all the existing numerical discretization schemes for DAEs, since excellent textbooks with convergence results and many details are available, for instance [29, 73, 75–77, 90, 92, 134]. Our intention is to discuss the most commonly used methods, their construction principles, and some of their features. Efficient implementations use a bunch of additional ideas to improve the efficiency.

All methods work on a grid

$$\mathbb{G}_h = \{t_0 < t_1 < \ldots < t_{N-1} < t_N = t_f\}$$

with $N \in \mathbb{N}$ and step-sizes $h_k = t_{k+1} - t_k$, $k = 0, \ldots, N-1$. The maximum step-size is denoted by $h = \max\limits_{k=0,\ldots,N-1} h_k$. The methods generate a grid function $z_h : \mathbb{G}_h \longrightarrow \mathbb{R}^n$ with $z_h(t) \approx z(t)$ for $t \in \mathbb{G}_h$, where $z(t)$ denotes the solution of (1.1) with a consistent initial value z_0. The discretization schemes can be grouped into *one-step methods* with

$$z_h(t_{i+1}) = z_h(t_i) + h_i \Phi(t_i, z_h(t_i), h_i), \qquad i = 0, \ldots, N-1, \tag{4.1}$$

for a given consistent initial value $z_h(t_0) = z_0$ and *s-stage multi-step methods* with

$$z_h(t_{i+s}) = \Psi(t_i, \ldots, t_{i+s}, z_h(t_i), \ldots, z_h(t_{i+s-1}), h_i, \ldots, h_{i+s-1}), \quad i = 0, \ldots, N-s, \tag{4.2}$$

for given consistent initial values $z_h(t_0) = z_0, \ldots, z_h(t_{s-1}) = z_{s-1}$. Note that multi-step methods with $s > 1$ require an initialization procedure to compute $z_1, \ldots, z_{s-1}$. This can be realized by performing $s-1$ steps of a suitable one-step method or by using multi-step methods with $1, 2, \ldots, s-1$ stages successively for the first $s-1$ steps.

The aim is to construct convergent methods such that the *global error* $e_h : \mathbb{G}_h \longrightarrow \mathbb{R}^n$ defined by

$$e_h := z_h - \Delta_h(z), \qquad e_h(t) = z_h(t) - \Delta_h(z)(t),\ t \in \mathbb{G}_h,$$

satisfies

$$\lim_{h\to 0} \|e_h\|_\infty = 0$$

or even exhibits the order of convergence $p \in \mathbb{N}$, i.e.

$$\|e_h\|_\infty = \mathcal{O}(h^p) \text{ as } h \to 0.$$

Herein, $\Delta_h : \{z : [t_0, t_f] \longrightarrow \mathbb{R}^n\} \longrightarrow \{z_h : \mathbb{G}_h \longrightarrow \mathbb{R}^n\}$ denotes the restriction operator onto the set of grid functions on $\mathbb{G}_h$ defined by $\Delta_h(z)(t) = z(t)$ for $t \in \mathbb{G}_h$.

A convergence proof for a specific discretization scheme typically resembles the reasoning

$$\text{consistency} \quad + \quad \text{stability} \quad \Longrightarrow \quad \text{convergence},$$

compare [131]. Herein, consistency is not to be confused with consistent initial values. Instead, consistency of a discretization method measures how well the exact solution satisfies the discretization scheme. Detailed definitions of consistency and stability and convergence proofs for various classes of DAEs (index-one, Hessenberg DAEs up to order 3, constant/variable step-sizes) can be found in the above-mentioned textbooks [29, 73, 75–77, 90, 92, 134]. As a rule, one cannot in general expect the same order of convergence for differential and algebraic variables for higher index DAEs.

4.1 BDF Methods

The Backward Differentiation Formulas (BDF) are implicit multi-step methods and were introduced in [40, 61]. A BDF method with $s \in \mathbb{N}$ stages is based on the construction of interpolating polynomials, compare Fig. 1. Suppose the method has produced approximations $z_h(t_{i+k})$, $k = 0, \dots, s-1$, of z at the grid points t_{i+k}, $k = 0, \dots, s-1$. The aim is to determine an approximation $z_h(t_{i+s})$ of $z(t_{i+s})$, where $i \in \{0, \dots, N-s\}$.

To this end, let $P(t)$ be the interpolating polynomial of degree at most s with

$$P(t_{i+k}) = z_h(t_{i+k}), \qquad k = 0, \dots, s.$$

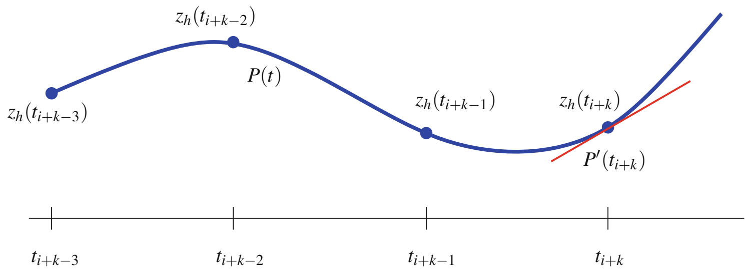

Fig. 1 Idea of BDF method: polynomial interpolation of approximations

The polynomial P can be expressed as

$$P(t) = \sum_{k=0}^{s} z_h(t_{i+k}) L_k(t), \qquad L_k(t) = \prod_{\ell=0,\ell\neq k}^{s} \frac{t - t_{i+\ell}}{t_{i+k} - t_{i+\ell}},$$

where the L_k's denote the Lagrange polynomials. Note that P interpolates the unknown vector $z_h(t_{i+s})$, which is determined implicitly by the postulation that P satisfies the DAE (1.1) at t_{i+s}, i.e.

$$F(t_{i+s}, z_h(t_{i+s}), P'(t_{i+s})) = 0. \tag{4.3}$$

The above representation of P yields

$$P'(t_{i+s}) = \sum_{k=0}^{s} z_h(t_{i+k}) L'_k(t_{i+s}) =: \frac{1}{h_{i+s-1}} \sum_{k=0}^{s} \alpha_k z_h(t_{i+k}),$$

with $\alpha_k = h_{i+s-1} L'_k(t_{i+s})$, $k = 0, \ldots, s$.

Example 4.1 The BDF methods with $s \leq 6$ and a constant step-size h read as follows, see [134, S. 335]:

$$\begin{aligned}
s = 1 &: hP'(t_{i+1}) = z_{i+1} - z_i \quad \text{(implicit Euler method)}\\
s = 2 &: hP'(t_{i+2}) = \frac{1}{2}(3z_{i+2} - 4z_{i+1} + z_i)\\
s = 3 &: hP'(t_{i+3}) = \frac{1}{6}(11z_{i+3} - 18z_{i+2} + 9z_{i+1} - 2z_i)\\
s = 4 &: hP'(t_{i+4}) = \frac{1}{12}(25z_{i+4} - 48z_{i+3} + 36z_{i+2} - 16z_{i+1} + 3z_i)\\
s = 5 &: hP'(t_{i+5}) = \frac{1}{60}(137z_{i+5} - 300z_{i+4} + 300z_{i+3} - 200z_{i+2} + 75z_{i+1} - 12z_i)\\
s = 6 &: hP'(t_{i+6}) = \frac{1}{60}(147z_{i+6} - 360z_{i+5} + 450z_{i+4} - 400z_{i+3} + 225z_{i+2}\\
&\qquad -72z_{i+1} + 10z_i)\,.
\end{aligned}$$

Abbreviations: $z_{i+k} = z_h(t_{i+k})$, $k = 0, \ldots, 6$.

Introducing the expression for $P'(t_{i+s})$ into (4.3) yields the nonlinear equation

$$F\left(t_{i+s}, z_h(t_{i+s}), \frac{1}{h_{i+s-1}} \sum_{k=0}^{s} \alpha_k z_h(t_{i+k})\right) = 0 \tag{4.4}$$

for $z_h(t_{i+s})$. Suppose (4.4) possesses a solution $z_h(t_{i+s})$ and the matrix pencil

$$F'_z + \frac{\alpha_s}{h_{i+s-1}} F'_{z'} \tag{4.5}$$

is regular at this solution, i.e., there exists a step-size h_{i+s-1} such that the matrix (4.5) is non-singular. Then the implicit function theorem allows to solve Eq. (4.4) locally for $z_h(t_{i+s})$ and to express it in the form (4.2). In practice Newton's method or the simplified Newton method is used to solve Eq. (4.4) numerically, which requires the non-singularity of the matrix in (4.5) at the Newton iterates.

BDF methods are appealing amongst implicit methods since the effort per integration step amounts to solving just one nonlinear equation of dimension n, whereas a fully implicit s-stage Runge–Kutta method requires to solve a nonlinear equation of dimension $n \cdot s$. For numerical purposes only the BDF methods with $s \leq 6$ are relevant, since the methods for $s > 6$ are unstable. The maximal attainable order of convergence of an s-stage BDF method is s.

Convergence results assuming fixed step-sizes for BDF methods for certain subclasses of the general DAE (1.1), amongst them are index-one problems and Hessenberg DAEs, can be found in [27, 63, 99, 111]. Variable step-sizes may result in non-convergent components of the algebraic variables for index-three Hessenberg DAEs, compare [64]. This is another motivation to use an index reducing stabilization technique as in Sect. 2.2.

The famous code DASSL, [29, 109], is based on BDF methods, but adds several features like an automatic step-size selection strategy, a variable order selection strategy, a root finding strategy, and a parametric sensitivity module to the basic BDF method. Moreover, the re-use of Jacobians for one or more integration steps and numerically efficient divided difference schemes for the calculation of the interpolating polynomial P increase the efficiency of the code. The code ODASSL by Führer [56] and Führer and Leimkuhler [57] extends DASSL to overdetermined DAEs, which occur, e.g., for the GGL stabilization in Sect. 2.2. In these codes, the error tolerances for the algebraic variables of higher index DAEs have to be scaled by powers of $1/h$ compared to those of the differential states since otherwise the automatic step-size selection algorithm breaks down frequently, compare [110]. An enhanced version of DASSL is available in the package SUNDIALS, [80], which provides several methods (Runge–Kutta, Adams, BDF) for ODEs and DAEs in one software package.

4.2 Runge–Kutta Methods

A Runge–Kutta method with $s \in \mathbb{N}$ stages for (1.1) is a one-step method of type

$$z_h(t_{i+1}) = z_h(t_i) + h_i \Phi(t_i, z_h(t_i), h_i) \tag{4.6}$$

with the increment function

$$\Phi(t, z, h) := \sum_{j=1}^{s} b_j k_j(t, z, h) \tag{4.7}$$

and the stage derivatives $k_j(t, z, h)$, $j = 1, \ldots, s$. The stage derivatives k_j are implicitly defined by the system of $n \cdot s$ nonlinear equations

$$F\left(t_i + c_1 h_i, z_{i+1}^{(1)}, k_1\right) = 0, \tag{4.8}$$

$$\vdots$$

$$F\left(t_i + c_s h_i, z_{i+1}^{(s)}, k_s\right) = 0, \tag{4.9}$$

where

$$z_{i+1}^{(\ell)} := z_h(t_i) + h_i \sum_{j=1}^{s} a_{\ell j}\, k_j, \qquad \ell = 1, \ldots, s, \tag{4.10}$$

are approximations of z at the intermediate time points $t_i + c_\ell h$, $\ell = 1, \ldots, s$. The coefficients in the Runge–Kutta method are collected in the Butcher array

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Commonly used Runge–Kutta methods for DAEs are the RADAU IIA methods and the Lobatto IIIA and IIIC methods. These methods are stiffly accurate, i.e., they satisfy $c_s = 1$ and $a_{sj} = b_j$ for $j = 1, \ldots, s$. This is a very desirable property for DAEs since it implies that (4.9) and $z_{i+1}^{(s)} = z_h(t_{i+1})$ hold at $t_{i+1} = t_i + c_s h_i$. Runge–Kutta methods, which are not stiffly accurate, can be used as well. However, for those it has to be enforced that the approximation $z_h(t_{i+1})$ satisfies the algebraic constraints of the DAE at t_{i+1}. This can be achieved by projecting the output of the Runge–Kutta method onto the algebraic constraints, compare [18].

Example 4.2 (RADAU IIA) The RADAU IIA methods with $s = 1, 2, 3$ are defined by the following Butcher arrays, compare [134, Beispiel 6.1.5]:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 1/3 & 5/12 & -1/12 \\ 1 & 3/4 & 1/4 \\ \hline & 3/4 & 1/4 \end{array} \qquad \begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}$$

The maximal attainable order of convergence is $2s - 1$.

Example 4.3 (Lobatto IIIA and Lobatto IIIC) The Lobatto IIIA methods with $s = 2, 3$ are defined by the following Butcher arrays, compare [134, Beispiel 6.1.6]:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & \\ 1/2 & 5/24 & 1/3 & -1/24 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

The Lobatto IIIC methods with $s = 2, 3$ are defined by the following Butcher arrays, compare [134, Beispiel 6.1.8]:

$$\begin{array}{c|cc} 0 & 1/2 & -1/2 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array} \qquad \begin{array}{c|ccc} 0 & 1/6 & -1/3 & 1/6 \\ 1/2 & 1/6 & 5/12 & -1/12 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

The maximal attainable order of convergence is $2s - 2$. A combined method of Lobatto IIIA and IIIC methods for mechanical multibody systems can be found in [124].

The main effort per integration step is to solve the system of nonlinear equations (4.8)–(4.9) for the unknown vector of stage derivatives $k = (k_1, \dots, k_s)^\top$ by Newton's method or by a simplified version of it, where the Jacobian matrix is kept constant for a couple of iterations or integration steps. Another way to reduce the computational effort is to consider ROW methods or half-explicit Runge–Kutta methods as in Sect. 4.3.

Convergence results and order conditions for Runge–Kutta methods applied to DAEs can be found in, e.g., [28, 75, 84, 85].

4.3 Rosenbrock-Wanner (ROW) Methods

In this section, we introduce and discuss the so-called *Rosenbrock-Wanner(ROW) methods* for DAEs, cf. [121], where H.H. Rosenbrock introduced this method class. ROW methods are one-step methods, which are based on implicit Runge–Kutta methods. In literature, these methods are also called *Rosenbrock methods*, *linearly-implicit or semi-implicit Runge–Kutta methods*, cf. [73].

The motivation to introduce an additional class of integration methods is to avoid solving a fully nonlinear system of dimension $n \cdot s$ and to solve instead of that only linear systems. Thus, the key idea for the derivation of Rosenbrock-Wanner methods is to perform one Newton-step to solve Eqs. (4.8)–(4.9) for a Runge–Kutta method with $a_{ij} = 0$ for $i < j$ (diagonally implicit RK method, cf. [73]). We rewrite these equations for such a method and an autonomous implicit DAE,

$$F(z, z') = 0, \tag{4.11}$$

as follows:

$$F\left(z_h(t_i) + h_i\left(\sum_{j=1}^{\ell-1} a_{\ell j}\, k_j + a_{\ell\ell}\, k_\ell\right), k_\ell\right) = 0, \quad \ell = 1, \ldots, s. \tag{4.12}$$

Due to the fact that we consider a diagonally implicit RK method, the above equations are decoupled and can be solved successively. Then, performing one Newton-step with starting value $k_\ell^{(0)}$ leads to

$$\left(F'_z\left(z_{i+1}^{(\ell-1)}, k_\ell^{(0)}\right) h_i \cdot a_{\ell\ell} + F'_{z'}\left(z_{i+1}^{(\ell-1)}, k_\ell^{(0)}\right)\right)\left(k_\ell - k_\ell^{(0)}\right) = -F\left(z_{i+1}^{(\ell-1)}, k_\ell^{(0)}\right), \tag{4.13}$$

for $\ell = 1, \ldots, s$.

We come to the general class of Rosenbrock-Wanner methods by proceeding with the following steps. First, we take as starting value $k_\ell^{(0)} = 0$ for $\ell = 1, \ldots, s$. Then, the Jacobians are evaluated at the fixed point $z_h(t_i)$ instead of $z_{i+1}^{(\ell-1)}$, which saves computational costs substantially. Moreover, linear combinations of the previous stages $k_j, j = 1, \ldots, \ell$ are introduced. And last but not least, the method is extended to general non-autonomous implicit DAEs as Eq. (1.1). We obtain the following class of Rosenbrock methods

$$F\left(t_i + c_\ell h_i, z_{i+1}^{(\ell-1)}, 0\right) + h_i J_z \sum_{j=1}^{\ell} \gamma_{\ell j} k_j + J_{z'} k_\ell + \gamma_\ell J_t = 0, \quad \ell = 1, \ldots, s, \tag{4.14}$$

with

$$J_z := F_z(t_i, z_h(t_i), 0), \tag{4.15}$$

$$J_{z'} := F_{z'}(t_i, z_h(t_i), 0), \tag{4.16}$$

$$J_t := F_t(t_i, z_h(t_i), 0). \tag{4.17}$$

The solution at the next time point t_{i+1} is computed exactly as in the case of Runge–Kutta methods:

$$z_h(t_{i+1}) = z_h(t_i) + h_i \Phi(t_i, z_h(t_i), h_i), \qquad \Phi(t, z, h) := \sum_{j=1}^{s} b_j k_j(t, z, h), \tag{4.18}$$

with the stage derivatives $k_j(t, z, h)$ defined by the *linear* system (4.14). An example for a ROW method is the linearly implicit Euler method ($s = 1$), for which the stage derivative is defined as follows

$$F\,(t_i, z_i, 0) + h_i J_z k_1 + J_{z'} k_1 = 0. \tag{4.19}$$

For semi-explicit DAEs of the form (2.5)–(2.6), a ROW method as defined above reads as

$$z_h^x(t_{i+1}) = z_h^x(t_i) + \sum_{j=1}^{s} b_j k_j^x(t_i, z_h^x(t_i), z_h^y(t_i), h_i) \tag{4.20}$$

$$z_h^y(t_{i+1}) = z_h^y(t_i) + \sum_{j=1}^{s} b_j k_j^y(t_i, z_h^x(t_i), z_h^y(t_i), h_i), \tag{4.21}$$

with

$$\begin{aligned} &\begin{pmatrix} f\left(t_i + c_\ell h_i, z_{i+1}^{x,(\ell-1)}, z_{i+1}^{y,(\ell-1)}\right) \\ g\left(t_i + c_\ell h_i, z_{i+1}^{x,(\ell-1)}, z_{i+1}^{y,(\ell-1)}\right) \end{pmatrix} + h_i \cdot \begin{pmatrix} f_{z^x} & f_{z^y} \\ g_{z^x} & g_{z^y} \end{pmatrix} \sum_{j=1}^{\ell} \gamma_{\ell j} \begin{pmatrix} k_j^x \\ k_j^y \end{pmatrix} \\ &\qquad + \begin{pmatrix} -k_\ell^x \\ 0 \end{pmatrix} + \gamma_\ell \begin{pmatrix} f_t \\ g_t \end{pmatrix} = 0, \end{aligned} \tag{4.22}$$

for $\ell = 1, \ldots, s$. Herein, we have set $z = ((z^x)^\top, (z^y)^\top)^\top = (x^\top, y^\top)^\top$ and

$$F(t, z, z') = \begin{pmatrix} f(t, x, y) - x' \\ g(t, x, y) \end{pmatrix}. \tag{4.23}$$

Up to now, we have considered ROW methods with exact Jacobian matrices $J_z = F_z$, $J_{z'} = F_{z'}$. There is an additional class of integration methods, which uses for J_z arbitrary matrices (‘inexact Jacobians’)—such methods are called *W-methods*, see [73, 146, 147].

We further remark that related integration methods can be derived, if other starting values are used for the stage derivatives, instead of $k_\ell^{(0)} = 0$ as it is done to derive Eq. (4.14), cf. [67, 68]—the methods derived there as well as ROW and W-methods can be seen to belong the common class of *linearized implicit Runge–Kutta methods*.

An introduction and more detailed discussion of ROW methods can be found in [73]; convergence results for general one-step methods (including ROW methods) applied to DAEs are available in [41]. Moreover, ROW methods for index-one DAEs in semi-explicit form are studied in [53, 117, 120, 135]; index-one problems and singularly perturbed problems are discussed in [23, 24, 74]. Analysis results and specific methods for the equations of motion of mechanical multibody systems, i.e., index-three DAEs in semi-explicit form are derived in [145, 146]; compare also the results in [14, 106, 123].

4.4 *Half-Explicit Methods*

In this section, we briefly discuss the so-called *half-explicit* Runge–Kutta methods, here for autonomous index-two DAEs in semi-explicit form. That is, we consider DAE systems of the form

$$x'(t) = f(x(t), y(t)), \tag{4.24}$$

$$0 = g(x(t)), \tag{4.25}$$

with initial values (x_0, y_0) that are assumed to be consistent. To derive the class of half-explicit Runge–Kutta methods, it is more convenient to use stages rather than the stage-derivatives k_ℓ as before. In particular, for the semi-explicit DAE (4.24), (4.25), we define stages for the differential and the algebraic variables as

$$X_{i\ell} := x_h(t_i) + h_i \sum_{j=1}^{s} a_{\ell j} k_j^x, \quad Y_{i\ell} := y_h(t_i) + h_i \sum_{j=1}^{s} a_{\ell j} k_j^y, \quad \ell = 1, \dots, s. \tag{4.26}$$

Then, it holds

$$\begin{aligned} X_{i\ell} &= x_h(t_i) + h_i \sum_{j=1}^{s} a_{\ell j} k_j^x \\ &= x_h(t_i) + h_i \sum_{j=1}^{s} a_{\ell j} f\left(x_h(t_i) + \sum_{m=1}^{s} a_{jm} k_m^x,\ y_h(t_i) + \sum_{m=1}^{s} a_{jm} k_m^y\right) \\ &= x_h(t_i) + h \sum_{j=1}^{s} a_{\ell j} f(X_{ij}, Y_{ij}). \end{aligned} \tag{4.27}$$

Using this notation and the coefficients of an explicit Runge–Kutta scheme, half-explicit Runge–Kutta methods as firstly introduced in [75] are defined as follows

$$X_{i\ell} = x_h(t_i) + h_i \sum_{j=1}^{\ell-1} a_{\ell j} f(X_{nj}, Y_{nj}), \qquad \ell = 1, \dots, s, \tag{4.28}$$

$$0 = g(X_{i\ell}), \tag{4.29}$$

$$x_h(t_{i+1}) = x_h(t_i) + h_i \sum_{\ell=1}^{s} b_\ell f(X_{i\ell}, Y_{i\ell}), \tag{4.30}$$

$$0 = g(x_h(t_{i+1})). \tag{4.31}$$

The algorithmic procedure is as follows: We start with $X_{i1} = x_h(t_i)$ assumed to be consistent. Then, taking Eq. (4.28) for X_{i2} and inserting into Eq. (4.29) lead to

$$0 = g(X_{i2}) = g\left(x_h(t_i) + a_{21} h_i f(X_{i1}, Y_{i1})\right), \tag{4.32}$$

this is a nonlinear equation that can be solved for Y_{i1}. Next, we calculate X_{i2} from Eq. (4.28) and, accordingly, Y_{i2}, etc. For methods with $c_s = 1$, one obtains an approximation for the algebraic variable at the next time-point by $y_h(t_{i+1}) = Y_{is}$. The key idea behind this kind of integration schemes is to apply an explicit Runge–Kutta scheme for the differential variable and to solve for the algebraic variable implicitly.

Convergence studies for this method class applied to index-two DAEs can be found in [26, 75]. In [7, 12, 105] the authors introduce a slight modification of the above stated scheme, which improves the method class concerning order conditions and computational efficiency. To be more precise, *partitioned half-explicit Runge–Kutta methods* for index-two DAEs in semi-explicit form are defined in the following way:

$$\begin{aligned}
X_{i1} &= x_h(t_i), \quad Y_{i1} = y_h(t_i),\\
X_{i\ell} &= x_h(t_i) + h_i \sum_{j=1}^{\ell-1} a_{\ell j} f(X_{ij}, Y_{ij}),\\
\bar{X}_{i\ell} &= x_h(t_i) + h_i \sum_{j=1}^{\ell} \bar{a}_{\ell j} f(X_{ij}, Y_{ij}),\\
0 &= g(\bar{X}_{i\ell}),\\
\ell &= 2, \dots s+1,\\
x_h(t_{i+1}) &= X_{i,s+1}, \qquad y_h(t_{i+1}) = Y_{i,s+1}.
\end{aligned} \tag{4.33}$$

Results concerning the application of half-explicit methods to index-one DAE are available in [13]; the application to index-three DAEs is discussed in [107].

4.5 Examples

Some illustrative examples with DAEs are discussed. Example 4.4 addresses an index-three mechanical multibody system of a car on a bumpy road. A docking maneuver of a satellite to a tumbling target is investigated in Example 4.5. Herein, the use of quaternions leads to a formulation with an index-one DAE.

Example 4.4 We consider a vehicle simulation for the ILTIS on a bumpy road section. A detailed description of the mechanical multibody system is provided in [128]. The system was modeled by SIMPACK [81] and the simulation results were obtained using the code export feature of SIMPACK and the BDF method DASSL [29]. The mechanical multibody system consists of 11 rigid bodies with a total of 25 degrees of freedom (DOF) (chassis with 6 DOF, wheel suspension with 4 DOF in total, wheels with 12 DOF in total, steering rod with 1 DOF, camera with 2 DOF). The motion is restricted by 9 algebraic constraints. Figure 2 illustrates the test track with bumps and the resulting pitch and roll angles, and the vertical excitation of the chassis. The integration tolerance within DASSL is set to 10^{-4} for the differential states and to 10^8 for the algebraic states (i.e., no error control was performed for the algebraic states).

Example 4.5 We consider a docking maneuver of a service satellite (S) to a tumbling object (T) on an orbit around the earth, compare [103]. Both objects are able to rotate freely in space and quaternions are used to parametrize their orientation. Note that, in contrast to Euler angles, quaternions lead to a continuous parametrization of the orientation without singularities.

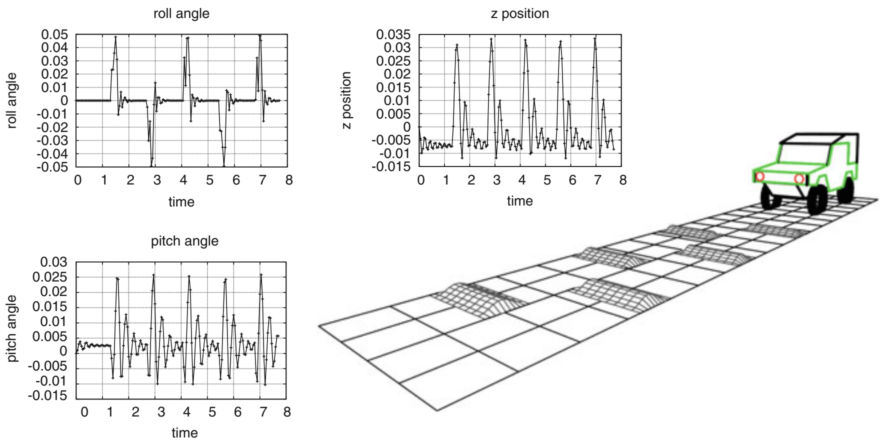

Fig. 2 Simulation results of the ILTIS on a bumpy road: roll angle, pitch angle, vertical excitation of chassis

The relative dynamics of S and T are approximately given by the *Clohessy-Wilshire-Equations*

$$\begin{aligned} x''(t) &= 2ny'(t) - 3n^2x(t) + a_x(t), \\ y''(t) &= -2nx'(t) + a_y(t), \\ z''(t) &= -n^2z(t) + a_z(t), \end{aligned}$$

where $(x, y, z)^\top$ is the relative position of S and T, $a = (a_x, a_y, a_z)^\top$ is a given control input (thrust) to S, $n = \sqrt{\mu/a_t^3}$, a_t is the semi-major axis of the orbit (assumed to be circular), and μ is the gravitational constant.

The direction cosine matrix using quaternions $q = (q_1, q_2, q_3, q_4)^\top$ is defined by

$$R(q)^\top = \begin{pmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2\,(q_1q_2 + q_3q_4) & 2\,(q_1q_3 - q_2q_4) \\ 2\,(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2\,(q_2q_3 + q_1q_4) \\ 2\,(q_1q_3 + q_2q_4) & 2\,(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{pmatrix}.$$

The matrix $R(q)$ represents the rotation matrix from rotated to non-rotated state. The orientation of S and T with respect to an unrotated reference coordinate system is described by quaternions $q^S = (q_1^S, q_2^S, q_3^S, q_4^S)^\top$ for S and $q^T = (q_1^T, q_2^T, q_3^T, q_4^T)^\top$ for T. With the angular velocities $\omega^S = (\omega_1^S, \omega_2^S, \omega_3^S)^\top$ and $\omega^T = (\omega_1^T, \omega_2^T, \omega_3^T)^\top$ the quaternions obey the differential equations

$$(q^\alpha)'(t) = \frac{1}{2}\begin{pmatrix} \omega^\alpha(t) \\ 0 \end{pmatrix} \otimes q^\alpha(t), \qquad \alpha \in \{S, T\}, \tag{4.34}$$

where the operator $\otimes$ is defined by

$$\begin{pmatrix} \omega \\ 0 \end{pmatrix} \otimes q = \begin{pmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix}.$$

Assuming a constant mass distribution and body fixed coordinate systems that coincide with the principle axes, S and T obey the gyroscopic equations

$$\begin{aligned} (\omega_1^S)'(t) &= \frac{1}{J_{11}^S}\left(\omega_2^S(t)\omega_3^S(t)\left(J_{22}^S - J_{33}^S\right) + u_1(t)\right), \\ (\omega_2^S)'(t) &= \frac{1}{J_{22}^S}\left(\omega_1^S(t)\omega_3^S(t)\left(J_{33}^S - J_{11}^S\right) + u_2(t)\right), \\ (\omega_3^S)'(t) &= \frac{1}{J_{33}^S}\left(\omega_2^S(t)\omega_1^S(t)\left(J_{11}^S - J_{22}^S\right) + u_3(t)\right), \end{aligned}$$

$$(\omega_1^T)'(t) = \frac{1}{J_{11}^T}\left(\omega_2^T(t)\omega_3^T(t)\left(J_{22}^T - J_{33}^T\right)\right),$$

$$(\omega_2^T)'(t) = \frac{1}{J_{22}^T}\left(\omega_1^T(t)\omega_3^T(t)\left(J_{33}^T - J_{11}^T\right)\right),$$

$$(\omega_3^T)'(t) = \frac{1}{J_{33}^T}\left(\omega_2^T(t)\omega_1^T(t)\left(J_{11}^T - J_{22}^T\right)\right).$$

Herein $u = (u_1, u_2, u_3)^\top$ denotes a time-dependent torque input to S.

The quaternions are normalized to one by the algebraic constraints

$$0 = (q_1^\alpha)^2 + (q_2^\alpha)^2 + (q_3^\alpha)^2 + (q_4^\alpha)^2 - 1, \qquad \alpha \in \{S, T\},$$

which has to be obeyed since otherwise a drift-off would occur owing to numerical discretization errors. In order to incorporate these algebraic constraints, we treat $(q_4^S, q_4^T)^\top$ as algebraic variables and drop the differential equations for q_4^S and q_4^T in (4.34). In summary, we obtain an index-one DAE with differential state $(x, y, z, x', y', z', \omega_1^S, \omega_2^S, \omega_3^S, q_1^S, q_2^S, q_3^S, \omega_1^T, \omega_2^T, \omega_3^T, q_1^T, q_2^T, q_3^T)^\top \in \mathbb{R}^{18}$, algebraic state $(q_4^S, q_4^T)^\top \in \mathbb{R}^2$, and time-dependent control input $(a, u)^\top \in \mathbb{R}^6$ for S.

Figure 3 shows some snapshots of a docking maneuver on the time interval $[0, 667]$ with initial states

$$q^S(0) = (0,0,0,1)^\top, \qquad q^T(0) = (-0.05, 0, 0, 0.99875)^\top,$$

$$\omega^S(0) = (0,0,0)^\top, \qquad \omega^T(0) = (0, 0.0349, 0.017453)^\top,$$

$$(x(0), y(0), z(0))^\top = (0, -100, 0)^\top, \qquad (x'(0), y'(0), z'(0))^\top = (0,0,0)^\top,$$

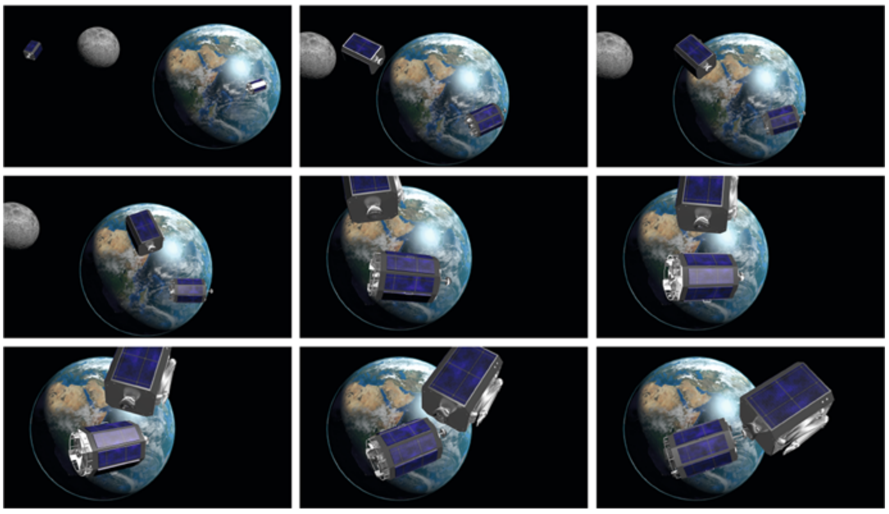
Fig. 3 Snapshots for the docking maneuver

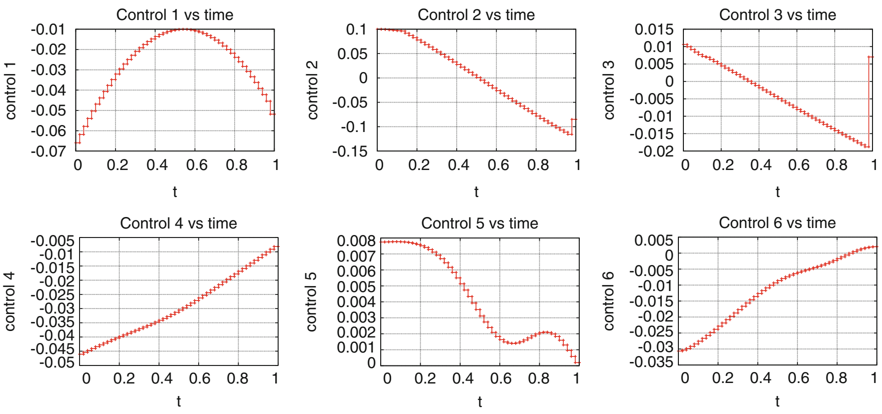

Fig. 4 Control input for the docking maneuver: ma_x, ma_y, ma_z with $m = 100$ (*top from left to right*), u_1, u_2, u_3 (*bottom from left to right*)

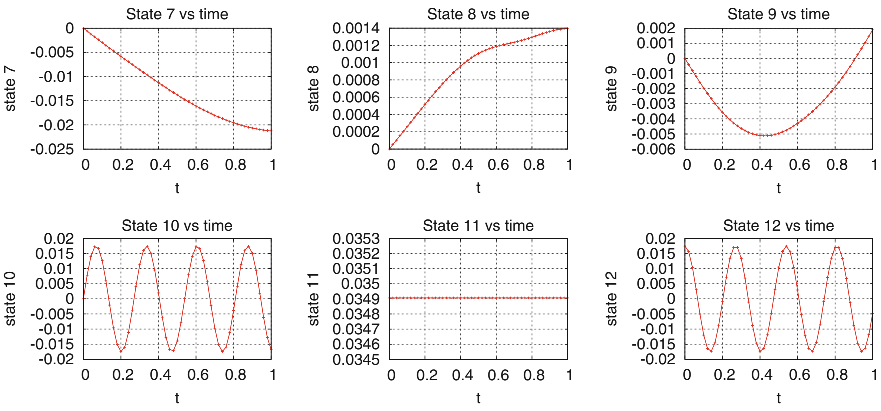

Fig. 5 Angular velocities of the service satellite S and the tumbling target T: ω_1^S, ω_2^S, ω_3^S (*top from left to right*), ω_1^T, ω_2^T, ω_3^T (*bottom from left to right*)

and parameters $a_t = 7071000$, $\mu = 398 \cdot 10^{12}$, $J_{11}^{\alpha} = 1000$, $J_{22}^{\alpha} = 2000$, $J_{33}^{\alpha} = 1000$, $\alpha \in \{S, T\}$. The integration tolerance within DASSL is set to 10^{-10} for the differential states and to 10^{-4} for the algebraic states. Figure 4 depicts the control inputs $m \cdot a = m \cdot (a_x, a_y, a_z)^{\top}$ with satellite mass $m = 100$ and $u = (u_1, u_2, u_3)^{\top}$. Finally, Fig. 5 shows the angular velocities ω^S and ω^T.

5 Co-simulation

In numerical system simulation, it is an essential task to simulate the dynamic interaction of different subsystems, possibly from different physical domains, modeled with different approaches, to be solved with different numerical solvers (multiphysical system models). Especially, in vehicle engineering, this becomes more and more important, because for a mathematical model of a modern passenger car or commercial vehicle, mechanical subsystems have to be coupled with flexible components, hydraulic subsystems, electronic and electric devices, and other control units. The mathematical models for all these subsystems are often given as DAE, but, typically, they substantially differ in their complexities, time constants, and scales; hence, it is not advisable to combine *all* model equations to *one entire* DAE and to solve it numerically with *one* integration scheme. In contrast, modern co-simulation strategies aim at using a specific numerical solver, i.e., DAE integration method, for each subsystem and to exchange only a limited number of coupling quantities at certain communication time points. Thus, it is important to analyze the behavior of such coupled simulation strategies, 'co-simulation', where the coupled subsystems are mathematically described as DAEs.

In addition to that, also the coupling may be described with an algebraic constraint equation; that is, DAE-related aspects and properties also arise here. Typical examples for such situations are network modeling approaches in general and, in particular, modeling of coupled electric circuits and coupled substructures of mechanical multibody systems, see [11].

Co-simulation techniques and their theoretical background are studied for a long time, see, for instance, the survey papers [83, 143]. In these days, a new interface standard has developed, the 'Functional Mock-Up Interface (FMI) for Model-Exchange and Co-Simulation', (https://www.fmi-standard.org/). This interface is supported from more and more commercial CAE-software tools and finds more and more interest in industry for application projects. Additionally, the development of that standard and its release has also stimulated new research activities concerning co-simulation.

A coupled system of $r \geq 2$ fully implicit DAEs initial value problems reads as

$$0 = F(t, z_i(t), z_i'(t), u_i(t)) = 0, \; t \in [t_0, t_f], \; z_i(t_0) = z_{i,0,} \quad i = 1, \dots r \tag{5.1}$$

with initial values assumed to be consistent and the *(subsystem-) outputs*

$$\xi_i(t) := \Xi_i(t, z_i(t), u_i(t)),$$

and the *(subsystem-) inputs* u_i that are given by *coupling conditions*

$$u_i(t) = h_i(\xi_1, \dots, \xi_r), \; i = 1, 2, \dots, \text{ i.e., } u = h(\xi),$$

where we have set $u := (u_1^\top, \dots, u_r^\top)^\top$, $\xi := (\xi_1^\top, \dots, \xi_r^\top)^\top$ and $h := (h_1^\top, \dots, h_r^\top)^\top : \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$, with $n_y = n_{y_1} + \dots + n_{y_r}$, $n_{u_1} + \dots + n_{u_r} = n_u$. Moreover, we assume here

$$\frac{\partial h_i}{\partial \xi_i} = 0,$$

that is, the inputs of system i do not depend on his own output. If the subsystems DAEs are in semi-explicit form, Eq. (5.1) has to be replaced by

$$\begin{aligned}\dot{x}_i(t) &= f_i(t, x_i(t), y_i(t), u_i(t)),\\ 0 &= g_i(t, x_i(t), y_i(t), u_i(t)),\end{aligned}$$

with $t \in [t_0, t_f]$ and $(x_i(t_0), y_i(t_0)) = (x_{i,0}, y_{i,0})$ with consistent initial values. This representation is called *block-oriented*; it describes the subsystems as blocks with inputs and outputs that are coupled.

In principle, it is possible to set up one monolithic system including the coupling conditions and output equations as additional algebraic equations:

$$\begin{aligned}\dot{x}_i &= f_i(t, x_i(t), y_i(t), u_i(t)),\\ 0 &= g_i(t, x_i(t), y_i(t), u_i(t)),\\ 0 &= u_i(t) - h(\xi(t)),\\ 0 &= \xi_i(t) - \Xi_i(t, x_i(t), u_i(t)), \quad i = 1, \dots, r.\end{aligned}$$

This entire system could be solved with one single integration scheme, which is, however, as indicated above typically not advisable. In contrast, in co-simulation strategies, also referred to as modular time-integration [125] or distributed time integration [11], the subsystem equations are solved separately on consecutive time-windows. Herein, the time integration of each subsystem within one time-window or macro step can be realized with a different step-size adapted to the subsystem (*multirate* approach), or even with different appropriate integration schemes (*multimethod* approach). During the integration process of one subsystem, the needed coupling quantities, i.e., inputs from other subsystems, are approximated—usually based on previous results. At the end of each macro step, coupling data is exchanged. To be more precise, for the considered time interval, we introduce a (macro) time grid $\mathbb{G} := \{T_0, \dots, T_N\}$ with $t_0 = T_0 < T_1 < \dots < T_N = t_f$. Then, the mentioned time-windows or macro steps are given by $[T_n, T_{n+1}]$, $n = 0, \dots, N-1$ and each subsystem is integrated independently from the others in each macro step $T_n \to T_{n+1}$, only using a typically limited number of coupling quantities as information from the other subsystems. The macro time points T_n are also called communication points, since here, typically, coupling data is exchanged between the subsystems.

5.1 Jacobi, Gauss-Seidel, and Dynamic-Iteration Schemes

An overview on co-simulation schemes and strategies can be found, e.g., in [11, 104, 125]. There are, however, two main approaches how the above sketched co-simulation can be realized. The crucial differences are the strategy (order) how the subsystems are integrated within the macro steps and, accordingly, how coupling quantities are handled and approximated. The first possible approach is a completely *parallel* scheme and is called *Jacobi scheme* (or co-simulation/coupling of Jacobi-type). As the name indicates, the subsystems are integrated here in parallel and, thus, they have to use extrapolated input quantities during the current macro step, cf. Fig. 6. In contrast to this, the second approach is a *sequential* one, it is called *Gauss-Seidel scheme* (or co-simulation/coupling of Gauss-Seidel-type). For the special case of two coupled subsystems, $r = 2$, this looks as follows: one subsystem is integrated first on the current macro step using extrapolated input data yielding a (numerical) solution for this first system. Then, the second subsystem is integrated on the current macro step but, then, using already computed results from the first subsystem for the coupling quantities (since results from the first subsystem for the current macro step are available, in fact). The results from the first subsystem may be available on a fine micro time grid—within the macro step—or even as function of time, e.g., as dense output from the integration method; additionally, (polynomial) interpolation may also be used, cf. Fig. 6.

The sequential Gauss-Seidel scheme can be generalized straightforwardly to $r > 2$ coupled subsystems: The procedure is sequential, i.e., the subsystems are numerically integrated one after another and for the integration of the i-th subsystem results from the subsystems $1, \dots, i-1$ are available for the coupling quantities, whereas data from chronologically upcoming subsystems $i+1, \dots, r$ have to be extrapolated based on information from previous communication points.

The extra- and interpolation, respectively, are realized using data from previous communication points and, typically, polynomial extra- and interpolation approaches are taken. That is, in the macro step $T_n \to T_{n+1}$, the input of subsystem i is extrapolated using data from the communication points $T_{n-k}, \dots, T_n$,

$$\tilde{u}_i(t) = \Psi_i(t; u_i(T_{n-k}), \dots, u_i(T_n)) = \sum_{j=0}^{k} u_i(T_{n-j}) \prod_{l=0, l\neq j}^{k} \frac{t - T_{n-l}}{T_{n-j} - T_{n-l}},$$

$t \in [T_n, T_{n+1}]$ and with the extrapolation polynomial Ψ_i with degree $\leq k$; for interpolation, e.g., for Gauss-Seidel schemes, we have correspondingly

$$\tilde{u}_i(t) = \Psi_i(t; u_i(T_{n-k}), \dots, u_i(T_{n+1})) = \sum_{j=0}^{k+1} u_i(T_{n+1-j}) \prod_{l=0, l\neq j}^{k+1} \frac{t - T_{n+1-l}}{T_{n+1-j} - T_{n+1-l}}.$$

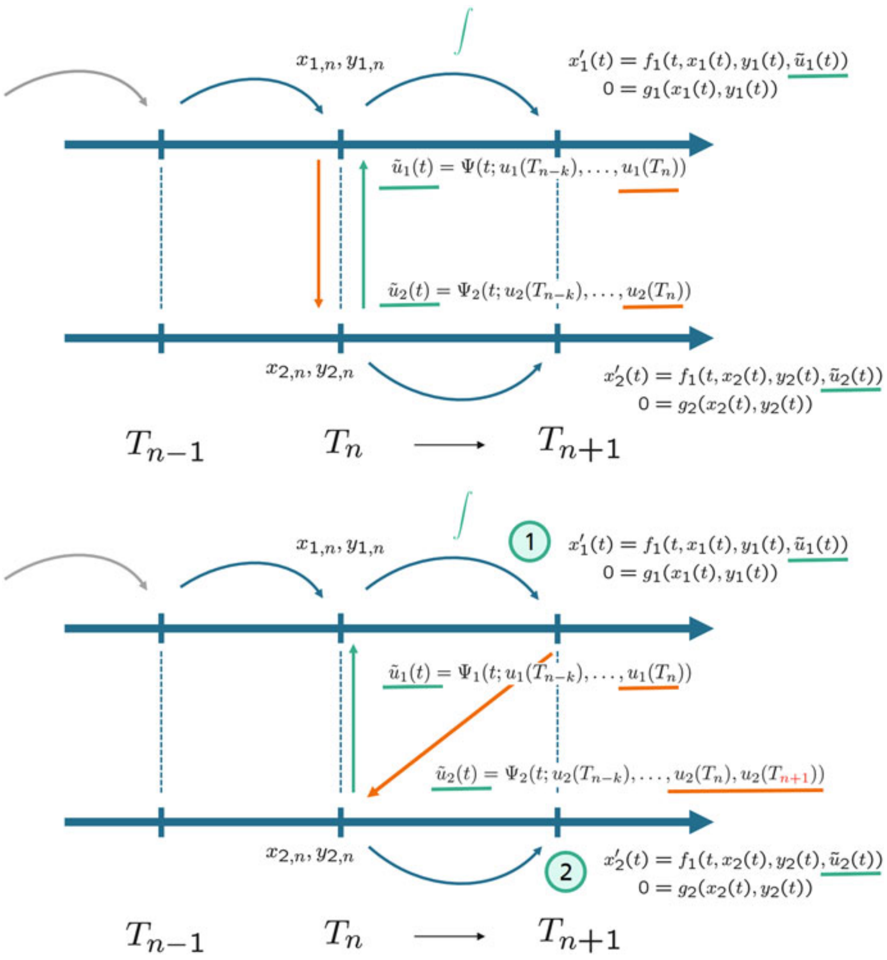
$\int$
$x_{1,n}, y_{1,n}$
$x_1'(t) = f_1(t, x_1(t), y_1(t), \tilde{u}_1(t))$
$0 = g_1(x_1(t), y_1(t))$
$\tilde{u}_1(t) = \Psi(t; u_1(T_{n-k}), \dots, u_1(T_n))$
$\tilde{u}_2(t) = \Psi_2(t; u_2(T_{n-k}), \dots, u_2(T_n))$
$x_{2,n}, y_{2,n}$
$x_2'(t) = f_1(t, x_2(t), y_2(t), \tilde{u}_2(t))$
$0 = g_2(x_2(t), y_2(t))$
T_{n-1}
T_n
T_{n+1}
$\int$
1
$x_{1,n}, y_{1,n}$
$x_1'(t) = f_1(t, x_1(t), y_1(t), \tilde{u}_1(t))$
$0 = g_1(x_1(t), y_1(t))$
$\tilde{u}_1(t) = \Psi_1(t; u_1(T_{n-k}), \dots, u_1(T_n))$
$\tilde{u}_2(t) = \Psi_2(t; u_2(T_{n-k}), \dots, u_2(T_n), u_2(T_{n+1}))$
$x_{2,n}, y_{2,n}$
2
$x_2'(t) = f_1(t, x_2(t), y_2(t), \tilde{u}_2(t))$
$0 = g_2(x_2(t), y_2(t))$
T_{n-1}
T_n
T_{n+1}

Fig. 6 Jacobi (*upper diagram*) and Gauss-Seidel (*lower diagram*) co-simulation schemes

The most simple extrapolation is that of zero-order, $k = 0$, leading to 'frozen' coupling quantities

$$u_i(t) = u_i(T_n), \quad t \in [T_n, T_{n+1}].$$

A third approach to establish a simulation of coupled systems are the so-called *dynamic iteration schemes*, [11, 20, 21], also referred to as waveform relaxation methods, [82, 94]. Here, the basic idea is to solve the subsystems iteratively on each macro step using coupling data information from previous iteration steps, in order to decrease simulation errors. How the subsystems are solved in each iteration step can be in a sequential fashion (Gauss-Seidel) or all in parallel (Jacobi or Picard), cf. [11, 20]. The schemes defined above are contained in a corresponding dynamic iteration scheme by performing exactly one iteration step.

5.2 Stability and Convergence

First of all, we point out that there is a decisive difference between convergence and stability issues for coupled ODEs on the one hand and for coupled DAEs on the other hand. The stability problems that may appear for coupled ODEs with stiff coupling terms resemble the potential problems when applying an explicit integration method to stiff ODEs—thus, these difficulties can be avoided by using sufficiently small macro step-sizes $H_n = T_{n+1} - T_n$, cf. [9, 11, 104]. In the DAE-case, however, reducing the macro steps does not generally lead to an improvement; here, it is additionally essential that a certain contractivity condition is satisfied, see [9, 11, 21, 125].

5.2.1 The ODE-Case

For problems with coupled ODEs, convergence is studied, e.g., in [8, 10, 16, 17]. For coupled ODEs systems that are free of algebraic loops—this is guaranteed, for instance, provided that there is no direct feed-through, i.e., $\partial \Xi_i / \partial u_i = 0, \quad i = 1, \ldots, r$, for a precise definition see [10, 16]—we have the following global error estimation for a co-simulation with a Jacobi scheme with constant macro step-size $H > 0$ assumed to be sufficiently small,

$$\varepsilon^x \leq C \left(\sum_{i=1}^{r} \varepsilon_i^x + H^{k+1} \right), \tag{5.2}$$

where k denotes the order of the extrapolation and ε_i^x is the global error in subsystem i and ε^x is the overall global error, cf. [8, 10]. That is, the errors from the subsystems contribute to the global error, as well as the error from extra-(inter-)polation, $\mathcal{O}(H^{k+1})$. These results can be straightforwardly deduced following classical convergence analysis for ODE time integration schemes.

5.2.2 The DAE-Case

For detailed analysis and both convergence and stability results for coupled DAE systems, we refer the reader to [9, 11, 20, 125] and the literature cited therein. In the sequel we summarize and sketch some aspects from these research papers.

As already said, in the DAE case, the situation becomes more difficult. Following the lines of [20], we consider the following coupled DAE-IVP representation

$$x_i'(t) = f_i(x(t), y(t)), \tag{5.3}$$

$$0 = g_i(x(t), y(t)), \quad i = 1, \ldots, r, \tag{5.4}$$

with $x = (x_1^\top, \ldots, x_r^\top)^\top$, $y = (y_1^\top, \ldots, y_r^\top)^\top$ and initial conditions $(x_i(t_0), y_i(t_0)) = (x_{i,0}, y_{i,0})$, $i = 1, \ldots, r$. For the following considerations, we assume that the IVP(s) possess a unique global solution and that the right-hand side functions f_i, g_i are sufficiently often continuously differentiable and, moreover, that it holds

$$\frac{\partial g_i}{\partial y_i}$$

is non-singular for $i = 1, \ldots, r$ in a neighborhood of a solution (index-one condition for each subsystem). Notice that this representation differs from the previously stated block-oriented form. Equations (5.3)–(5.4) are, however, more convenient, in order to derive and to state the mentioned stability conditions, the coupling here is realized by the fact that all right-hand side functions f_i, g_i of each subsystem do depend on the entire differential and algebraic variables.

As before, we denote by a $\tilde{\cdot}$ quantities that are only available as extra- or interpolated quantity. Thus, establishing a co-simulation scheme of Jacobi-type yields for the i-th subsystem in macro step $T_n \to T_{n+1}$

$$\begin{aligned} x_{i,n}' &= f_i(\tilde{x}_{1,n}, \ldots, \tilde{x}_{i-1,n}, x_{i,n}, \tilde{x}_{i+1,n}, \ldots, \tilde{x}_{r,n}, \\ &\qquad \tilde{y}_{1,n}, \ldots, \tilde{y}_{i-1,n}, y_{i,n}, \tilde{y}_{i+1,n}, \ldots, \tilde{y}_{r,n}), \\ 0 &= g_i(\tilde{x}_{1,n}, \ldots, \tilde{x}_{i-1,n}, x_{i,n}, \tilde{x}_{i+1,n}, \ldots, \tilde{x}_{r,n}, \\ &\qquad \tilde{y}_{1,n}, \ldots, \tilde{y}_{i-1,n}, y_{i,n}, \tilde{y}_{i+1,n}, \ldots, \tilde{y}_{r,n}). \end{aligned}$$

Accordingly, for a Gauss-Seidel-type scheme, we obtain

$$\begin{aligned} x_{i,n}' &= f_i(x_{1,n}, \ldots, x_{i,n}, \tilde{x}_{i+1,n}, \ldots, \tilde{x}_{r,n}, \\ &\qquad y_{1,n}, \ldots, y_{i,n}, \tilde{y}_{i+1,n}, \ldots, \tilde{y}_{r,n}), \\ 0 &= g_i(x_{1,n}, \ldots, x_{i,n}, \tilde{x}_{i+1,n}, \ldots, \tilde{x}_{r,n}, \\ &\qquad y_{1,n}, \ldots, y_{i,n}, \tilde{y}_{i+1,n}, \ldots, \tilde{y}_{r,n}). \end{aligned}$$

With $g = (g_1, \ldots, g_r)^\top$, a sufficient (not generally necessary) *contractivity condition for stability* is derived and proven in [20]. The condition is given by

$$\alpha := \|g_y^{-1} g_{\tilde{y}}\| < 1,$$

with $\tilde{y} = (\tilde{y}_1, \dots, \tilde{y}_r)$. For a detailed list of requirements and assumptions to be taken as well as for a proof and consequences, the reader is referred to [20, 21]. For the special case $r = 2$, the above condition leads to the following for the Jacobi-type scheme:

$$\alpha = \left\| \begin{pmatrix} 0 & g_{1,y_1}^{-1} g_{1,y_2} \\ g_{2,y_2}^{-1} g_{2,y_1} & 0 \end{pmatrix} \right\| < 1,$$

whereas, for the Gauss-Seidel-type scheme, we obtain

$$\alpha = \left\| \begin{pmatrix} g_{1,y_1} & 0 \\ g_{2,y_1} & g_{2,y_2} \end{pmatrix}^{-1} \begin{pmatrix} 0 & g_{1,y_2} \\ 0 & 0 \end{pmatrix} \right\| < 1.$$

An immediate consequence is that for a Jacobi-scheme of two coupled DAEs with no coupling in the algebraic equation, i.e., $g_{i,y_j} = 0$, for $i \neq j$, we have $\alpha = 0$.

As a further example, we discuss two mechanical multibody systems coupled via a kinematic constraint:

$$\begin{aligned} M_i(q_i)q_i'' &= \psi_i(q_i, q_i') - G_i^\top(q)\lambda, \quad i = 1, 2, \\ 0 &= \gamma(q), \end{aligned}$$

with $q = (q_1^\top, q_2^\top)^\top$ and $G(q) := \partial\gamma/\partial q$ and $G_i(q) := \partial\gamma/\partial q_i$, $i = 1, 2$. Performing an index reduction by twice differentiating the coupling constraint and setting $v_i := q_i'$, $a_i := v_i'$ as well as $x_i := (q_i^\top, v_i^\top)^\top$ and $y_1 = a_1^\top$, $y_2 := (a_2^\top, \lambda^\top)^\top$, $f_i := (v_i^\top, a_i^\top)^\top$, we are in the previously stated general framework:

$$\begin{aligned} x_1' &= f_1 & x_2' &= f_2 \\ 0 &= M_1 a_1 - \psi_1 + G_1^\top \lambda & 0 &= \begin{bmatrix} M_2 a_2 - \psi_2 + G_2^\top \lambda \\ G_1 a_1 + G_2 a_2 + \gamma^{(II)} \end{bmatrix} \\ &=: g_1(x_1, x_2, y_1, y_2) & &=: g_2(x_1, x_2, y_1, y_2) \end{aligned} .$$

Herein, $\gamma^{(II)}$ contains the remainder of the second derivative of γ without the term $G_1 a_1 + G_2 a_2$.

That is, the only coupling is via algebraic variables and in algebraic equations. If we set up a Jacobi-scheme, in macro step $T_n \to T_{n+1}$, in subsystem 1, we have to use extrapolated values from subsystem 2, i.e., y_2 is replaced by

$$\tilde{y}_2(t) = \Psi_1^y(t; y_2(T_{n-k}), \dots, y_2(T_n))$$

and in subsystem 1, accordingly, $\tilde{y}_1(t) = \Psi_2^y(t; y_1(T_{n-k}), \ldots, y_1(T_n))$. The above contractivity condition in this case reads

$$\left\| \begin{pmatrix} 0 & 0 & M_1^{-1}G_1^\top \\ M_2^{-1}G_2^\top R_2^{-1}G_1 & 0 & 0 \\ -R_2^{-1}G_1 & 0 & 0 \end{pmatrix} \right\| < 1,$$

with $R_i := G_i M_i^{-1} G_i^\top$, $i = 1, 2$.

Analogously, we can consider a Gauss-Seidel-type scheme. Starting with subsystem 1, we have to extrapolate here y_2 from previous macro steps yielding x_1, y_1, which then can be evaluated during time-integration of subsystem 2. Stating the contractivity condition for this case and noticing that only the algebraic variable λ has to be extrapolated from previous time points, the relevant (λ-)part of the condition requires

$$\|R_2^{-1}R_1\| = \|(G_2M_2G_2^\top)^{-1}(G_1M_1^{-1}G_1^\top)\| < 1. \tag{5.5}$$

We observe in both cases that mass and inertia properties of the coupled systems may strongly influence the stability of the co-simulation. In particular for the latter sequential Gauss-Seidel scheme, the order of integration has an essential impact on stability, i.e., the choice of system 1 and 2, respectively, should be taken such that the left-hand side of (5.5) is as small as possible.

This result has been developed and proven earlier in [11] for a more general framework, which is slightly different than our setup and for which the coupled mechanical systems are also a special case. In that paper, a method for stabilization (reducing α) is suggested. In [125], the authors also study stability and convergence of coupled DAE systems in a rather general framework and propose a strategy for stabilization as well.

For the specific application field of electric circuit simulation, the reader is referred to [20, 21] and the references therein. A specific consideration of coupled mechanical multibody systems is provided in [8, 9] and in [126], where the coupling of a multibody system and a flexible structure is investigated and an innovative coupling strategy is proposed. Lately, analysis results on coupled DAE systems solved with different co-simulation strategies and stabilization approaches are provided by the authors of [129, 130]. In [19], a multibody system model of a wheel-loader described as index-three DAE in a commercial software package is coupled with a particle code for soft-soil modeling, in order to establish a coupled digging simulation.

The general topic of coupled DAE system is additionally discussed in the early papers [82, 89, 94].

A multirate integrator for constrained dynamical systems is derived in [96], which is based on a discrete variational principle. The resulting integrator is symplectic and momentum preserving.

6 Real-Time Simulation

An important field in modern numerical system simulation is *real-time scenarios*. Here, a numerical model is coupled with the real world and both are interacting dynamically. A typical area, in which such couplings are employed, is interactive simulators ('human/man-in-the-loop'), such as driving simulators or flight simulators, see [58], but also interactively used software (simulators), e.g., for training purposes, cf. [98]. Apart from that, real-time couplings are used in tests for electronic control units (ECU tests) and devices ('hardware-in-the-loop'—HiL), see, e.g., [15, 122] and in the field of model based controllers ('model/software-in-the-loop'—MiL/SiL), see, e.g., [42, 43].

It is characteristic for all the mentioned fields that a numerical model *replaces* a part of the real world. In case of an automotive control unit test, the real control unit hardware is coupled with a numerical model of the rest of the considered vehicle; in case of an interactive driving simulator, the simulator hardware and, by that, the driver or the operator, respectively, is also coupled with a virtual vehicle. The benefits of such couplings are tremendous—tests and studies can be performed under fully accessible and reproducible conditions in the laboratory. Investigations and test runs with real cars and drivers can be reduced and partially avoided, which can save time, costs, and effort substantially. From the perspective of the numerical model, it receives from the real world environment signals as inputs (e.g., the steering-wheel angle from human driver in a simulator) and gives back its dynamical behavior as output (e.g., the car's reaction is transmitted to the simulator hardware, which, in turn, follows that motion making the driver feel as he would sit in a real car). It is crucial for a realistic realization of such a coupling that the simulation as well as the communication are sufficiently fast. That is, after delivering an input to the numerical model, the real world component expects a response after a fixed time ΔT—and the numerical model has to be simulated for that time span and has to feed back the response *on time*. Necessary for that is that the considered numerical simulation satisfies the *real-time condition*: the computation (or simulation) time ΔT_{comp} has to be smaller or equal than the simulated time ΔT.

Physical models are often described as differential equations (mechanical multibody systems that represent a vehicle model). Satisfying the real-time condition here means accordingly that the numerical time integration of the IVP

$$F(t, z(t), z'(t), u(t)) = 0, \quad t \in [T_i; T_i + \Delta T]$$
$$z(T_i) = z_{0,i},$$

is executed with a total computation time that is smaller or equal than ΔT. If a complete real-time simulation shall be run on a time horizon $[t_0; t_f]$ which is divided by an equidistant time-grid $\{T_0, \dots, T_N\}$, $t_0 = T_0$, $t_f = T_N$, $T_{i+1} - T_i = \Delta T$, the real-time condition must be guaranteed for any subinterval of length ΔT. In fact, this is a coupling exactly as in classical co-simulation—with the decisive difference that one partner is not a numerical model, but a real world component and, thus, the numerical model simulation must satisfy the real-time condition. Obviously, whether or not the real-time condition can be satisfied, strongly depends both on the numerical time integration method and the differential equation and its properties itself. In principle, any time integration method can be applied, provided that the resulting simulation satisfies the real-time condition.

The fulfillment of the real-time condition as stated above has, however, to be assured *deterministically* in each macro time step $T_i \rightarrow T_i + \Delta T$—at least in applications, where breaking this condition leads to a critical system shutdown (e.g., hardware simulators, HiL-tests). Whence, the chosen integration methods should not have indeterministic elements like step-size control or iterative inner methods (solution of nonlinear systems by Newton-like methods): varying iteration numbers lead to a varying computation time. Consequently, for real-time application, time integration methods with fixed time-steps and with a fixed number of possible iterations are preferred. Additionally, to save computation time, typically, low-order methods are in use, which is also caused by the fact that in the mentioned application situations, the coupled simulation needs not to be necessarily highly accurate, but stable.

6.1 Real-Time Integration of DAEs

For non-stiff ODE models, which have to be simulated under real-time conditions, even the simple explicit Euler scheme is frequently used. For stiff ODEs, the linearly implicit methods as discussed in Sect. 4.3 are evident, since for these method class, only linear systems have to be solved internally, which leads to an a priori known, fixed, and moderate computational effort, see [14, 15, 49, 118] and the references therein.

Since all typical and work-proven DAE time integration methods are at least partially implicit leading to the need of iterative computations, it is a common approach to avoid DAE models for real-time applications already in the modeling process (generally, for real-time applications, often specific modeling techniques are applied), whenever it is possible. However, this is often impossible in many application cases of practical relevance. For instance, the above-mentioned examples from the automotive area require a mechanical vehicle model, which is usually realized as mechanical multibody system model, whose underlying equations of motion are often a DAE as stated in Eq. (2.13). Thus, there is a need for DAE time integration schemes that are stable and highly efficient also for DAEs of realistic complexities.

Time integration methods for DAEs with a special focus on real-time applications and the fulfillment of the real-time condition are addressed, e.g., in [14, 15, 31, 32, 39, 44, 49, 50, 119]. In the sequel, we present a specific integration method for the MBS equations of motion (2.13) in its index-two formulation on velocity-level.

For the special case of the semi-explicit DAE describing a mechanical multibody system, compare (2.13), the following linearly implicit method can be applied, which is based on the linearly implicit Euler scheme. The first step is to reduce the index from three to two by replacing the original algebraic equations by its first time-derivative,

$$G(q)v = 0,$$

which is linear in v. The numerical scheme proposed in [14, 31] consists in handling the time-step for the position coordinates explicitly and requiring that the algebraic equation on velocity level is satisfied, i.e.,

$$G(q_{i+1})v_{i+1} = 0.$$

In particular, this leads to the set of linear equations as follows

$$q_{i+1} = q_i + h_i v_i,$$
$$\begin{pmatrix} M - hJ_v - h^2 J_v & G^\top(q_i) \\ G(q_{i+1}) & 0 \end{pmatrix} \begin{pmatrix} v_{i+1} - v_i \\ h\lambda_i \end{pmatrix} = \begin{pmatrix} hf_i + h^2 J_q v_i \\ -G(q_{i+1})v_i \end{pmatrix},$$

where $J_{q/v} := \partial f / \partial(q/v)(q_i, v_i)$.

An important issue is naturally the *drift-off*, cf. Sect. 2, in the neglected algebraic constraints—here, in the above method for the index-two version of the MBS DAE, the error in the algebraic equation on position-level, i.e., $0 = g(q)$, may grow linearly in time; this effect is even more severe, since a low-order method is in use. Classical strategies to stabilize this drifting are projection approaches, cf., e.g., [73, 100], which are usually of adaptive and iterative character. The authors in [14, 31] propose and discuss a non-iterative projection strategy, which consists, in fact, in one special Newton-step for the KKT conditions related to the constrained optimization problem that is used for projection; thus, only one additional linear equation has to be solved in each time-step. The authors show that using this technique leads to a bound for the error on position level, which is independent of time. An alternative way to stabilize the drift-off effect without substantially increasing the computational effort is the Baumgarte stabilization, cf. Sect. 2 and [31, 48, 122].

7 Parametric Sensitivity Analysis and Adjoints

The parametric sensitivity analysis is concerned with parametric initial value problems subject to DAEs on the interval $[t_0, t_f]$ given by

$$F(t, z(t), z'(t), p) = 0, \tag{7.1}$$

$$z(t_0) = z_0(p), \tag{7.2}$$

where $p \in \mathbb{R}^m$ is a parameter vector and the mapping $z_0 : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ is at least continuously differentiable. We assume that the initial value problem possesses a solution for every p in some neighborhood of a given nominal parameter $\hat{p}$ and denote the solution by $z(t;p)$. In order to quantify the influence of the parameter on the solution, we are interested in the so-called *sensitivities* (*sensitivity matrices*)

$$S(t) := \frac{\partial z}{\partial p}(t;\hat{p}) \qquad \text{for } t \in [t_0, t_f]. \tag{7.3}$$

Throughout we tacitly assume that the sensitivities actually exist.

In many applications, e.g., from optimal control or optimization problems involving DAEs, one is not directly interested in the sensitivities $S(\cdot)$ themselves but in the gradient of some function $g : \mathbb{R}^m \longrightarrow \mathbb{R}$ defined by

$$g(p) := \varphi(z(t_f;p),p), \tag{7.4}$$

where $\varphi : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$ is continuously differentiable. Of course, if the sensitivities $S(\cdot)$ are available, the gradient of g at $\hat{p}$ can easily be computed by the chain rule as

$$\nabla g(\hat{p}) = S(t_f)^\top \nabla_z \varphi(z(t_f;\hat{p}),\hat{p}) + \nabla_p \varphi(z(t_f;\hat{p}),\hat{p}). \tag{7.5}$$

However, often the explicit computation of S is costly and should be avoided. Then the question for alternative representations of the gradient $\nabla g(\hat{p})$ arises, which avoids the explicit computation of S. This alternative representation can be derived using an adjoint DAE. Both approaches are analytical in the sense that they provide the correct gradient, if round-off errors are not taken into account.

Remark 7.1 The computation of the gradient using S is often referred to as the *forward mode* and the computation using adjoints as the *backward* or *reverse mode* in the context of automatic differentiation, compare [72]. Using automatic differentiation is probably the most convenient way to compute the above gradient, since powerful tools are available, see the web-page familywww.autodiff.org.

The same kind of sensitivity investigations can be performed either for the problem (7.1)–(7.2) in continuous time or for discretizations thereof by means of one-step or multi-step methods.

7.1 Sensitivity Analysis in Discrete Time

7.1.1 The Forward Mode

Suppose a suitable discretization scheme of (7.1)–(7.2) is given, which provides approximations $z_h(t_i;p)$ at the grid points $t_i \in \mathbb{G}_h$ in dependence on the parameter

p. We are interested in the sensitivities

$$S_h(t_i) := \frac{\partial z_h}{\partial p}(t_i;\hat{p}) \in \mathbb{R}^{n\times m} \qquad \text{for } t_i \in \mathbb{G}_h$$

for a nominal parameter $\hat{p} \in \mathbb{R}^m$. As the computations are performed on a finite grid, these sensitivities can be obtained by differentiating the discretization scheme with respect to p. This procedure is called internal numerical differentiation (IND) and was introduced in [25].

To be more specific, let $\hat{p}$ be a given nominal parameter and consider the one-step method

$$z_h(t_0;\hat{p}) = z_0(\hat{p}), \tag{7.6}$$

$$z_h(t_{i+1};\hat{p}) = z_h(t_i;\hat{p}) + h_i\Phi(t_i, z_h(t_i;\hat{p}), h_i, \hat{p}), \quad i = 0,1,\ldots,N-1. \tag{7.7}$$

Differentiating both equations with respect to p and evaluating the equations at $\hat{p}$ yields

$$S_h(t_0) = z_0'(\hat{p}), \tag{7.8}$$

$$S_h(t_{i+1}) = S_h(t_i) + h_i\left(\frac{\partial\Phi}{\partial z}[t_i]S_h(t_i) + \frac{\partial\Phi}{\partial p}[t_i]\right), \quad i = 0,1,\ldots,N-1. \tag{7.9}$$

Herein, we used the abbreviation $[t_i]$ for $(t_i, z_h(t_i;\hat{p}), h_i, \hat{p})$. Evaluation of (7.8)–(7.9) yields the desired sensitivities $S_h(t_i)$ of $z_h(t_i;\hat{p})$ at the grid points, if the increment function Φ of the one-step method and the function z_0 are differentiable with respect to z and p, respectively. Note that the function z_0 can be realized by the projection method in LSQ(p) in Sect. 3.2 and sufficient conditions for its differentiability are provided by Theorem 3.1.

The computation of the partial derivatives of Φ is more involved. For a Runge–Kutta method Eqs. (4.7)–(4.9) (with an additional dependence on the parameter p) have to be differentiated with respect to z and p. Details can be found in [68, Sect. 5.3.2].

The same IND approach can be applied to multi-step methods. Differentiation of the scheme (4.2) and the consistent initial values

$$z_h(t_0;p) = z_0(p),\ z_h(t_1;p) = z_1(p),\ \ldots,\ z_h(t_{s-1};p) = z_{s-1}(p)$$

with respect to p and evaluation at $\hat{p}$ yields the formal scheme

$$S_h(t_\ell) = z_\ell'(\hat{p}), \qquad \ell = 0,\ldots,s-1,$$

$$S_h(t_{i+s}) = \sum_{\ell=0}^{s-1} \frac{\partial\Psi}{\partial z_{i+\ell}} \cdot S_h(t_{i+\ell}) + \frac{\partial\Psi}{\partial p}, \qquad i = 0,\ldots,N-s.$$

More specifically, for an s-stage BDF method the function ψ is implicitly given by (4.4) (with an additional dependence on the parameter p). Differentiation of (4.4) with respect to p yields

$$\left(\frac{\partial F}{\partial z}[t_{i+s}]+\frac{\alpha_s}{h_{i+s-1}}\frac{\partial F}{\partial z'}[t_{i+s}]\right)S_h(t_{i+s})+\sum_{\ell=0}^{s-1}\frac{\alpha_\ell}{h_{i+s-1}}\frac{\partial F}{\partial z'}[t_{i+s}]S_h(t_{i+\ell})+\frac{\partial F}{\partial p}[t_{i+s}]=0 \tag{7.10}$$

and, if the iteration matrix $M := F'_z[t_{i+s}] + \frac{\alpha_s}{h_{i+s-1}}F'_{z'}[t_{i+s}]$ is non-singular,

$$S_h(t_{i+s}) = -M^{-1}\cdot\left(\sum_{\ell=0}^{s-1}\frac{\alpha_\ell}{h_{i+s-1}}\frac{\partial F}{\partial z'}[t_{i+s}]S_h(t_{i+\ell}) + \frac{\partial F}{\partial p}[t_{i+s}]\right).$$

Herein, we used the abbreviation $[t_{i+s}] = \left(t_{i+s}, z_h(t_{i+s}), \frac{1}{h_{i+s-1}}\sum_{k=0}^{s}\alpha_k z_h(t_{i+k})\right)$.

7.1.2 The Backward Mode and Adjoints

Consider the function g in (7.4) subject to a discretization scheme, i.e.

$$g_h(p) := \varphi(z_h(t_N;p),p). \tag{7.11}$$

We intend to compute the gradient of g_h at $\hat{p}$. Using the sensitivity $S_h(t_N)$ the gradient is given by

$$\nabla g_h(\hat{p}) = S_h(t_N)^\top \nabla_z\varphi(z_h(t_N;\hat{p}),\hat{p}) + \nabla_p\varphi(z_h(t_N;\hat{p}),\hat{p}).$$

Now we are interested in an alternative representation of the gradient without the sensitivity $S_h(t_N)$. To this end consider the one-step method in (7.6)–(7.7). Following [68, Sect. 5.3.2] define the auxiliary functional

$$g_h^a(p) := g_h(p) + \sum_{i=0}^{N-1}\lambda_h(t_{i+1})^\top\left(z_h(t_{i+1};p) - z_h(t_i;p) - h_i\Phi(t_i,z_h(t_i;p),h_i,p)\right)$$

with multipliers $\lambda_h(t_1),\ldots,\lambda_h(t_N)$ that will be specified later. Note that $g_h^a \equiv g_h$ for all discrete trajectories satisfying (7.6)–(7.7). The gradient of g_h^a at $\hat{p}$ computes to

$$\begin{aligned}\nabla g_h^a(\hat{p}) &= S_h(t_N)^\top\nabla_z\varphi(z_h(t_N;\hat{p}),\hat{p}) + \nabla_p\varphi(z_h(t_N;\hat{p}),\hat{p})\\ &\quad+\sum_{i=0}^{N-1}\left(S_h(t_{i+1}) - S_h(t_i) - h_i\frac{\partial\Phi}{\partial z}[t_i]S_h(t_i) - h_i\frac{\partial\Phi}{\partial p}[t_i]\right)^\top\lambda_h(t_{i+1})\end{aligned}$$

$$
\begin{aligned}
&= S_h(t_N)^\top \nabla_z \varphi(z_h(t_N;\hat p),\hat p) + \nabla_p \varphi(z_h(t_N;\hat p),\hat p) \\
&\quad + \sum_{i=1}^{N} S_h(t_i)^\top \lambda_h(t_i) - \sum_{i=0}^{N-1} \left(S_h(t_i) + h_i \frac{\partial \Phi}{\partial z}[t_i] S_h(t_i) \right)^\top \lambda_h(t_{i+1}) \\
&\quad - \sum_{i=0}^{N-1} h_i \frac{\partial \Phi}{\partial p}[t_i]^\top \lambda_h(t_{i+1}) \\
&= S_h(t_N)^\top \left(\lambda_h(t_N) + \nabla_z \varphi(z_h(t_N;\hat p),\hat p)\right) + \nabla_p \varphi(z_h(t_N;\hat p),\hat p) \\
&\quad + \sum_{i=1}^{N-1} S_h(t_i)^\top \left(\lambda_h(t_i) - \lambda_h(t_{i+1}) - h_i \frac{\partial \Phi}{\partial z}[t_i]^\top \lambda_h(t_{i+1}) \right) \\
&\quad - S_h(t_0)^\top \left(\lambda_h(t_1) + h_0 \frac{\partial \Phi}{\partial z}[t_0]^\top \lambda_h(t_1) \right) - \sum_{i=0}^{N-1} h_i \frac{\partial \Phi}{\partial p}[t_i]^\top \lambda_h(t_{i+1})
\end{aligned}
$$

In order to eliminate the sensitivities, we choose the multipliers λ_h such that they satisfy the *adjoint equations*

$$\lambda_h(t_N) = -\nabla_p \varphi(z_h(t_N;\hat p),\hat p), \tag{7.12}$$

$$\lambda_h(t_i) = \lambda_h(t_{i+1}) + h_i \frac{\partial \Phi}{\partial z}[t_i]^\top \lambda_h(t_{i+1}), \qquad i = 0,\ldots,N-1. \tag{7.13}$$

The adjoint equations have to be solved backwards in time starting at t_N. With this choice the gradient of g_h^a reduces to

$$\nabla g_h^a(\hat p) = \nabla_p \varphi(z_h(t_N;\hat p),\hat p) - S_h(t_0)^\top \lambda_h(t_0) - \sum_{i=0}^{N-1} h_i \frac{\partial \Phi}{\partial p}[t_i]^\top \lambda_h(t_{i+1})$$

width $S_h(t_0) = z_0'(\hat p)$. Since g_h and g_h^a coincide for all discrete trajectories satisfying (7.6)–(7.7), the following theorem holds, see [68, Theorems 5.3.2, 5.3.3] for a proof:

Theorem 7.1 *We have*

$$\nabla g_h(\hat p) = \nabla g_h^a(\hat p) = \nabla_p \varphi(z_h(t_N;\hat p),\hat p) - S_h(t_0)^\top \lambda_h(t_0) - \sum_{i=0}^{N-1} h_i \frac{\partial \Phi}{\partial p}[t_i]^\top \lambda_h(t_{i+1}),$$

where $\lambda_h(\cdot)$ satisfies the adjoint Eqs. (7.12)–(7.13). Moreover, the combined discretization scheme (7.7) and (7.13) for z_h and λ_h is symplectic.

Remark 7.2 Computing the gradient of g_h via the adjoint approach is more efficient than using the sensitivities, because the adjoint equations do not depend on the dimension of p, whereas the sensitivity equations (7.8)–(7.9) are matrix difference equations for the $n \times m$-matrices $S_h(\cdot)$.

7.2 Sensitivity Analysis in Continuous Time

7.2.1 The Forward Mode

The IND approach is based on the differentiation of the discretization scheme. Applying the same idea to the parametric DAE in continuous time (7.1)–(7.2) yields the *sensitivity DAE*

$$\frac{\partial F}{\partial z}[t] \cdot S(t) + \frac{\partial F}{\partial z'}[t] \cdot S'(t) + \frac{\partial F}{\partial p}[t] = 0, \qquad t \in [t_0, t_f], \tag{7.14}$$

$$S(t_0) = z_0'(\hat{p}) \tag{7.15}$$

for the sensitivities $S(t)$ in (7.3). We used the abbreviation $[t] = (t, z(t;\hat{p}), z'(t;\hat{p}), \hat{p})$ and assumed that

$$S'(t) = \frac{\partial^2 z}{\partial p \partial t}(t;\hat{p}).$$

Note that the derivative $z_0'(\hat{p})$ can be obtained by a sensitivity analysis of the least-squares problem LSQ(p) in Sect. 3.2. Moreover, the sensitivity analysis in Theorem 3.1 provides a consistent initial value for the sensitivity DAE (7.14)–(7.15).

Now, the initial value problems for z and S in (7.1)–(7.2) and (7.14)–(7.15) can be solved simultaneously using some suitable one-step or multi-step method. Since efficient implementations often use approximate Jacobians, automatic step-size algorithms, or order selection strategies, the resulting numerical solutions $z_h(\cdot;\hat{p})$ and $S_h(\cdot)$ satisfy $S_h(\cdot) \approx \partial z_h/\partial p(\cdot;\hat{p})$ only up to some tolerance. As a result, the gradient of g in (7.5) will be accurate only in the range of a given integration tolerance. The forward approach using sensitivities is investigated in more detail, e.g., in [29, 37, 79, 87, 101] and a comparison is provided in [52].

A connection to the IND approach arises if the same discretization scheme and the same step-sizes for both DAEs are used. For the BDF method we obtain

$$F\left(t_{i+s}, z_h(t_{i+s}), \frac{1}{h_{i+s-1}} \sum_{\ell=0}^{s} \alpha_\ell z_h(t_{i+\ell}), \hat{p}\right) = 0$$

for $i = 0, \dots, N - s$. Application of the same BDF method with the same step-sizes to the sensitivity DAE (7.14) yields

$$\frac{\partial F}{\partial z}[t_{i+s}] \cdot S_h(t_{i+s}) + \sum_{\ell=0}^{s} \frac{\alpha_\ell}{h_{i+s-1}} \frac{\partial F}{\partial z'}[t_{i+s}] \cdot S_h(t_{i+\ell}) + \frac{\partial F}{\partial p}[t_{i+s}] = 0,$$

for $i = 0, \dots, N-s$. The latter coincides with the IND approach in (7.10). Hence, the discrete and continuous forward modes commute under discretization with the same method and the same step-sizes. The same is true for the Runge–Kutta method (4.7)–(4.10) applied to (7.1), i.e.,

$$z_h(t_{i+1}) = z_h(t_i) + h_i \sum_{j=1}^{s} b_j k_j(t_i, z_h(t_i), h_i, \hat{p}), \tag{7.16}$$

where $k_j(t_i, z_h(t_i), h_i, \hat{p}), j = 1, \dots, s$, are implicitly defined by

$$F\left(t_i + c_\ell h_i, z_h(t_i) + h_i \sum_{j=1}^{s} a_{\ell j}\, k_j, k_\ell, \hat{p}\right) = 0, \qquad \ell = 1, \dots, s. \tag{7.17}$$

Application of the same Runge–Kutta method with the same step-sizes to the sensitivity DAE (7.14) yields

$$S_h(t_{i+1}) = S_h(t_i) + h_i \sum_{j=1}^{s} b_j K_j,$$

where $K_j, j = 1, \dots, s$, are implicitly given by the system of linear equations

$$\frac{\partial F}{\partial z}[t_i + c_\ell h_i]\left(S_h(t_i) + h_i \sum_{j=1}^{s} a_{\ell j}\, K_j\right) + \frac{\partial F}{\partial z'}[t_i + c_\ell h_i] \cdot K_\ell + \frac{\partial F}{\partial p}[t_i + c_\ell h_i] = 0$$

for $\ell = 1, \dots, s$. With

$$K_j = \frac{\partial k_j}{\partial z}[t_i] S_h(t_i) + \frac{\partial k_j}{\partial p}[t_i], \qquad j = 1, \dots, s,$$

the latter coincides with the IND approach for (7.16)–(7.17).

7.2.2 The Backward Mode and Adjoints

Consider the function g in (7.4), i.e., $g(p) = \varphi(z(t_f;p),p)$. Using the sensitivity $S(t_f)$ the gradient is given by

$$\nabla g(\hat{p}) = S(t_f)^\top \nabla_z \varphi(z(t_f;\hat{p}),\hat{p}) + \nabla_p \varphi(z(t_f;\hat{p}),\hat{p}).$$

As in the discrete case we are interested in an alternative representation of the gradient without the sensitivity $S(t_f)$. To this end we define the auxiliary functional

$$g^a(p) := g(p) + \int_{t_0}^{t_f} \lambda(t)^\top F(t,z(t;p),z'(t;p),p)dt,$$

where λ is a suitable function to be defined later. Differentiation with respect to p, evaluation at $\hat{p}$, and integration by parts yield

$$\begin{aligned}
\nabla g^a(\hat{p}) &= S(t_f)^\top \nabla_z \varphi(z(t_f;\hat{p}),\hat{p}) + \nabla_p \varphi(z(t_f;\hat{p}),\hat{p}) \\
&\quad + \int_{t_0}^{t_f} \left(F'_z[t]\cdot S(t) + F'_{z'}[t]\cdot S'(t) + F'_p[t]\right)^\top \lambda(t)dt \\
&= S(t_f)^\top \left(F'_{z'}[t_f]^\top \lambda(t_f) + \nabla_z \varphi(z(t_f;\hat{p}),\hat{p})\right) - S(t_0)^\top F'_{z'}[t_0]^\top \lambda(t_0) \\
&\quad + \nabla_p \varphi(z(t_f;\hat{p}),\hat{p}) + \int_{t_0}^{t_f} F'_p[t]^\top \lambda(t)dt \\
&\quad + \int_{t_0}^{t_f} S(t)^\top \left(F'_z[t]^\top \lambda(t) - \frac{d}{dt}\left(F'_{z'}[t]^\top \lambda(t)\right)\right)dt.
\end{aligned}$$

Since we like to avoid the sensitivities $S(t)$ and $S(t_f)$ we define the *adjoint DAE*

$$F'_{z'}[t_f]^\top \lambda(t_f) + \nabla_z \varphi(z(t_f;\hat{p}),\hat{p}) = 0, \tag{7.18}$$

$$F'_z[t]^\top \lambda(t) - \frac{d}{dt}\left(F'_{z'}[t]^\top \lambda(t)\right) = 0. \tag{7.19}$$

Please note that this derivation is a formal derivation only and it is not clear whether the adjoint DAE (7.18)–(7.19) actually possesses a solution. In fact, it may not have a solution in general. The existence and stability of solutions of the adjoint DAE subject to structural assumptions were investigated in [36]. Details can be found in [68, Sect. 5.3.3] as well.

If the adjoint DAE possesses a solution, then the gradient of g^a is represented by

$$\nabla g^a(\hat{p}) = -S(t_0)^\top F'_{z'}[t_0]^\top \lambda(t_0) + \nabla_p \varphi(z(t_f;\hat{p}),\hat{p}) + \int_{t_0}^{t_f} F'_p[t]^\top \lambda(t)dt$$

with $S(t_0) = z'_0(\hat{p})$ and as in the discrete case it coincides with $\nabla g(\hat{p})$, compare [68, Sect. 5.3.3].

Remark 7.3 Solving the DAE (7.1)–(7.2) and (7.18)–(7.19) simultaneously by some suitable one-step or multi-step method in general does not commute with the discrete adjoint approach.

7.3 Example

Example 7.1 is concerned with a trolley moving on a surface, which leads to an index-three DAE. Herein, a parametric sensitivity analysis is performed and the sensitivity of the states w.r.t. to some parameters is computed using the forward mode.

Example 7.1 Consider the motion of a trolley of mass m_1 on a one-dimensional surface described by the function $h(x)$, which is supposed to be at least twice continuously differentiable, see Fig. 7.

Let a load of mass m_2 be attached to the trolley's center of gravity with a massless rod of length $\ell > 0$. The equations of motion are given by the following index-

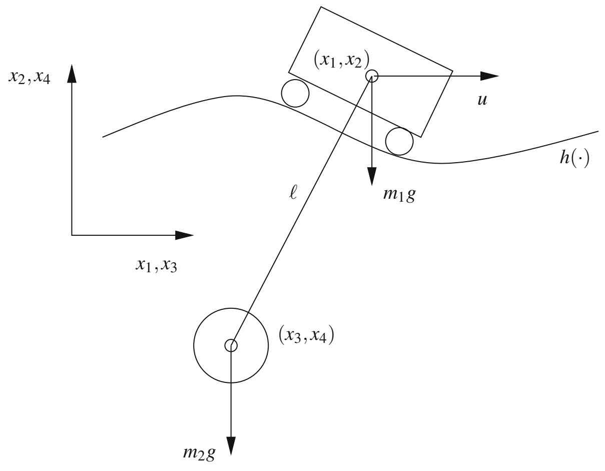

Fig. 7 Configuration of the trolley

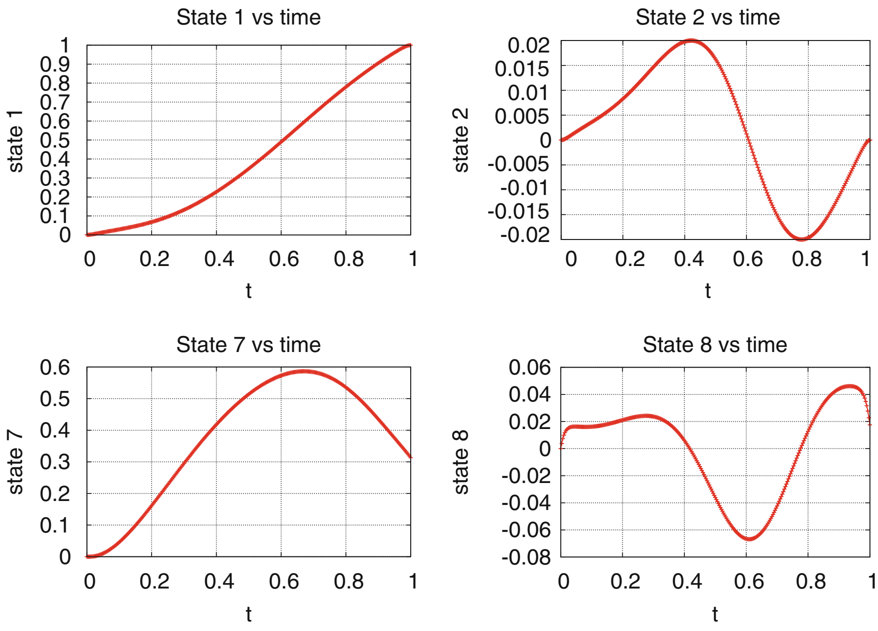

Fig. 8 Positions of trolley (*top*) and velocities of load (*bottom*) (normalized time interval [0, 1])

three DAE:

$$
\begin{aligned}
m_1 x_1''(t) &= u(t) - 2\lambda_1(t)(x_1(t) - x_3(t)) + \lambda_2(t)h'(x_1(t)),\\
m_1 x_2''(t) &= -m_1 g - 2\lambda_1(t)(x_2(t) - x_4(t)) - \lambda_2(t),\\
m_2 x_3''(t) &= 2\lambda_1(t)(x_1(t) - x_3(t)),\\
m_2 x_4''(t) &= -m_2 g + 2\lambda_1(t)(x_2(t) - x_4(t)),\\
0 &= (x_1(t) - x_3(t))^2 + (x_2(t) - x_4(t))^2 - \ell^2,\\
0 &= x_2(t) - h(x_1(t)).
\end{aligned}
$$

Herein, (x_1, x_2) denotes the trolley's center of gravity, (x_3, x_4) the load's position, λ_1, λ_2 the algebraic variables, and $u(t)$ a given control input.

Figures 8 shows the results of a simulation using the software OCPID-DAE1, see http://www.optimal-control.de, on the interval $[0, 2.79]$ (scaled to the normalized interval $[0, 1]$) with $m_1 = 0.3$, $m_2 = 0.5$, $\ell = 0.75$, $g = 9.81$, and $h(x) = 0.02\sin(2\pi x)$. Figure 9 shows the control input u and the algebraic variables λ_1 and λ_2. The computations were performed for the GGL-stabilized system.
Figure 10 shows the sensitivities of some states w.r.t. to m_1.

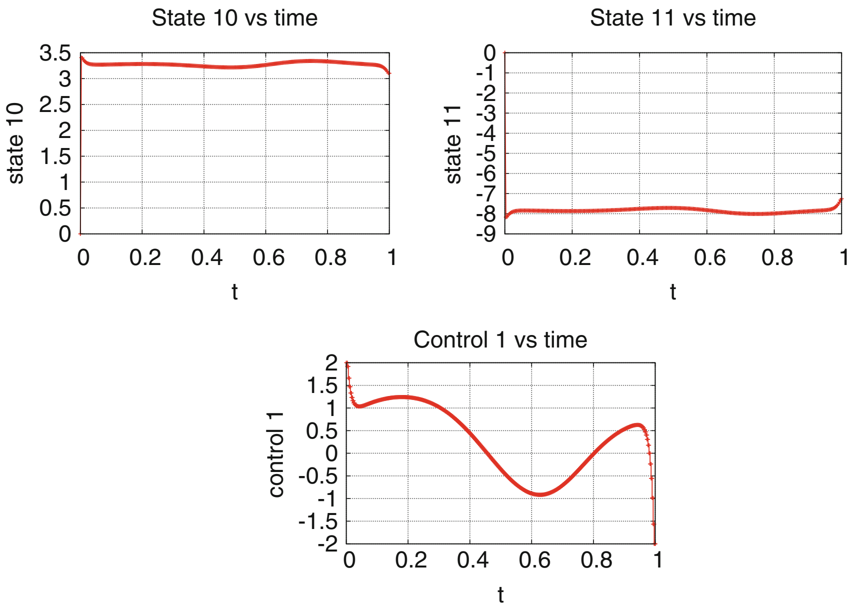

Fig. 9 Algebraic variables (λ_1, λ_2) (*top*) and control input u (*bottom*) (normalized time interval $[0, 1]$)

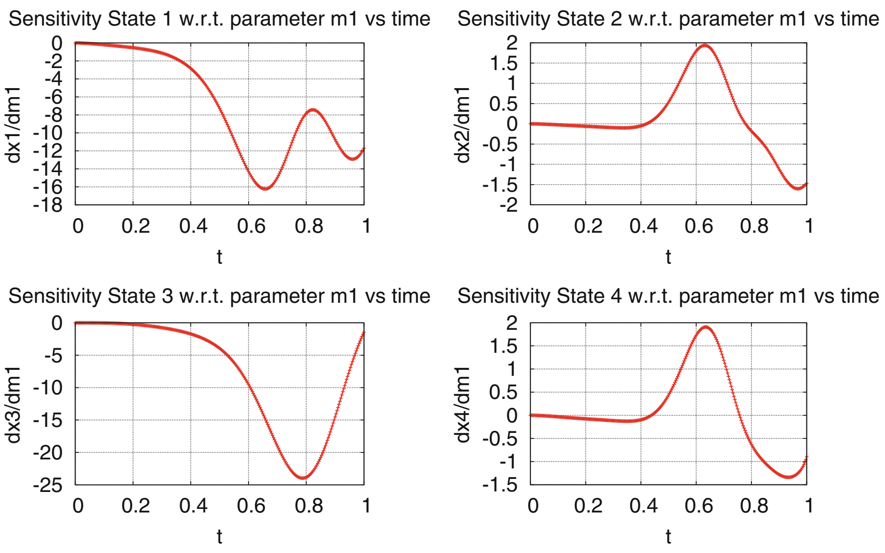

Fig. 10 Sensitivities of positions of trolley (*top*) and load (*bottom*) w.r.t. to m_1 (normalized time interval $[0, 1]$)

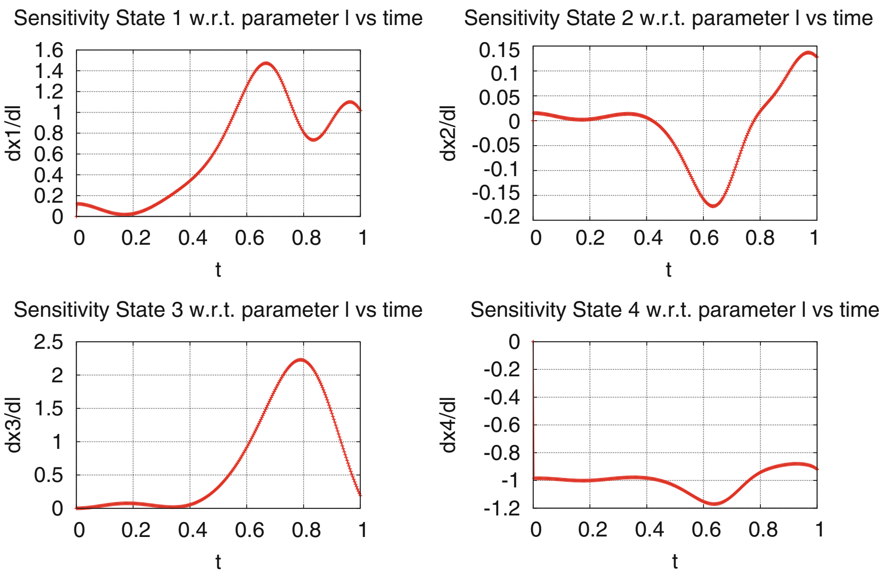

Fig. 11 Sensitivities of positions of trolley (*top*) and load (*bottom*) w.r.t. to ℓ (normalized time interval [0, 1])

Figure 11 shows the sensitivities of some states w.r.t. to ℓ.

8 Switched Systems and Contact Problems

Many applications lead to DAE models with piecewise defined dynamics. Herein, the different DAE models are only valid in defined regions of the state space. Those regions are separated and bounded by manifolds, which are typically implicitly defined by state-dependent switching functions. A transition from one region (i.e., one DAE) to another (with another DAE) occurs, if the switching function changes its sign, i.e., the switching function indicates a switch in the dynamic system. Moreover, a transition from one region to another may come along with a discontinuity of some state components. For instance, contact and friction forces acting between two or more colliding rigid bodies typically lead to discontinuities in the velocity components of the state of a mechanical multibody system.

More general classes of switched DAEs and the existence and stability of solutions are discussed in [97]. The controllability of switched DAEs is investigated in [91]. Hybrid optimal control problems and necessary conditions can be found in [59, 133, 136].

8.1 Hybrid Systems and Switching Functions

It is convenient to view the dynamic process as a hybrid system, compare [114, 142]. To this end, the status of the system is characterized by a finite set of modes $M = \{1, \dots, P\}$. In mode $m \in M$, the state evolves according to the DAE

$$\begin{aligned} x'(t) &= f^m(x(t), y(t)), \\ 0 &= g^m(x(t), y(t)). \end{aligned}$$

The system remains in mode m as long as the trajectory $z(t) = (x(t), y(t))^\top$ stays within the set

$$\mathscr{S}^m := \{x \in X \mid s^m(x) \geq 0\},$$

where for each $m \in M$, $s^m : \mathbb{R}^n \longrightarrow \mathbb{R}$ is called *switching function of mode m*. For simplicity we exclude vector-valued switching functions in order to avoid situations with multiple active switching functions, which are difficult to resolve. $Z = X \times Y \subseteq \mathbb{R}^n \times \mathbb{R}^m$ defines the space of possible differential and algebraic states.

A transition from mode m to another mode $\tilde{m}$ becomes possible only in the event that x is about to cross the boundary of $\mathscr{S}^m$ at some time point $\hat{t}$, i.e., if $s^m(x(\hat{t}^-)) = 0$ and $s^m(x(t)) < 0$ for some $t > \hat{t}$ provided the process would be continued with the dynamics of mode m. Herein, $x(\hat{t}^\pm)$ denote the left- and right-sided limits of x at $\hat{t}$, respectively. The time point $\hat{t}$ in the above situation is called *switching point*.

In case of a transition from mode m to $\tilde{m}$ at time $\hat{t}$, the following jump condition applies to the differential state:

$$x(\hat{t}^+) = x(\hat{t}^-) + d^{m\to\tilde{m}}(x(\hat{t}^-)). \tag{8.1}$$

Herein, $d^{m\to\tilde{m}} : X \longrightarrow X$ denotes the jump function for a transition from mode m to mode $\tilde{m}$. The transition from mode m to mode $\tilde{m}$ is possible only if the state $x(\hat{t}^-)$ belongs to some set $X^{m\to\tilde{m}} \subseteq X$. Moreover, $x(\hat{t}^+)$ is supposed to be consistent with the DAE.

The following assumption provides a sufficient condition for a proper crossing of the switching manifold $\{x \in X \mid s^m(x) = 0\}$ in mode m.

Assumption 8.1 *Let the condition*

$$x'(\hat{t}^-)^\top \nabla s^m(x(\hat{t}^-)) < 0$$

be satisfied whenever the system is in mode $m \in M$ and $\hat{t}$ is a point with $s^m(x(\hat{t}^-)){=}0$.

In the case $x'(\hat{t}^-)^\top \nabla s^m(x(\hat{t}^-)) = 0$, the trajectory is tangential to the manifold $\mathscr{S}^m$ and it may or may not cross the manifold or it may even stay on the manifold. These cases are difficult to handle in general and bifurcation and non-uniqueness issues may occur. Even if Assumption 8.1 holds, infinitely many switches (Zeno phenomenon) may occur with $\lim_{i\to\infty}(\hat{t}_{i+1} - \hat{t}_i) = 0$, where the $\hat{t}_i$'s denote the switching times. The continuation of the trajectory beyond such an accumulation point (the Zeno point) is nontrivial in general. Often, the trajectory is continued such that it stays on the switching manifold.

The simulation of a hybrid system subject to Assumption 8.1 can be performed as follows:

Algorithm 1 (Hybrid System Simulation Using Switching Functions)

(0) Init: Choose a consistent initial value $z_h(t_0) = (x_h(t_0), y_h(t_0))^\top$ at $t = t_0$, an initial mode $m_0 \in M$ with $s^{m_0}(x_h(t_0)) > 0$, a final time $t_f > t_0$, and set $k = 0$.

(1) Stop the integration, if $t_k = t_f$.

(2) Perform one step of a numerical integration scheme with a suitable step-size h to the DAE

$$\begin{aligned} x'(t) &= f^{m_k}(x(t), y(t)), \\ 0 &= g^{m_k}(x(t), y(t)), \end{aligned}$$

and compute the approximation $z_h(t_{k+1}) = (x_h(t_{k+1}), y_h(t_{k+1}))^\top$ at time $t_{k+1} = \min\{t_f, t_k + h\}$.

(3) If $s^{m_k}(x_h(t_{k+1})) > 0$, set $m_{k+1} = m_k$, $k \leftarrow k + 1$, and go to (1). Otherwise go to (4).

(4) If $s^{m_k}(x_h(t_{k+1})) = 0$, find $\tilde{m}$ with $x_h(t_{k+1}) \in X^{m_k \to \tilde{m}}$, update the state by

$$x_h(t_{k+1}) = x_h(t_{k+1}) + d^{m_k \to \tilde{m}}(x_h(t_{k+1})),$$

compute a corresponding consistent initial value $y_h(t_{k+1})$, update the mode by $m_{k+1} = \tilde{m}$, set $k \leftarrow k + 1$, and go to (1). Otherwise go to (5).

(5) If $s^{m_k}(x_h(t_{k+1})) < 0$, determine a step-size $\tau \in [0, 1]$ such that $s^{m_k}(x_h(t_k + \tau h)) = 0$ using the integration scheme in (2), set $t_{k+1} = t_k + \tau h$ and $z_h(t_{k+1}) = (x_h(t_{k+1}), y_h(t_{k+1}))^\top$, and go to (4).

In order to determine the step-size τ in step (5) of Algorithm 1, a root of the function

$$r(\tau) := s^{m_k}(x_h(t_k + \tau h))$$

has to be found. If $r(0) = s^{m_k}(x_h(t_k)) > 0$ and $r(1) = s^{m_k}(x_h(t_k + h)) < 0$, a root can be located by the bisection method with a linear rate of convergence. Such a root τ is guaranteed to exist in $[0, 1]$ by the intermediate value theorem, if the mapping $\zeta : [0, 1] \longrightarrow \mathbb{R}^n$ with $\zeta(\tau) := x_h(t_k + \tau h)$ is continuous. If ζ is continuously differentiable, then we may apply Newton's method in order to find a root of r and hope for an at least super-linear convergence rate. The Newton iteration reads

$$\tau_{\ell+1} = \tau_\ell - \frac{r(\tau_\ell)}{r'(\tau_\ell)}, \qquad \ell = 0, 1, 2, \dots,$$

where

$$r'(\tau) = \zeta'(\tau)^\top \nabla s^{m_k}(\zeta(\tau)).$$

Note that the iteration is well defined, if Assumption 8.1 holds. In fact, the weaker condition $\zeta'(\hat{\tau})^\top \nabla s^{m_k}(\zeta(\hat{\tau})) \neq 0$ in a root $\hat{t}$ of r would be sufficient for a locally super-linear convergence of Newton's method. Often, interpolation techniques are used to compute $\zeta(\tau)$ and $\zeta'(\tau)$ approximately in order to avoid the frequent evaluation of the discretization scheme, see [29, Sect. 5.3.3] for further details.

Example 8.1 Let $x = (x_1, x_2, x_3, x_4)^\top$ and $y = (y_1, y_2)^\top$ be the differential and algebraic states of a pendulum of mass 1 and length 1 with a wall described by the switching function $s^1(x) = x_2 + \frac{1}{2}$ for mode 1 and the set

$$\mathscr{S}^1 = \{(x_1, x_2, x_3, x_4)^\top \mid s^1(x_2) \geq 0\}.$$

In mode 1 (free mode) the pendulum moves according to the DAE (GGL-stabilization)

$$\begin{aligned}
x_1'(t) &= x_3(t) - 2x_1(t)y_2(t),\\
x_2'(t) &= x_4(t) - 2x_2(t)y_2(t),\\
x_3'(t) &= \qquad - 2x_1(t)y_1(t),\\
x_4'(t) &= -g - 2x_2(t)y(t),\\
0 &= x_1(t)^2 + x_2(t)^2 - 1,\\
0 &= x_1(t)x_3(t) + x_2(t)x_4(t).
\end{aligned}$$

If the position $(x_1, x_2)^\top$ hits the boundary of $\mathscr{S}^1$ at some $\hat{t}$, i.e., if $x_2(\hat{t}) = -\frac{1}{2}$, the jump condition

$$\begin{pmatrix} x_3(\hat{t}^+) \\ x_4(\hat{t}^+) \end{pmatrix} = \begin{pmatrix} x_3(\hat{t}^-) \\ x_4(\hat{t}^-) \end{pmatrix} - (1+\varepsilon)\begin{pmatrix} x_3(\hat{t}^-) \\ x_4(\hat{t}^-) \end{pmatrix} = -\varepsilon \begin{pmatrix} x_3(\hat{t}^-) \\ x_4(\hat{t}^-) \end{pmatrix}$$

applies and the mode remains unchanged. Herein, $\varepsilon \in [0, 1]$ is the elasticity constant.

Figure 12 shows the results of a simulation with the code DASRT of the contact problem using switching functions in the time interval $[0, 6]$ with $\varepsilon = 0.9$, initial state $x(0) = (1, 0, 0, 0)^\top$, $y(0) = (0, 0)^\top$, and error tolerance 10^{-10} for the differential states.

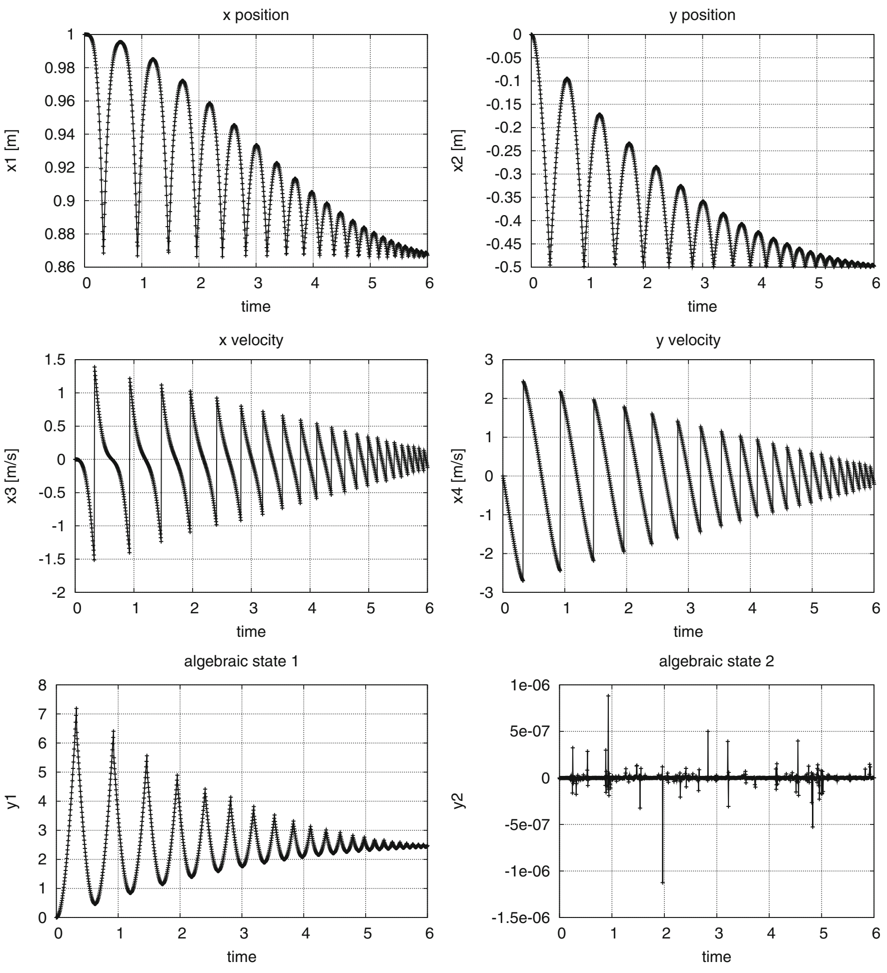

Fig. 12 Numerical simulation of pendulum with contact surface and switching functions

The results illustrate the Zeno phenomenon since the sequence of contact points accumulates. A natural continuation beyond the accumulation point is the constant solution with the pendulum being at rest on the switching manifold.

8.2 Parametric Sensitivity Analysis for Switched Systems

We add a parameter vector $p \in \mathbb{R}^q$ to the problem setting in Sect. 8.1 and investigate the sensitivity of a solution of the hybrid system with respect to the parameter vector.

To this end, let the state evolve according to the parameter-dependent DAE

$$\begin{aligned} x'(t) &= f^m(x(t), y(t), p), \\ 0 &= g^m(x(t), y(t), p) \end{aligned}$$

in mode $m \in M$ within the set

$$\mathscr{S}^m(p) := \{x \in X \mid s^m(x,p) \geq 0\}, \qquad s^m : \mathbb{R}^n \times \mathbb{R}^q \longrightarrow \mathbb{R}.$$

In case of a transition from mode m to $\tilde{m}$ at time $\hat{t}$, the following jump condition applies to the differential state:

$$x(\hat{t}^+) = x(\hat{t}^-) + d^{m \to \tilde{m}}(x(\hat{t}^-), p). \tag{8.2}$$

Herein, $d^{m \to \tilde{m}} : X \times \mathbb{R}^q \longrightarrow X$ denotes the parametric jump function for a transition from mode m to mode $\tilde{m}$, where a transition is possible if $x(\hat{t}^-)$ belongs to some set $X^{m \to \tilde{m}}(p) \subseteq X$. The jump function d in (8.2) has to be chosen such that it provides consistent differential states $x(\hat{t}^+)$.

The functions f^m, g^m, s^m, $m \in M$, and $d^{m \to \tilde{m}}$, $m, \tilde{m} \in M$, are supposed to be at least continuously differentiable with respect to all arguments.

Let $z(t;p) = (x(t;p), y(t;p))^\top$ for $t \in [t_0, t_f]$ denote a solution of the switched system for the parameter p with a consistent initial value $z(t_0;p) = z_0(p) = (x_0(p), y_0(p))^\top$ in mode m with $s^m(x_0(p), p) > 0$. In particular, let $\hat{z}(t) := (\hat{x}(t), \hat{y}(t))^\top$ with $\hat{z}(t) = z(t;\hat{p})$ and $\hat{z}_0 = z_0(\hat{p})$ denote the solution for a fixed nominal parameter $\hat{p} \in \mathbb{R}^q$.

We are interested in computing the sensitivities

$$S_x(t) := \frac{\partial x}{\partial p}(t;\hat{p}), \qquad S_y(t) := \frac{\partial y}{\partial p}(t;\hat{p})$$

assuming their existence in $[t_0, t_f]$.

While in mode m with $s^m(x(t;\hat{p}), \hat{p}) > 0$, the sensitivities solve the sensitivity DAE

$$\begin{aligned} S_x'(t) &= A^m(t) S_x(t) + B^m(t) S_y(t) + c^m(t), \\ 0 &= G^m(t) S_x(t) + H^m(t) S_y(t) + k^m(t), \end{aligned}$$

with

$$\begin{aligned} A^m(t) &:= \frac{\partial f^m}{\partial x}(\hat{x}(t), \hat{y}(t), \hat{p}), & G^m(t) &:= \frac{\partial g^m}{\partial x}(\hat{x}(t), \hat{y}(t), \hat{p}), \\ B^m(t) &:= \frac{\partial f^m}{\partial y}(\hat{x}(t), \hat{y}(t), \hat{p}), & H^m(t) &:= \frac{\partial g^m}{\partial y}(\hat{x}(t), \hat{y}(t), \hat{p}), \\ c^m(t) &:= \frac{\partial f^m}{\partial p}(\hat{x}(t), \hat{y}(t), \hat{p}), & k^m(t) &:= \frac{\partial g^m}{\partial p}(\hat{x}(t), \hat{y}(t), \hat{p}), \end{aligned}$$

compare Sect. 7.

We investigate, what happens at a switching point $\hat{t}$ in mode m with parameter $\hat{p}$. Then, we have

$$s^m(x(\hat{t}^-;\hat{p});\hat{p}) = 0. \tag{8.3}$$

In general, this switching point will depend on the parameter. Define

$$r(t,p) := s^m(x(t^-;p),p)$$

for p close to $\hat{p}$. Equation (8.3) implies

$$\hat{r} := r(\hat{t},\hat{p}) = 0.$$

Assumption 8.2 *The switching point $\hat{t}$ in mode m satisfies*

$$0 \neq \frac{\partial r}{\partial t}(\hat{t},\hat{p}) = \frac{d}{dt}s^m(x(\hat{t}^-;\hat{p});\hat{p}) = x'(\hat{t}^-;\hat{p})^\top \nabla_x s^m(x(\hat{t}^-;\hat{p});\hat{p}).$$

If Assumption 8.2 holds, then, by the implicit function theorem, there exist neighborhoods $B_\delta(\hat{p})$, $\delta > 0$, $B_\varepsilon(\hat{t})$, $\varepsilon > 0$, and a continuously differentiable mapping $T : B_\delta(\hat{p}) \longrightarrow B_\varepsilon(\hat{t})$ with

$$\hat{t} = T(\hat{p}) \qquad \text{and} \qquad r(T(p),p) = 0 \quad \forall p \in B_\delta(\hat{p}),$$

and

$$T'(\hat{p}) = -\left(\frac{\partial r}{\partial t}(\hat{t},\hat{p})\right)^{-1} \frac{\partial r}{\partial p}(\hat{t},\hat{p})$$

with

$$\begin{aligned}
\frac{\partial r}{\partial t}(\hat{t},\hat{p}) &= x'(\hat{t}^-;\hat{p})^\top \nabla_x s^m(x(\hat{t}^-;\hat{p});\hat{p}),\\
\frac{\partial r}{\partial p}(\hat{t},\hat{p}) &= \nabla_x s^m(\hat{x}(\hat{t}^-);\hat{p})^\top S_x(\hat{t}^-) + \nabla_p s^m(\hat{x}(\hat{t}^-);\hat{p}).
\end{aligned}$$

Introducing $T(p)$ into (8.2) yields the relation

$$x(T(p)^+;p) = x(T(p)^-;p) + d^{m\to\tilde{m}}(x(T(p)^-;p),p). \tag{8.4}$$

Herein, we assume that the transition $m \to \tilde{m}$ is stable under small perturbations in p. Differentiation of (8.2) with respect to p and evaluation at $\hat{p}$ yields

$$
\begin{aligned}
x'(\hat{t}^+;\hat{p})T'(\hat{p}) + S_x(\hat{t}^+) = x'(\hat{t}^-;\hat{p})T'(\hat{p}) &+ S_x(\hat{t}^-) \\
&+\nabla_x d^{m\to\tilde{m}}(\hat{x}(t^-),\hat{p})^\top \left(x'(\hat{t}^-;\hat{p})T'(\hat{p}) + S_x(\hat{t}^-)\right) \\
&+\nabla_p d^{m\to\tilde{m}}(\hat{x}(t^-),\hat{p})^\top .
\end{aligned}
$$

Rearranging terms leads to an update rule for the sensitivity S_x at the switching point $\hat{t}$ according to

$$
\begin{aligned}
S_x(\hat{t}^+) = &\left(x'(\hat{t}^-;\hat{p}) - x'(\hat{t}^+;\hat{p}) + \nabla_x d^{m\to\tilde{m}}(\hat{x}(t^-),\hat{p})^\top x'(\hat{t}^-;\hat{p})\right) T'(\hat{p}) \\
&+ \left(I + \nabla_x d^{m\to\tilde{m}}(\hat{x}(t^-),\hat{p})^\top\right) S_x(\hat{t}^-) \\
&+\nabla_p d^{m\to\tilde{m}}(\hat{x}(t^-),\hat{p})^\top .
\end{aligned}
\tag{8.5}
$$

If x is continuous at $\hat{t}$, i.e., $d \equiv 0$, then the update rule for S_x reduces to

$$
S_x(\hat{t}^+) = S_x(\hat{t}^-) + \left(x'(\hat{t}^-;\hat{p}) - x'(\hat{t}^+;\hat{p})\right) T'(\hat{p}).
$$

If, in addition, x' is continuous at $\hat{t}$, then S_x is continuous at $\hat{t}$ as well.

After $x(\hat{t}^+)$ and $S_x(\hat{t}^+)$ have been computed by (8.2) and (8.5), the algebraic component $S_y(\hat{t}^+)$ has to be computed consistently.

Note that this update rule is only valid under Assumption 8.2, which ensures a proper crossing of the switching manifold. If Assumption 8.2 does not hold at $\hat{t}$, it is not clear how to update the sensitivity S_x and in general the state may not depend continuously differentiable on p.

A related parametric sensitivity analysis for mechanical multibody systems using switching functions can be found in [144, Sect. 3.9] and [112]. An adjoint calculus for switched DAEs is derived in [114].

Remark 8.1 Please note that Assumptions 8.1 and 8.2 are crucial in the above analysis. These assumptions are often explicitly or implicitly assumed by standard integrators like DASRT or SCILAB/DASKR. The user needs to be aware of this since codes may fail if the assumptions are not met. As pointed out earlier, it is in general not clear how to continue integration (especially in the context of sensitivity analysis) if the assumptions are not satisfied. In case of the Zeno phenomenon, it is often assumed that the solution stays on the switching manifold. Modifications in the codes are necessary in such cases.

8.3 Contact and Friction in Mechanical Multibody Systems

Mechanical multibody dynamics taking into account contact forces and friction forces are, beyond doubt, the most important examples of switched systems and include particular impact and friction models, compare, e.g., [3, 139]. Suitable discretization schemes for such systems, which typically do not locate impact points exactly but work with a fixed step-size instead, are introduced in [2, 6, 113]. Extensions towards large-scale systems and tailored algorithms for complementarity problems can be found in [5, 137, 138, 140]. Impact models and the interpretation of the mechanical multibody system as a measure differential equation can be found in [65, 88, 102].

The equations of motion of a mechanical multibody system with contact and friction are given by

$$\begin{aligned} q'(t) &= v(t), \\ M(q(t))v'(t) &= f(q(t), v(t)) + F_C(q(t)), \end{aligned}$$

In the above model, $q(t) \in \mathbb{R}^n$ denotes the vector of generalized coordinates, v its velocity, $f(q, v)$ the vector of generalized forces, and $M(q)$ the non-singular mass matrix.

The above multibody system is augmented by an impact model that relates the velocity $v(\hat{t}^-)$ right before an impact to the velocity $v(\hat{t}^+)$ right after the impact. The impact model typically leads to a discontinuity of some components of the velocity vector v at a contact point $\hat{t}$ and hence, the velocity components are only of bounded variation in general. The vector $F_C(q)$ contains the contact and friction forces, which apply only in the case of a contact between the rigid bodies of the multibody system, compare [60, 132]. Whether or not a contact between bodies occurs, is measured by distance functions $s_k : \mathbb{R}^n \longrightarrow \mathbb{R}$ with

$$s_k(q) \geq 0, \qquad k = 1, \ldots, m.$$

Herein, a contact at time $\hat{t}$ occurs, if $s_k(q(\hat{t})) = 0$ for some $k \in \{1, \ldots, m\}$. In case of a contact, the resulting contact and friction force $F_{C,k}(q)$ is an element of the *friction cone*

$$FC_k(q) := \{F^n + F^t \mid F^n = n_k(q)\lambda, F^t = D_k(q)\beta, \lambda \geq 0, \psi(\beta) \leq \mu_k\lambda\}.$$

Herein, $F^n = F_k^n(q)$ denotes the contact force into the normal direction of the contact surface, which can be expressed as $F_k^n(q) = n_k(q)\lambda_k$ with the normal vector $n_k(q) = \nabla s_k(q)$ to the contact manifold $\mathscr{S}_k(q) = \{q \in \mathbb{R}^n \mid s_k(q) = 0\}$. λ_k satisfies the *Signorini contact conditions*

$$0 \leq s_k(q) \qquad \perp \qquad \lambda_k \geq 0,$$

which is a complementarity system for λ_k. The operator $\perp$ in $0 \leq a \perp b \geq 0$ is defined by $a \geq 0$, $b \geq 0$, and $ab = 0$.

The force $F^t = F_k^t(q)$ is the tangential force owing to friction in the contact manifold, which can be expressed as $F_k^t(q) = D_k(q)\beta_k$, where the columns of the matrix $D_k(q)$ span the friction space. For isotropic Coulomb friction, which we assume throughout, the function ψ is given by $\psi(\beta) := \|\beta\|_2$ and $\mu_k \geq 0$ is the friction coefficient at the contact manifold $\mathscr{S}_k(q)$. The norm $\|\cdot\|_2$ causes some difficulties as $\|\beta_k\|_2 \leq \mu_k\lambda_k$ leads to a non-smooth constraint. To overcome this difficulty, the norm $\|\cdot\|_2$ is typically approximated by $\|\cdot\|_1$, which leads to the following polyhedral approximation of the friction cone:

$$\overline{FC}_k(q) := \{F^n + F^t \mid F^n = n_k(q)\lambda, F^t = D_k(q)\beta, \lambda \geq 0, \|\beta\|_1 \leq \mu_k\lambda\},$$

compare [60, 132].

Depending on the choice of the friction cone, the total contact force is then defined by

$$F_C(q) = \sum_{k:s_k(q)=0} F_{C,k}(q)$$

with $F_C(q)$ being an element either of the total friction cone

$$FC(q) = \sum_{k:s_k(q)=0} FC_k(q)$$

or its polyhedral approximation

$$\overline{FC}(q) = \sum_{k:s_k(q)=0} \overline{FC}_k(q).$$

If a contact occurs at time $\hat{t}$, i.e., $s_k(q(\hat{t})) = 0$ for some $k \in \{1, \ldots, m\}$, the *impact model*

$$\nabla s_k(q(\hat{t}))^\top (v(\hat{t}^+) + \varepsilon_k v(\hat{t}^-)) = 0$$

applies. Herein, $\varepsilon_k \in [0, 1]$ denotes the elasticity constant. A fully elastic contact occurs if $\varepsilon_k = 1$. An inelastic contact occurs if $\varepsilon_k = 0$.

An approach to determine the friction force $F_k^t = D_k(q(\hat{t}))\beta_k$ at a contact is based on the *maximum dissipation principle*, compare [132]. Herein, the corresponding friction force F_k^t maximizes the rate of energy dissipation for a given normal force $F_k^n = n_k(q(\hat{t}))\lambda_k$ at a contact. This principle leads to the following convex optimization problem for β:

$$\text{Maximize} \qquad -v(\hat{t}^+)^\top D_k(q(\hat{t}))\beta \qquad \text{s.t.} \qquad \psi(\beta) \le \mu_k\lambda_k. \tag{8.6}$$

Let β_k be a solution of (8.6). If $\lambda_k > 0$, then $\beta = 0$ satisfies the Slater condition for this convex optimization problem, and a necessary and sufficient condition for the solution β_k reads as follows, compare [38, Theorem 6.1.1, Proposition 6.3.1]: There exists a multiplier $\zeta_k \in \mathbb{R}$ such that

$$0 \in D_k(q(\hat{t}))^\top v(\hat{t}^+) + \zeta_k \partial\psi(\beta_k), \tag{8.7}$$

$$0 \le \zeta_k \quad \perp \quad \mu_k\lambda_k - \psi(\beta_k) \ge 0. \tag{8.8}$$

Herein, $\partial\psi = \partial(\|\cdot\|_2)$ denotes the generalized gradient of the locally Lipschitz continuous function ψ, which is given by

$$\partial\psi(\beta) = \begin{cases} \frac{1}{\|\beta\|_2}\beta, & \text{if } \beta \neq 0, \\ \{\alpha \mid \|\alpha\|_2 \le 1\}, & \text{if } \beta = 0. \end{cases}$$

If $\lambda_k = 0$, then $\beta = 0$ is the only feasible point in (8.6) and the conditions (8.7)–(8.8) are satisfied, e.g., by choosing

$$\zeta_k = \|D_k(q(\hat{t}))^\top v(\hat{t}^+)\|_2 \quad \text{and} \quad \alpha = \begin{cases} -\frac{1}{\zeta_k} D_k(q(\hat{t}))^\top v(\hat{t}^+), & \text{if } \zeta_k > 0, \\ 0, & \text{if } \zeta_k = 0. \end{cases}$$

Note that in either case $\alpha \in \partial\psi(0)$. Instead of $\psi(\beta) = \|\beta\|_2$ we may use the approximation $\|\beta\|_1$ in (8.6), which transforms the convex optimization problem in fact into a linear program. To this end, β is replaced by $\beta = \beta^+ - \beta^-$ with $\beta^+, \beta^- \ge 0$:

$$\begin{aligned} &\text{Maximize} \ -v(\hat{t}^+)^\top D_k(q(\hat{t}))(\beta^+ - \beta^-) \\ &\text{s.t.} \qquad e^\top(\beta^+ + \beta^-) \le \mu_k\lambda_k, \ \beta^+ \ge 0, \ \beta^- \ge 0. \end{aligned} \tag{8.9}$$

Herein, we exploited the relation $\|\beta\|_1 = e^\top(\beta^+ + \beta^-)$ with the vector $e = (1, \dots, 1)^\top$ of all ones of appropriate dimension. First order necessary and sufficient conditions for a solution $\beta_k = \beta_k^+ - \beta_k^-$ of (8.9) yield

$$\begin{aligned}
0 &= \left[D_k(q(\hat{t})), -D_k(q(\hat{t}))\right]^\top v(\hat{t}^+) + \zeta_k \begin{pmatrix} e \\ e \end{pmatrix} - \begin{pmatrix} \eta_k^+ \\ \eta_k^- \end{pmatrix}, \\
0 &\le \zeta_k \quad \perp \quad \mu_k \lambda_k - e^\top(\beta_k^+ + \beta_k^-) \ge 0, \\
0 &\le \beta_k^\pm \quad \perp \quad \eta_k^\pm \ge 0.
\end{aligned}$$

Multiplication of the first equation by $(\beta_k^+, \beta_k^-)^\top$ from the left and exploitation of the complementarity conditions yield

$$\begin{aligned}
0 &\le \tilde{\beta}_k \quad \perp \quad \tilde{D}_k(q(\hat{t}))^\top v(\hat{t}^+) + \zeta_k e \ge 0, \\
0 &\le \zeta_k \quad \perp \quad \mu_k \lambda_k - e^\top \tilde{\beta}_k \ge 0
\end{aligned}$$

with

$$\tilde{D}_k(q(\hat{t})) := \left[D_k(q(\hat{t})), -D_k(q(\hat{t}))\right], \quad \tilde{\beta}_k = [\beta_k^+, \beta_k^-]^\top.$$

Note that the matrix $\tilde{D}_k$ is balanced, i.e., if $\tilde{D}_k$ contains a column c, then it contains $-c$ as well. Summarizing, the equations of motion with contact and friction forces satisfy the following complementarity system:

$$\begin{aligned}
q'(t) &= v(t), \\
M(q(t))v'(t) &= f(q(t), v(t)) + \sum_{k=1}^{m} n_k(q(t))\lambda_k(t) + D_k(q(t))\beta_k(t), \\
0 &\le s_k(q(t)) \quad \perp \quad \lambda_k(t) \ge 0, \\
0 &\le \tilde{\beta}_k(t) \quad \perp \quad \tilde{D}_k(q(t))^\top v(t^+) + \zeta_k(t) e \ge 0, \\
0 &\le \zeta_k(t) \quad \perp \quad \mu_k \lambda_k(t) - e^\top \tilde{\beta}_k(t) \ge 0
\end{aligned}$$

and

$$\nabla s_k(q(t))^\top (v(t^+) + \varepsilon_k v(t^-)) = 0 \qquad \text{if } s_k(q(t)) = 0$$

with $k = 1, \ldots, m$. A semi-implicit discretization scheme for the system was suggested in [6, 132]. Let $z^\ell = (q^\ell, v^\ell, \lambda^\ell, \beta^\ell, \zeta^\ell)$ be the state at time t_ℓ and $h > 0$ a step-size. Let the index set of active contacts at t_ℓ be defined by

$$A^\ell := \{k \in \{1, \ldots, m\} \mid s_k(q^\ell + hv^\ell) \le 0\}.$$

Let $\lambda_{A^\ell}^{\ell+1} = (\lambda_k^{\ell+1})_{k \in A^\ell}$ and likewise for $\tilde{\beta}_{A^\ell}^{\ell+1}$ and $\zeta_{A^\ell}^{\ell+1}$.

Then $z^{\ell+1} = (q^{\ell+1}, v^{\ell+1}, \lambda_{A^\ell}^{\ell+1}, \tilde{\beta}_{A^\ell}^{\ell+1}, \zeta_{A^\ell}^{\ell+1})$ solves the following complementarity problem:

$$q^{\ell+1} - q^{\ell} = hv^{\ell+1}, \tag{8.10}$$

$$M(q^{\ell+1})\left(v^{\ell+1} - v^{\ell}\right) = h\left(f(q^\ell, v^\ell) + \sum_{k \in A^\ell} n_k(q^\ell)\lambda_k^{\ell+1} + \tilde{D}_k(q^\ell)\tilde{\beta}_k^{\ell+1}\right), \tag{8.11}$$

$$0 \le \lambda_k^{\ell+1} \quad \perp \quad \nabla s_k(q^\ell)^\top (v^{\ell+1} + \varepsilon_k v^\ell) \ge 0, \quad (k \in A^\ell) \tag{8.12}$$

$$0 \le \tilde{\beta}_k^{\ell+1} \quad \perp \quad \tilde{D}_k(q^\ell)^\top v^{\ell+1} + \zeta_k^{\ell+1} e \ge 0, \quad (k \in A^\ell) \tag{8.13}$$

$$0 \le \zeta_k^{\ell+1} \quad \perp \quad \mu_k \lambda_k^{\ell+1} - e^\top \tilde{\beta}_k^{\ell+1} \ge 0. \quad (k \in A^\ell) \tag{8.14}$$

Convergence results and alternative discretizations are discussed in [4, 6, 60, 113].

It remains to discuss, how the nonlinear complementarity problem can be solved. If M is independent of q or if $M(q^{\ell+1})$ was replaced by $M(q^\ell)$, then the problem is actually a linear complementarity problem, compare [3, 46, 138, 139], and it could be solved by Lemke's algorithm [95] or as in [5, 137]. Another approach is to use a semi-smooth Newton method, compare [86, 115, 116]. Herein, the complementarity system (8.10)–(8.14) is rewritten as the nonlinear and non-smooth equation

$$0 = G(z^{\ell+1}) \tag{8.15}$$
$$= \begin{pmatrix} q^{\ell+1} - q^\ell - hv^{\ell+1} \\ M(q^{\ell+1})\left(v^{\ell+1} - v^\ell\right) - h\left(f(q^\ell, v^\ell) + \sum_{k \in A^\ell} n_k(q^\ell)\lambda_k^{\ell+1} + \tilde{D}_k(q^\ell)\tilde{\beta}_k^{\ell+1}\right) \\ \phi_{FB}(\lambda_k^{\ell+1}, \nabla s_k(q^\ell)^\top (v^{\ell+1} + \varepsilon_k v^\ell)) \quad (k \in A^\ell) \\ \phi_{FB}(\tilde{\beta}_k^{\ell+1}, \tilde{D}_k(q^\ell)^\top v^{\ell+1} + \zeta_k^{\ell+1} e) \quad (k \in A^\ell) \\ \phi_{FB}(\zeta_k^{\ell+1}, \mu_k \lambda_k^{\ell+1} - e^\top \tilde{\beta}_k^{\ell+1}) \quad (k \in A^\ell) \end{pmatrix},$$

where $\phi_{FB}(a, b) := \sqrt{a^2 + b^2} - a - b$ denotes the Lipschitz continuous Fischer-Burmeister function, see [55]. Let $\partial G(z)$ denote Clarke's generalized Jacobian of G, compare [38] and [70] for details on how to compute it. Then, a root of G can be obtained by the following basic version of the semi-smooth Newton method.

Algorithm 2 Semi-Smooth Newton Method

(0) Init: Choose tolerance $tol > 0$ and an initial guess for $z^{\ell+1}$, e.g., $z^{(0)} = (q^\ell + hv^\ell, v^\ell, 0, 0, 0)^\top$. Set $j = 0$.

(1) If $\|G(z^{(j)})\| \le tol$, set $z^{\ell+1} = z^{(j)}$ and STOP.

(2) Compute an element $V^{(j)} \in \partial G(z^{(j)})$ and the Newton direction $d^{(j)}$ by solving the linear equation

$$V^{(j)} d = -G(z^{(j)}).$$

(4) Set $z^{(j+1)} = z^{(j)} + d^{(j)}, j \leftarrow j + 1$, and go to (1).

The following examples summarize results, which have been obtained by applying Algorithm 2 to mechanical multibody systems with contact and friction.

Example 8.2 Consider a bouncing and rotating ball with radius $r = 1$, mass $m = 1$, and moment of inertia $J = 1$ in the $x - z$-plane with $q = (x, z, \theta)^\top$, $M = \mathrm{diag}(m, m, J)$, $f(q, q') = (0, -mg, 0)^\top$, $g = 9.81$, and $s(q) = z - r$. The friction space is spanned by

$$\tilde{D}(q) = \begin{pmatrix} -1 & 1 \\ 0 & 0 \\ r & -r \end{pmatrix}.$$

Figure 13 shows a simulation of the bouncing ball in the time interval [0, 10] with initial state $q(0) = (0, 10, 0)^\top$, $v(0) = (1, 10, -5)^\top$, friction coefficient $\mu = 0.2$, and elasticity parameter $\varepsilon = 0.675$. The states q, v, λ, β, and ζ as functions of time are depicted in Fig. 14.

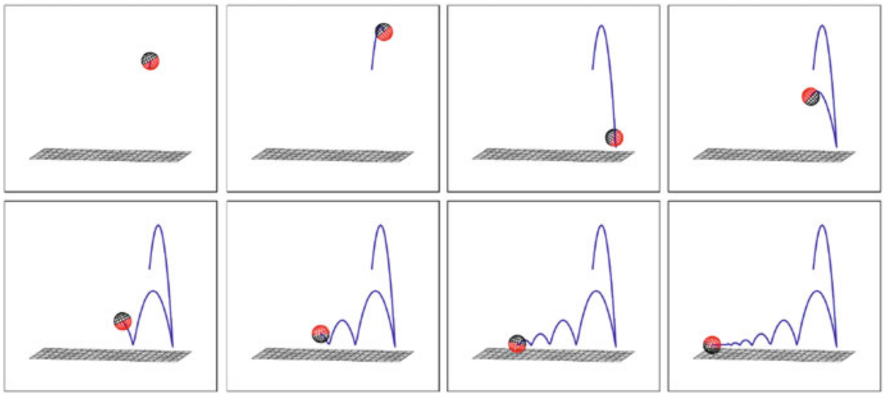

Fig. 13 Snapshot of a bouncing and rotating ball with contact and friction

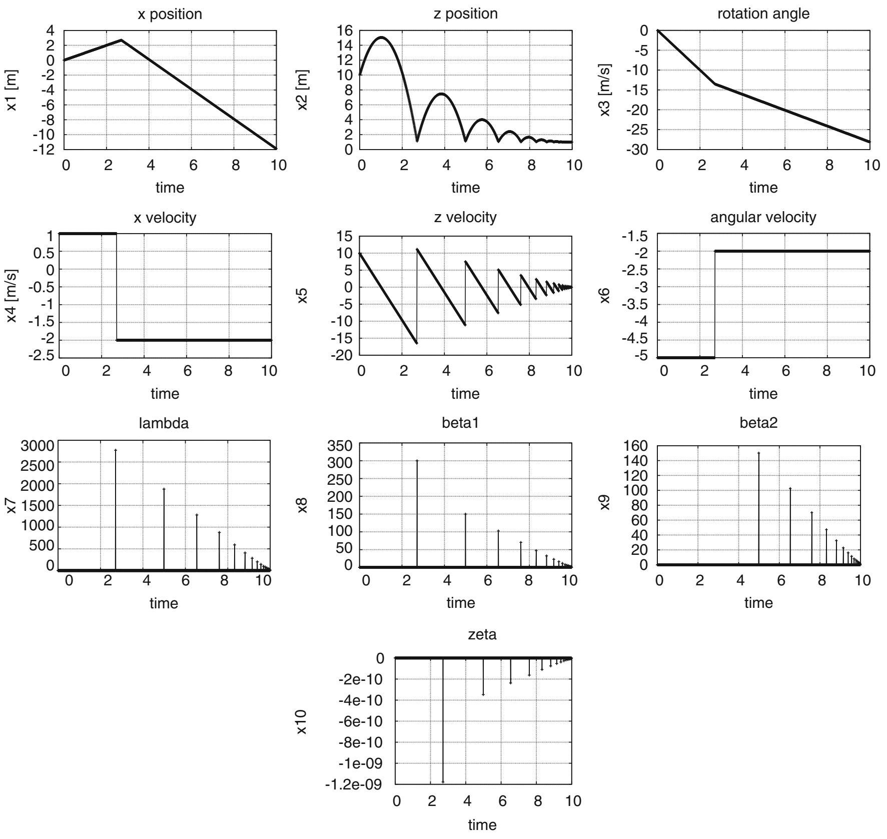

Fig. 14 Snapshot of a bouncing and rotating ball with contact and friction

Example 8.3 Consider a billiard table and the motion of a sphere on the table in the $x - y$-plane. For simplicity, the friction on the table is neglected, but friction forces and contact forces at the boundaries of the table are taken into account with elasticity constant $\varepsilon = 0.9$ and friction coefficient $\mu = 0.5$. The radius of the sphere is $r = 0.04$ [m], its mass is $m = 0.1$ [kg], and its moment of inertia is $J = 1$. The generalized coordinates are $q = (x, y, \theta)^\top$, the mass matrix is $M = \text{diag}(m, m, J)$, the generalized forces are $f(q, q') = (0, 0, 0)^\top$, and the switching function for the

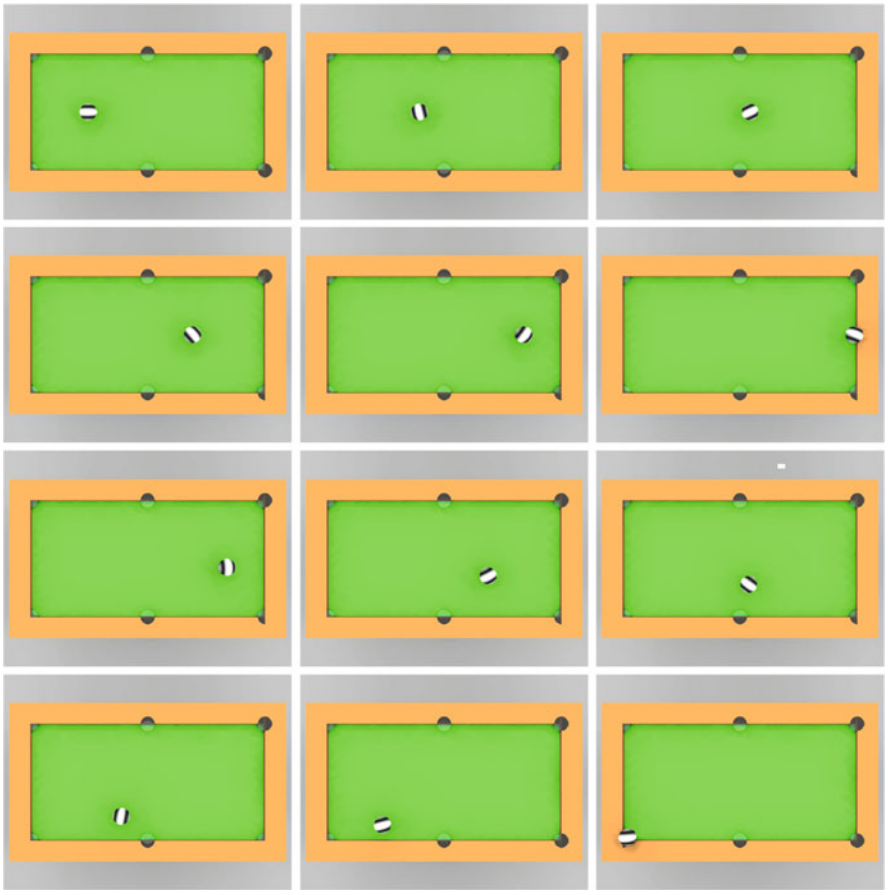

Fig. 15 Snapshot of a billiard table with contact and friction at the borders of the table. For better visibility the sphere was enlarged by a factor of two in the pictures

opposite boundary of the table is $s(q) = y - r$. The friction space is spanned by

$$D(q) = \begin{pmatrix} -1 & 1 \\ 0 & 0 \\ r & -r \end{pmatrix}.$$

Figure 15 shows some snapshots of a simulation of the billiard problem in the time interval $[0, 2.05]$ with initial state $q(0) = (0, 3\ell/4, 0)^\top$, $v(0) = (0, -2, -11)^\top$, where $\ell = 2.24$ denotes the length of the table in [m]. The states q, v, λ, β, and ζ as functions of time are depicted in Fig. 16.

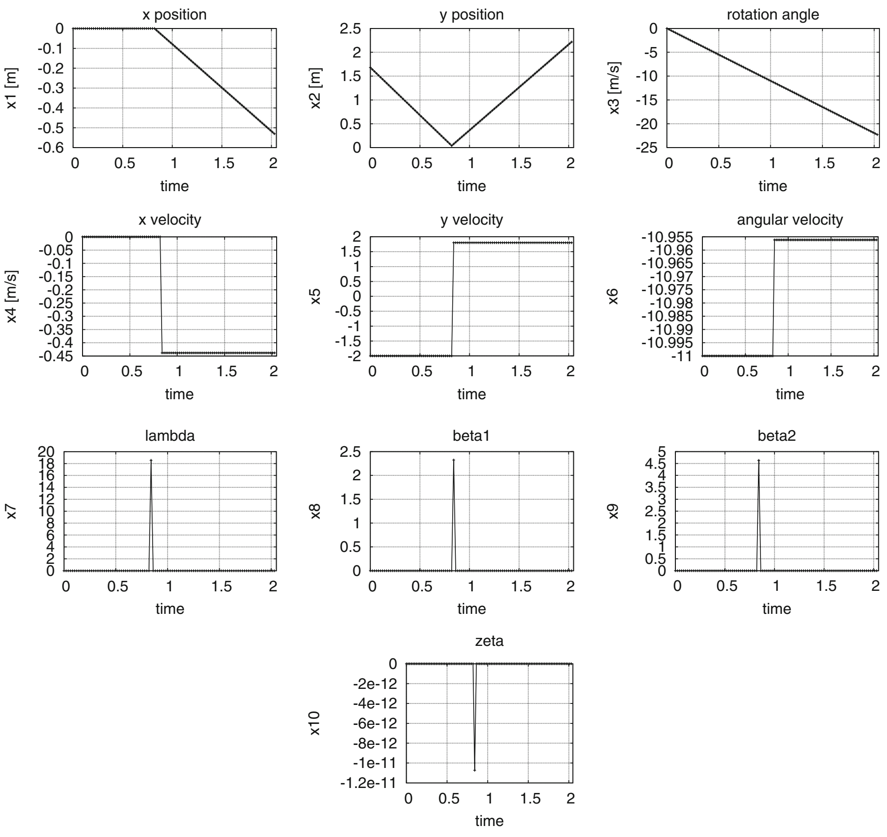

Fig. 16 Snapshot of a billiard table with contact and friction at the borders of the table

9 Conclusions

Simulation is a well-established and indispensable tool in industrial design procedures. Moreover, efficient simulation techniques are required in other disciplines such as controller design, parameter identification, or optimal control. This paper aims to provide an overview on different aspects in the simulation of DAE initial value problems. The focus was set on a choice of methods and concepts that are relevant in industrial simulation environments for coupled systems of potentially large size. These concepts build upon basic integration schemes and add features like sensitivity analysis (needed, e.g., in optimization procedures), contact dynamics, real-time schemes, or co-simulation techniques. Each of these topics is a field of research in its own right with many contributions. Only some of the many contributions could be mentioned in this overview paper and we refer the interested reader to the specialized literature in the bibliography.

References

1. Amodio, P., Mazzia, F.: Numerical solution of differential algebraic equations and computation of consistent initial/boundary conditions. J. Comput. Appl. Math. **87**, 135–146 (1997)
2. Anitescu, M.: Optimization-based simulation of nonsmooth rigid multibody dynamics. Math. Program. **105**(1(A)), 113–143 (2006)
3. Anitescu, M., Potra, F.A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. Nonlinear Dyn. **14**(3), 231–247 (1997)
4. Anitescu, M., Potra, F.A.: A time-stepping method for stiff multibody dynamics with contact and friction. Int. J. Numer. Methods Eng. **55**(7), 753–784 (2002)
5. Anitescu, M., Tasora, A.: An iterative approach for cone complementarity problems for nonsmooth dynamics. Comput. Optim. Appl. **47**(2), 207–235 (2010)
6. Anitescu, M., Potra, F.A., Stewart, D.E.: Time-stepping for three-dimensional rigid body dynamics. Comput. Methods Appl. Mech. Eng. **177**(3–4), 183–197 (1999)
7. Arnold, M.: Half-explicit Runge-Kutta methods with explicit stages for differential-algebraic systems of index 2. BIT **38**(3), 415–438 (1998)
8. Arnold, M.: Multi-rate time integration for large scale multibody system models. In: IUTAM Symposium on Multiscale Problems in Multibody System Contacts: Proceedings of the IUTAM Symposium held in Stuttgart, Germany, February 20–23, 2006, pp. 1–10. Springer, Dordrecht (2007)
9. Arnold, M.: Stability of sequential modular time integration methods for coupled multibody system models. J. Comput. Nonlinear Dyn. **5**, 031003 (2010)
10. Arnold, M.: Modular time integration of block-structured coupled systems without algebraic loops. In: Schöps, S., Bartel, A., Günther, M., ter Maten, E.J.W., Müller, P.C. (eds.) Progress in Differential-Algebraic Equations. Differential-Algebraic Equations Forum, pp. 97–106. Springer, Berlin/Heidelberg (2014)
11. Arnold, M., Günther, M.: Preconditioned dynamic iteration for coupled differential-algebraic systems. BIT Numer. Math. **41**(1), 001–025 (2001)
12. Arnold, M., Murua, A.: Non-stiff integrators for differential-algebraic systems of index 2. Numer. Algorithm. **19**(1–4), 25–41 (1998)
13. Arnold, M., Strehmel, K., Weiner, R.: Half-explicit Runge–Kutta methods for semi-explicit differential-algebraic equations of index 1. Numer. Math. **64**(1), 409–431 (1993)
14. Arnold, M., Burgermeister, B., Eichberger, A.: Linearly implicit time integration methods in real-time applications: DAEs and stiff ODEs. Multibody Syst. Dyn. **17**(2–3), 99–117 (2007)
15. Arnold, M., Burgermeister, B., Führer, C., Hippmann, G., Rill, G.: Numerical methods in vehicle system dynamics: state of the art and current developments. Veh. Syst. Dyn. **49**(7), 1159–1207 (2011)
16. Arnold, M., Clauß, C., Schierz, T.: Error analysis and error estimates for co-simulation in FMI for model exchange and co-simulation v2.0. Arch. Mech. Eng. **LX**, 75–94 (2013)
17. Arnold, M., Hante, S., Köbis, M.A.: Error analysis for co-simulation with force-displacement coupling. Proc. Appl. Math. Mech. **14**(1), 43–44 (2014)
18. Ascher, U.M., Petzold, L.R.: Projected implicit Runge-Kutta methods for differential-algebraic equations. SIAM J. Numer. Anal. **28**(4), 1097–1120 (1991)
19. Balzer, M., Burger, M., Däuwel, T., Ekevid, T., Steidel, S., Weber, D.: Coupling DEM particles to MBS wheel loader via co-simulation. In: Proceedings of the 4th Commercial Vehicle Technology Symposium (CVT 2016), pp. 479–488 (2016)
20. Bartel, A., Brunk, M., Günther, M., Schöps, S.: Dynamic iteration for coupled problems of electric circuits and distributed devices. SIAM J. Sci. Comput. **35**(2), B315–B335 (2013)
21. Bartel, A., Brunk, M., Schöps, S.: On the convergence rate of dynamic iteration for coupled problems with multiple subsystems. J. Comput. Appl. Math. **262**, 14–24 (2014). Selected Papers from NUMDIFF-13
22. Baumgarte, J.: Stabilization of constraints and integrals of motion in dynamical systems. Comput. Methods Appl. Mech. Eng. **1**, 1–16 (1972)

23. Becker, U.: Efficient time integration and nonlinear model reduction for incompressible hyperelastic materials. Ph.D. thesis, TU Kaiserslautern (2012)
24. Becker, U., Simeon, B., Burger, M.: On rosenbrock methods for the time integration of nearly incompressible materials and their usage for nonlinear model reduction. J. Comput. Appl. Math. **262**, 333–345 (2014). Selected Papers from NUMDIFF-13
25. Bock, H.G.: Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen, vol. 183. Bonner Mathematische Schriften, Bonn (1987)
26. Brasey, V., Hairer, E.: Half-explicit RungeKutta methods for differential-algebraic systems of index 2. SIAM J. Numer. Anal. **30**(2), 538–552 (1993)
27. Brenan, K.E., Engquist, B.E.: Backward differentiation approximations of nonlinear differential/algebraic systems. Math. Comput. **51**(184), 659–676 (1988)
28. Brenan, K.E., Petzold, L.R.: The numerical solution of higher index differential/algebraic equations by implicit methods. SIAM J. Numer. Anal. **26**(4), 976–996 (1989)
29. Brenan, K.E., Campbell, S.L., Petzold, L.R.: Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. Classics in Applied Mathematics, vol. 14. SIAM, Philadelphia (1996)
30. Brown, P.N., Hindmarsh, A.C., Petzold, L.R.: Consistent initial condition calculation for differential-algebraic systems. SIAM J. Sci. Comput. **19**(5), 1495–1512 (1998)
31. Burgermeister, B., Arnold, M., Esterl, B.: DAE time integration for real-time applications in multi-body dynamics. Z. Angew. Math. Mech. **86**(10), 759–771 (2006)
32. Burgermeister, B., Arnold, M., Eichberger, A.: Smooth velocity approximation for constrained systems in real-time simulation. Multibody Syst. Dyn. **26**(1), 1–14 (2011)
33. Büskens, C., Gerdts, M.: Differentiability of consistency functions for DAE systems. J. Optim. Theory Appl. **125**(1), 37–61 (2005)
34. Campbell, S.L., Gear, C.W.: The index of general nonlinear DAEs. Numer. Math. **72**, 173–196 (1995)
35. Campbell, S.L., Kelley, C.T., Yeomans, K.D.: Consistent initial conditions for unstructured higher index DAEs: a computational study. In: Computational Engineering in Systems Applications, France, pp. 416–421 (1996)
36. Cao, Y., Li, S., Petzold, L.R., Serban, R.: Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution. SIAM J. Sci. Comput. **24**(3), 1076–1089 (2003)
37. Caracotsios, M., Stewart, W.E.: Sensitivity analysis of initial-boundary-value problems with mixed PDEs and algebraic equations. Comput. Chem. Eng. **19**(9), 1019–1030 (1985)
38. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)
39. Cuadrado, J., Cardenal, J., Bayo, E.: Modeling and solution methods for efficient real-time simulation of multibody dynamics. Multibody Syst. Dyn. **1**(3), 259–280 (1997)
40. Curtiss, C.F., Hirschfelder, J.O.: Integration of stiff equations. Proc. Nat. Acad. Sci. U.S.A. **38**, 235–243 (1952)
41. Deuflhard, P., Hairer, E., Zugck, J.: One-step and extrapolation methods for differential-algebraic systems. Numer. Math. **51**(5), 501–516 (1987)
42. Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., Allgöwer, F.: Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. J. Process Control **12**(4), 577–585 (2002)
43. Diehl, M., Bock, H.G., Schlöder, J.P.: A real-time iteration scheme for nonlinear optimization in optimal feedback control. SIAM J. Control Optim. **43**(5), 1714–1736 (2005)
44. Dopico, D., Lugris, U., Gonzalez, M., Cuadrado, J.: Two implementations of IRK integrators for real-time multibody dynamics. Int. J. Numer. Methods Eng. **65**(12), 2091–2111 (2006)
45. Duff, I.S., Gear, C.W.: Computing the structural index. SIAM J. Algebr. Discrete Methods **7**(4), 594–603 (1986)
46. Ebrahimi, S., Eberhard, P.: A linear complementarity formulation on position level for frictionless impact of planar deformable bodies. Z. Angew. Math. Mech. **86**(10), 807–817 (2006)

47. Eich, E.: Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints. SIAM J. Numer. Anal. **30**(5), 1467–1482 (1993)
48. Eichberger, A., Rulka, W.: Process save reduction by macro joint approach: the key to real time and efficient vehicle simulation. Veh. Syst. Dyn. **41**(5), 401–413 (2004)
49. Engelhardt, L., Burger, M., Bitsch, G.: Real-time simulation of multibody systems for on-board applications. In: Proceedings of the First Joint International Conference on Multibody System Dynamics (IMSD2010) (2010)
50. Esterl, B., Butz, T., Simeon, B., Burgermeister, B.: Real-time capable vehicletrailer coupling by algorithms for differential-algebraic equations. Veh. Syst. Dyn. **45**(9), 819–834 (2007)
51. Estévez Schwarz, D.: Consistent initialization for index-2 differential-algebraic equations and its application to circuit simulation. Ph.D. thesis, Mathematisch-Naturwissenschaftlichen Fakultät II, Humboldt-Universität Berlin (2000)
52. Feehery, W.F., Tolsma, J.E., Barton, P.I.: Efficient sensitivity analysis of large-scale differential-algebraic systems. Appl. Numer. Math. **25**, 41–54 (1997)
53. Feng, A., Holland, C.D., Gallun, S.E.: Development and comparison of a generalized semi-implicit Runge–Kutta method with Gear's method for systems of coupled differential and algebraic equations. Comput. Chem. Eng. **8**(1), 51–59 (1984)
54. Fiacco, A.V.: Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Mathematics in Science and Engineering, vol. 165. Academic Press, New York (1983)
55. Fischer, A.: A special Newton-type optimization method. Optimization **24**, 269–284 (1992)
56. Führer, C.: Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen: Theorie, numerische Ansätze und Anwendungen. Ph.D. thesis, Fakultät für Mathematik und Informatik, Technische Universität München (1988)
57. Führer, C., Leimkuhler, B.J.: Numerical solution of differential-algebraic equations for constraint mechanical motion. Numer. Math. **59**, 55–69 (1991)
58. Gallrein, A., Baecker, M., Burger, M., Gizatullin, A.: An advanced flexible realtime tire model and its integration into Fraunhofer's driving simulator. SAE Technical Paper 2014-01-0861 (2014)
59. Garavello, M., Piccoli, B.: Hybrid necessary principle. SIAM J. Control Optim. **43**(5), 1867–1887 (2005)
60. Gavrea, B.I., Anitescu, M., Potra, F.A.: Convergence of a class of semi-implicit time-stepping schemes for nonsmooth rigid multibody dynamics. SIAM J. Optim. **19**(2), 969–1001 (2008)
61. Gear, C.W.: Simultaneous numerical solution of differential-algebraic equations. IEEE Trans. Circuit Theory **18**(1), 89–95 (1971)
62. Gear, C.W.: Differential-algebraic equation index transformations. SIAM J. Sci. Stat. Comput. **9**, 39–47 (1988)
63. Gear, C.W., Petzold, L.R.: ODE methods for the solution of differential/algebraic systems. SIAM J. Numer. Anal. **21**(4), 716–728 (1984)
64. Gear, C.W., Leimkuhler, B., Gupta, G.K.: Automatic integration of Euler-Lagrange equations with constraints. J. Comput. Appl. Math. **12**(13), 77–90 (1985)
65. Geier, T., Foerg, M., Zander, R., Ulbrich, H., Pfeiffer, F., Brandsma, A., van der Velde, A.: Simulation of a push belt CVT considering uni- and bilateral constraints. Z. Angew. Math. Mech. **86**(10), 795–806 (2006)
66. Gerdts, M.: Optimal control and real-time optimization of mechanical multi-body systems. Z. Angew. Math. Mech. **83**(10), 705–719 (2003)
67. Gerdts, M.: Parameter optimization in mechanical multibody systems and linearized runge-kutta methods. In: Buikis, A., Ciegis, R., Flitt, A.D. (eds.) Progress in Industrial Mathematics at ECMI 2002. Mathematics in Industry, vol. 5, pp. 121–126. Springer, Heidelberg (2004)
68. Gerdts, M.: Optimal Control of ODEs and DAEs. Walter de Gruyter, Berlin/Boston (2012)
69. Gerdts, M., Büskens, C.: Consistent initialization of sensitivity matrices for a class of parametric DAE systems. BIT Numer. Math. **42**(4), 796–813 (2002)
70. Gerdts, M., Kunkel, M.: A nonsmooth Newton's method for discretized optimal control problems with state and control constraints. J. Ind. Manag. Optim. **4**(2), 247–270 (2008)

71. Gopal, V., Biegler, L.T.: A successive linear programming approach for initialization and reinitialization after discontinuities of differential-algebraic equations. SIAM J. Sci. Comput. **20**(2), 447–467 (1998)
72. Griewank, A., Walther, A.: Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation, 2nd edn. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2008)
73. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer Series in Computational Mathematics, vol. 14, 2nd edn. Springer, Berlin/Heidelberg/New York (1996)
74. Hairer, E., Lubich, C., Roche, M.: Error of Rosenbrock methods for stiff problems studied via differential algebraic equations. BIT **29**(1), 77–90 (1989)
75. Hairer, E., Lubich, C., Roche, M.: The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods. Lecture Notes in Mathematics, vol. 1409. Springer, Berlin/Heidelberg/New York (1989)
76. Hairer, E., Norsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I: Nonstiff Problems. Springer Series in Computational Mathematics, vol. 8, 2nd edn. Springer, Berlin/Heidelberg/New York (1993)
77. Hairer, E., Lubich, C., Wanner, G.: Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations. Reprint of the Second 2006 edition. Springer, Berlin (2010)
78. Hansen, B.: Computing consistent initial values for nonlinear index-2 differential-algebraic equations. Seminarberichte Humboldt-Universität Berlin, 92-1, 142–157 (1992)
79. Heim, A.: Parameteridentifizierung in differential-algebraischen Gleichungssystemen. Master's thesis, Mathematisches Institut, Technische Universität München (1992)
80. Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S.: Sundials: suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. **31**(3), 363–396 (2005)
81. INTEC GmbH. SIMPACK – Analysis and Design of General Mechanical Systems. Weßling
82. Jackiewicz, Z., Kwapisz, M.L Convergence of waveform relaxation methods for differential-algebraic systems. SIAM J. Numer. Anal. **33**(6), 2303–2317 (1996)
83. Jackson, K.R.: A survey of parallel numerical methods for initial value problems for ordinary differential equations. IEEE Trans. Magn. **27**(5), 3792–3797 (1991)
84. Jay, L.: Collocation methods for differential-algebraic equations of index 3. Numer. Math. **65**, 407–421 (1993)
85. Jay, L.: Convergence of Runge-Kutta methods for differential-algebraic systems of index 3. Appl. Numer. Math. **17**, 97–118 (1995)
86. Jiang, H.: Global convergence analysis of the generalized Newton and Gauss-Newton methods of the Fischer-Burmeister equation for the complementarity problem. Math. Oper. Res. **24**(3), 529–543 (1999)
87. Kiehl, M.: Sensitivity analysis of ODEs and DAEs - theory and implementation guide. Optim. Methods Softw. **10**(6), 803–821 (1999)
88. Kleinert, J., Simeon, B., Dreßler, K.: Nonsmooth contact dynamics for the large-scale simulation of granular material. Technical report, Fraunhofer ITWM, Kaiserslautern, Germany. J. Comput. Appl. Math. (2015, in press). http://dx.doi.org/10.1016/j.cam.2016.09.037
89. Kübler, R., Schiehlen, W.: Two methods of simulator coupling. Math. Comput. Model. Dyn. Syst. **6**(2), 93–113 (2000)
90. Kunkel, P., Mehrmann, V.: Differential-Algebraic Equations. Analysis and Numerical Solution, vol. viii, 377 p. European Mathematical Society Publishing House, Zürich (2006)
91. Küsters, F., Ruppert, M.G.-M., Trenn, S.: Controllability of switched differential-algebraic equations. Syst. Control Lett. **78**, 32–39 (2015)
92. Lamour, R., März, R., Tischendorf, C.: Differential-Algebraic Equations: A Projector Based Analysis. Differential-Algebraic Equations Forum. Springer, Berlin (2013)

93. Leimkuhler, B., Petzold, L.R., Gear, C.W.: Approximation methods for the consistent initialization of differential-algebraic equations. SIAM J. Numer. Anal. **28**(1), 205–226 (1991)
94. Lelarasmee, E., Ruehli, A.E., Sangiovanni-Vincentelli, A.L.: The waveform relaxation method for time-domain analysis of large scale integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **1**(3), 131–145 (1982)
95. Lemke, C.E.: The dual method of solving the linear programming problem. Naval Res. Log. Q. **1**, 36–47 (1954)
96. Leyendecker, S., Ober-Blöbaum, S.: A variational approach to multirate integration for constrained systems. In: Multibody Dynamics. Computational Methods and Applications. Selected Papers Based on the Presentations at the ECCOMAS Thematic Conference, Brussels, Belgium, 4–7 July 2011, pp. 97–121. Springer, Dordrecht (2013)
97. Liberzon, D., Trenn, S.: Switched nonlinear differential algebraic equations: solution theory, Lyapunov functions, and stability. Automatica **48**(5), 954–963 (2012)
98. Linn, J., Stephan, T., Carlson, J.S., Bohlin, R.: Fast simulation of quasistatic rod deformations for VR applications. In: Bonilla, L.L., Moscoso, M., Platero, G., Vega, J.M. (eds.) Progress in Industrial Mathematics at ECMI 2006. Springer, New York (2007)
99. Lötstedt, P., Petzold, L.R.: Numerical solution of nonlinear differential equations with algebraic constraints I: convergence results for backward differentiation formulas. Math. Comput. **46**, 491–516 (1986)
100. Lubich, C., Engstler, C., Nowak, U., Pöhle, U.: Numerical integration of constrained mechanical systems using MEXX*. Mech. Struct. Mach. **23**(4), 473–495 (1995)
101. Maly, T., Petzold, L.R.: Numerical methods and software for sensitivity analysis of differential-algebraic systems. Appl. Numer. Math. **20**(1), 57–79 (1996)
102. Michael, J., Gerdts, M.: A method to model impulsive multi-body-dynamics using Riemann-Stieltjes- Integrals. In: 8th Vienna International Conference on Mathematical Modelling, International Federation of Automatic Control, pp. 629–634 (2015)
103. Michael, J., Chudej, K., Gerdts, M., Pannek, J.: Optimal rendezvous path planning to an uncontrolled tumbling target. In: IFAC Proceedings Volumes (IFAC-PapersOnline), 19th IFAC Symposium on Automatic Control in Aerospace, ACA 2013, Wurzburg, Germany, 2–6 September 2013, vol. 19, pp. 347–352 (2013)
104. Miekkala, U., Nevanlinna, O.: Convergence of dynamic iteration methods for initial value problems. SIAM J. Sci. Stat. Comput. **8**(4), 459–482 (1987)
105. Murua, A.: Partitioned half-explicit Runge–Kutta methods for differential-algebraic systems of index 2. Computing **59**(1), 43–61 (1997)
106. Negrut, D., Sandu, A., Haug, E.J., Potra, F.A., Sandu, C.: A Rosenbrock-Nystrom state space implicit approach for the dynamic analysis of mechanical systems: II –the method and numerical examples. J. Multi-body Dyn. **217**(4), 273–281 (2003)
107. Ostermann, A.: A class of half-explicit Runge–Kutta methods for differential-algebraic systems of index 3. Appl. Numer. Math. **13**(1), 165–179 (1993)
108. Pantelides, C.C.: The consistent initialization of differential-algebraic systems. SIAM J. Sci. Stat. Comput. **9**(2), 213–231 (1988)
109. Petzold, L.R.: A description of DASSL: a differential/algebraic system solver. Rep. Sand 82-8637, Sandia National Laboratory, Livermore (1982)
110. Petzold, L.R.: Differential/algebraic equations are not ODE's. SIAM J. Sci. Stat. Comput. **3**(3), 367–384 (1982)
111. Petzold, L.R.: Recent developments in the numerical solution of differential/algebraic systems. Comput. Methods Appl. Mech. Eng. **75**, 77–89 (1989)
112. Pfeiffer, A.: Numerische Sensitivitätsanalyse unstetiger multidisziplinärer Modelle mit Anwendungen in der gradientenbasierten Optimierung. Fortschritt-Berichte VDI Reihe 20, Nr. 417. VDI–Verlag, Düsseldorf (2008)
113. Potra, F.A., Anitescu, M., Gavrea, B., Trinkle, J.: A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction. Int. J. Numer. Methods Eng. **66**(7), 1079–1124 (2006)

114. Pytlak, R., Suski, D.: On solving hybrid optimal control problems with higher index DAEs. Institute of Automatic Control and Robotics, Warsaw University of Technology, Warsaw, Poland (2015, Preprint)
115. Qi, L.: Convergence analysis of some algorithms for solving nonsmooth equations. Math. Oper. Res. **18**(1), 227–244 (1993)
116. Qi, L., Sun, J.: A nonsmooth version of Newton's method. Math. Program. **58**(3), 353–367 (1993)
117. Rentrop, P., Roche, M., Steinebach, G.: The application of Rosenbrock-Wanner type methods with stepsize control in differential-algebraic equations. Numer. Math. **55**(5), 545–563 (1989)
118. Rill, G.: A modified implicit Euler algorithm for solving vehicle dynamic equations. Multibody Syst. Dyn. **15**(1), 1–24 (2006)
119. Rill, G., Chucholowski, C.: Real time simulation of large vehicle systems. In: Proceedings of Multibody Dynamics 2007 (ECCOMAS Thematic Conference) (2007)
120. Roche, M.: Rosenbrock methods for differential algebraic equations. Numer. Math. **52**(1), 45–63 (1988)
121. Rosenbrock, H.H.: Some general implicit processes for the numerical solution of differential equations. Comput. J. **5**(4), 329–330 (1963)
122. Rulka, W., Pankiewicz, E.: MBS approach to generate equations of motions for hil-simulations in vehicle dynamics. Multibody Syst. Dyn. **14**(3), 367–386 (2005)
123. Sandu, A., Negrut, D., Haug, E.J., Potra, F.A., Sandu, C.: A Rosenbrock-Nystrom state space implicit approach for the dynamic analysis of mechanical systems: I—theoretical formulation. J. Multi-body Dyn. **217**(4), 263–271 (2003)
124. Schaub, M., Simeon, B.: Blended Lobatto methods in multibody dynamics. Z. Angew. Math. Mech. **83**(10), 720–728 (2003)
125. Schierz, T., Arnold, M.: Stabilized overlapping modular time integration of coupled differential-algebraic equations. Appl. Numer. Math. **62**(10), 1491–1502 (2012). Selected Papers from NUMDIFF-12
126. Schneider, F., Burger, M., Arnold, M., Simeon, B.: A new approach for force-displacement co-simulation using kinematic coupling constraints. Submitted to Z. Angew. Math. Mech. (2016)
127. Schulz, V.H., Bock, H.G., Steinbach, M.C.: Exploiting invariants in the numerical solution of multipoint boundary value problems for DAE. SIAM J. Sci. Comput. **19**(2), 440–467 (1998)
128. Schwartz, W., Frik, S., Leister, G.: Simulation of the IAVSD Road Vehicle Benchmark Bombardier Iltis with FASIM, MEDYNA, NEWEUL and SIMPACK. Technical Report IB 515/92-20, Robotik und Systemdynamik, Deutsche Forschungsanstalt für Luft- und Raumfahrt (1992)
129. Schweizer, B., Lu, D.: Stabilized index-2 co-simulation approach for solver coupling with algebraic constraints. Multibody Syst. Dyn. **34**(2), 129–161 (2014)
130. Schweizer, B., Li, P., Lu, D.: Implicit co-simulation methods: stability and convergence analysis for solver coupling approaches with algebraic constraints. Z. Angew. Math. Mech. **96**(8), 986–1012 (2016)
131. Stetter, H.J.: Analysis of Discretization Methods for Ordinary Differential Equations. Springer Tracts in Natural Philosophy, vol. 23. Springer, Berlin/Heidelberg/New York (1973)
132. Stewart, D.E.: Rigid-body dynamics with friction and impact. SIAM Rev. **42**(1), 3–39 (2000)
133. Stewart, D.E., Anitescu, M.: Optimal control of systems with discontinuous differential equations. Numer. Math. **114**(4), 653–695 (2010)
134. Strehmel, K., Weiner, R.: Numerik gewöhnlicher Differentialgleichungen. Teubner, Stuttgart (1995)
135. Strehmel, K., Weiner, R., Dannehl, I.: On error behaviour of partitioned linearly implicit Runge–Kutta methods for stiff and differential algebraic systems. BIT **30**(2), 358–375 (1990)
136. Sussmann, H.J.: A nonsmooth hybrid maximum principle. In: Stability and Stabilization of Nonlinear Systems. Proceedings of the 1st Workshop on Nonlinear Control Network, Held in Gent, Belgium, 15–16 March 1999, pp. 325–354. Springer, London (1999)

137. Tasora, A., Anitescu, M.: A fast NCP solver for large rigid-body problems with contacts, friction, and joints. In: Multibody Dynamics. Computational Methods and Applications. Revised, extended and selected papers of the ECCOMAS Thematic Conference on Multibody Dynamics 2007, Milano, Italy, 25–28 June 2007, pp. 45–55. Springer, Dordrecht (2009)
138. Tasora, A., Anitescu, M.: A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics. Comput. Methods Appl. Mech. Eng. **200**(5–8), 439–453 (2011)
139. Tasora, A., Anitescu, M.: A complementarity-based rolling friction model for rigid contacts. Meccanica **48**(7), 1643–1659 (2013)
140. Tasora, A., Negrut, D., Anitescu, M.: GPU-based parallel computing for the simulation of complex multibody systems with unilateral and bilateral constraints: an overview. In: Multibody Dynamics. Computational Methods and Applications. Selected papers based on the presentations at the ECCOMAS Conference on Multibody Dynamics, Warsaw, Poland, June 29–July 2, 2009, pp. 283–307. Springer, New York, NY (2011)
141. Trenn, S.: Solution concepts for linear DAEs: a survey. In: Ilchmann, A., Reis, T. (eds.) Surveys in Differential-Algebraic Equations I. Differential-Algebraic Equations Forum, pp. 137–172. Springer, Berlin (2013)
142. van der Schaft, A., Schumacher, H.: An Introduction to Hybrid Dynamical Systems. Springer, London (1989)
143. Veitl, A., Gordon, T., van de Sand, A., Howell, M., Valasek, M., Vaculin, O., Steinbauer, P.: Methodologies for coupling simulation models and codes in mechatronic system analysis and design. In: Proceedings of the 16th IAVSD Symposium on Dynamics of Vehicles on Roads and Tracks. Pretoria. Supplement to Vehicle System Dynamics, vol. 33, pp. 231–243. Swets & Zeitlinger (1999)
144. von Schwerin, R.: Multibody System Simulation: Numerical Methods, Algorithms, and Software. Lecture Notes in Computational Science and Engineering, vol. 7. Springer, Berlin/Heidelberg/New York (1999)
145. Wensch, J.: An eight stage fourth order partitioned Rosenbrock method for multibody systems in index-3 formulation. Appl. Numer. Math. **27**(2), 171–183 (1998)
146. Wensch, J., Strehmel, K., Weiner, R.: A class of linearly-implicit Runge–Kutta methods for multibody systems. Appl. Numer. Math. **22**(13), 381–398 (1996). Special Issue Celebrating the Centenary of Runge–Kutta Methods
147. Wolfbrandt, A., Steihaug, T.: An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff differential equations. Math. Comput. **33**(146), 521–534 (1979)