Technische Universität Berlin

Fakultät für Mathematik und Naturwissenschaften

Institut für Mathematik

Numerical Solution of Second Order Differential-Algebraic Equations

Diplomarbeit

von

Lena Wunderlich

bei

Prof. Dr. V. Mehrmann

im Studiengang Techno- und Wirtschaftsmathematik

Berlin, den 20. Juni 2005.

Contents

List of Figures

List of Tables

Die selbstständige und eigenhändige Anfertigung versichere ich an Eides statt.

Berlin, den 20. Juni 2005

Lena Wunderlich

Introduction

This work deals with the numerical solution of second order differential-algebraic equations (DAEs). The purpose is to examine different approaches to solve second order differential-algebraic systems with higher index numerically and to compare this approach with the solutions of corresponding first order systems. In particular, we consider modified backward differentiation formulas (BDF) and Runge-Kutta methods for the numerical solution of second order systems. Second order differential-algebraic equations arise naturally in technical applications and in the modeling and simulation of constrained dynamical systems. One important example is the simulation of multibody systems, where we usually have second order equations of motion to describe the dynamics of the system under certain constraints. These equations of motion are of differentiation index 3 and therefore belong to the class of higher index systems. In this work we are particularly interested in the numerical solution of this kind of systems. Other applications where higher order DAEs frequently arise are models of electrical circuits.
Why is there a need of solving second order systems directly? Usually, in the classical theory of ordinary differential equations, second order systems are solved by introducing new variables for the derivatives to transform the systems into first order systems and then solve the first order systems numerically. This classical transformation to first order systems may lead to certain difficulties when applied to DAEs. When we analyze the procedure of turning a differential-algebraic system into a first order system this approach may lead to solutions that have different smoothness requirements and the resulting first order system can have a higher index than the original system, see [24]. As demonstrated in [24] the transformation to first order can even be mathematically incorrect, i.e., the resulting first order system may not have a solution, while the original system is solvable. All this leads to instabilities arising in numerical solution methods. Another difficulty that arises in practice, is that the systems may be badly scaled and that there are disturbances and perturbations in the data, such that the transformation to first order leads to very different solutions in the perturbed systems, see [15]. To avoid these difficulties, we are interested in methods to directly solve the second order differential-algebraic systems.
Even if the transformation to first order systems does not increase the index the DAE may be of higher index itself, such as the equations of motion of multibody systems. A higher index leads to difficulties in the numerical solution as methods for the solution of first order DAEs may not converge for higher index problems. In the industrial simulation the code DASSL, which is based on BDF methods is often used to solve DAEs. The main problem when using

BDF methods is that these methods suffer a loss of accuracy whenever there is a change of stepsize or change of order in the methods. The first order BDF method (implicit Euler method) even fails to converge in the first steps after a change of stepsize for higher index problems. This behavior of BDF methods is responsible for instabilities arising in the numerical solutions and for a reduction of the convergence order for higher index DAEs.
Thus, there is a need of index reduction for higher index equations. By index reduction the index of the DAE is decreased using certain techniques and analytical equivalence transformations. Here, we only consider index reduction techniques for the equations of motion of multibody systems. One disadvantage of index reduction is that the dimension of the system is increased, as index reduction is usually done by adding extra equations (the derivatives of the constraints) to the equations of motion. Moreover, index reduction is an additional step in the integration procedure and we require extra assumptions and additional information about the structure of the equations of motion.
The advantages of solving second order systems directly are clear now. First of all, we do not need to transform to first order systems and thus we may avoid to increase the index of the DAEs unintentionally or impose stronger smoothness requirements on the systems as described above. In addition we can avoid to increase (almost double) the dimension of the systems by introducing new variables. Particularly when simulating complex mechanical systems in aircrafts or vehicle systems we have already large systems in the second order formulation. Further, we can avoid extra index reductions and instabilities in the numerical solutions, which arise when solving first order higher index DAEs. Thus, we can save additional steps in the integration procedure and have no increase in the dimension of the system or in the index, which is costly in computational time and effort as well as memory space.

The main focus of this work lies in the examination of modified BDF methods for the numerical solution of second order DAEs, especially of the second order equations of motion of multibody systems. We have implemented variable-step variable-order modified BDF methods up to order 2 for the numerical solution of the second order equations of motion, which are of differentiation index 3.
We have also developed a Runge-Kutta method based on collocation for second order systems and have implemented a 2-stage Runge-Kutta method for the solution of second order equations of motion.

This thesis is organized as follows. First, we give an introduction to multibody systems and show how to derive the equations of motion in Chapter 1. The equations of motion of multibody systems form a second order system of differential equations arising from the dynamics of the mechanical system together with algebraic equations, which represent different constraints.
In Chapter 2 we give some background information on first order differential-algebraic equations and semi-explicit systems and introduce the concept of BDF methods. We give some convergence results for the numerical solution of first order DAEs by BDF methods and show some problems which arise in the case of higher index.
Next, we consider second order DAEs of higher index and after some preliminary definitions we explain some problems arising in the order reduction of higher order systems in Chapter 3. We introduce modified BDF methods for

the numerical solution of second order higher index DAEs.
In Chapter 4 we describe in detail the implementation of the modified BDF methods applied to the second order equations of motion of multibody systems. We choose a variable-step variable-order implementation similar to the integrator DASSL ([4]).
After this, we give an overview of the numerical solution of first order higher index DAEs by Runge-Kutta methods in Chapter 5. Here, we restrict to semi-explicit systems of d-index 1,2 and 3 and give some convergence results.
Chapter 6 deals with Runge-Kutta methods for second order DAEs. We introduce the concepts of collocation and develop a 2-stage Runge-Kutta method of order 4 for the solution of second order systems based on collocation. Further, we describe the application of this method to multibody systems.
In Chapter 7 we describe certain index reduction techniques for the equations of motion of multibody systems in the first and second order form. Index reduction is a common method used to decrease the index of a DAE in order to obtain better numerical results. We consider the GGL-formulation and a strangeness-free formulation based on minimal extension, both in the first and second order form.
Finally, we examine the performance of the discussed methods with some numerical examples in Chapter 8. We compare the results obtained by the modified BDF methods with the results obtained by solving some test problems in the first order form with the integrator DASSL. Here, we use the index reduction techniques to obtain formulations of different index. In addition we give some results for the 2-stage Runge-Kutta method.

Chapter 1

The Equations of Motion of Multibody Systems

In the industrial simulation of mechanical systems the multibody approach is frequently used. When simulating a mechanical system, the system has to be transformed into a computer model. We model the mechanical system as a so-called *multibody system* (MBS). A multibody system is the result of describing the mechanical system by a finite number of simple components. These components consists of bodies, which have masses and torques and can either be rigid or elastic. They are linked together by massless interconnections, which can be force elements, like springs and dampers, or active control components, e.g., pneumatic or hydraulic components and mechatronic elements, or joints reducing the degrees of freedom.
The most important factor for the dynamical simulation of multibody systems is to know the structure and form of the equations of motion. In this chapter we will introduce a method for deriving the equations of motion of constrained mechanical systems by the Lagrange-Hamilton principle [16, 26] using the Euler-Lagrange equations. In the second part we derive the equations of motion for the most frequent classes of constrained multibody systems.
We will see later, that the equations of motion of constrained multibody systems form an important subclass of second order differential-algebraic equations with higher index.

1.1 The Euler-Lagrange Equations

To obtain the equations of motion for multibody systems, one follows the principles of classical mechanics, see [2, 15, 16, 26]. Let $q_1, \dots, q_{n_q}$ be the generalized position coordinates of a conservative mechanical system with $q = [q_1, \dots, q_{n_q}]^T$. A conservative system is a system in which work done by a force has the following properties. First, the work is independent of path, i.e., the work done by a force to move an object between any two points is independent of the path taken. Second, the work is equal to the difference between the final and initial values of an energy function and third, it is completely reversible, meaning that the work done by a conservative force is recoverable.

The term generalized coordinates means that the state of the system is determined uniquely by $q_1, \dots, q_{n_q}$ and that the q_i are independent of each other for $i = 1, \dots, n_q$. Here, n_q is the number of degrees of freedom of the system. With the potential energy $U(q)$ and the kinetic energy $T(q, \dot{q})$, one defines the Lagrange function

$$L(q, \dot{q}) := T(q, \dot{q}) - U(q). \tag{1.1}$$

Here, and in the following, we will use the dot operator as the derivative with respect to time t, i.e., $\dot{q} := \frac{d}{dt}q$, and we will call $\dot{q}_1, \dots, \dot{q}_{n_q}$ the velocity coordinates of the mechanical system. To derive the equations of motion, the Lagrange-Hamilton principle [16] is used. It states that the integral $\int_{t_0}^{t_1} L(q, \dot{q})dt$ has to be minimized for the Lagrange function L. For this variational problem we obtain the Euler-Lagrange equations as

$$\frac{d}{dt}\left(\frac{\partial L(\dot{q}, q)}{\partial \dot{q}}\right) - \frac{\partial L(\dot{q}, q)}{\partial q} = 0. \tag{1.2}$$

This system of n_q nonlinear equations form the *equations of motion* for a conservative system.
If the motion of the system is restricted by n_λ ideal constraints $g_1(q) = 0, \dots,$ $g_{n_\lambda}(q) = 0$, then the Lagrange function has a different form, see [26]. The term ideal constraints defines constraints which do not involve exchange of energy between the system and its environment. In this case (1.1) reads

$$L(q, \dot{q}, \lambda) = T(q, \dot{q}) - U(q) - \sum_{k=1}^{n_\lambda} \lambda_k g_k(q), \tag{1.3}$$

with the Lagrange multipliers $\lambda = [\lambda_1, \dots, \lambda_{n_\lambda}]^T$ and the Euler-Lagrange equations (1.2) become

$$\frac{d}{dt}\left(\frac{\partial L(\dot{w}, w)}{\partial \dot{w}}\right) - \frac{\partial L(\dot{w}, w)}{\partial w} = 0, \tag{1.4}$$

with $w = [q, \lambda]^T$.
In certain mechanical systems one distinguishes different types of constraints. If we consider n_s additional constraints $h_1(q, s) = 0, \dots, h_{n_s}(q, s) = 0$ with additional coordinates $s \in \mathbb{R}^{n_s}$, e.g., the contact point coordinates in contact problems, then we get the Lagrange function

$$L(q, \dot{q}, \lambda, \nu) = T(q, \dot{q}) - U(q) - \sum_{k=1}^{n_\lambda} \lambda_k g_k(q, s) - \sum_{k=1}^{n_s} \nu_k h_k(q, s), \tag{1.5}$$

where $\lambda = [\lambda_1, \dots, \lambda_{n_\lambda}]^T$ and $\nu = [\nu_1, \dots, \nu_{n_s}]^T$ are the Lagrange multipliers. Thus, we have the Euler-Lagrange equations (1.4) with $w = [q, \lambda, \nu]^T$.
By the Euler-Lagrange principle also non-conservative forces can be considered. A non-conservative force F_{nc} can be included through the principle of virtual work, see [2, 16]. The non-conservative force F_{nc} is given by

$$F_{nc} = \frac{d}{d(\delta q)}(\delta W_{nc}), \tag{1.6}$$

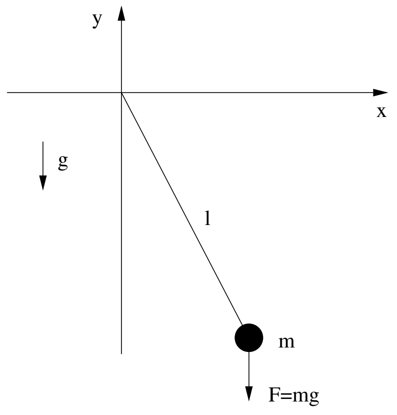

Figure 1.1: Ideal pendulum

where the virtual work δW_{nc} of the applied forces results from the applied forces F_i and their corresponding virtual displacements δq_i for $i = 1, \dots, n_q$ and is given by

$$\delta W_{nc} = \sum_{i=1}^{n_q} F_i \cdot \delta q_i. \tag{1.7}$$

The Hamilton principle for non-conservative systems has the general form

$$\int_{t_0}^{t_1} (\delta W_{nc} + \delta T) dt = 0. \tag{1.8}$$

In the special case of conservative forces we have $\delta W = -\delta U$, so that (1.8) equals $\int_{t_0}^{t_1} L(q, \dot{q}) dt$. For non-conservative systems we get the Euler-Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial L(\dot{w}, w)}{\partial \dot{w}}\right) - \frac{\partial L(\dot{w}, w)}{\partial w} = F_{nc}. \tag{1.9}$$

1.2 The Equations of Motion

Using the Euler-Lagrange equations derived in Section 1.1, the equations of motion for different mechanical multibody systems can be developed, see [8]. At first, we give a simple example to show how to derive the equations of motion.

Example 1.2.1. *The pendulum*
Consider an ideal pendulum of length l and mass m with the kinetic energy $T = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2)$, the potential energy $U = mgy$ and the geometric constraint $x^2 + y^2 - l^2 = 0$, see Figure 1.1. The Lagrange function is given by
$L = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) - mgy - \lambda(x^2 + y^2 - l^2)$, with the Lagrange parameter λ. In this case the Euler-Lagrange equations for $w = [x, y, \lambda]^T$ are given by

$$\begin{aligned} m\ddot{x} + 2x\lambda &= 0, \\ m\ddot{y} + 2y\lambda + mg &= 0, \\ x^2 + y^2 - l^2 &= 0. \end{aligned} \tag{1.10}$$

System (1.10) *forms the equations of motion of the pendulum.*

In the following we consider various classes of multibody systems having differently structured equations of motion.

1.2.1 Unconstrained Planar Multibody Systems

We start with the simple case of unconstrained mechanical multibody systems. The equations of motion for such systems can be written as

$$M(q)\ddot{q} = f_a(q, \dot{q}, t), \tag{1.11}$$

with the position coordinates $q \in \mathbb{R}^{n_q}$ and the positive definite mass matrix $M = T_{\dot{q}\dot{q}} := \frac{\partial^2 T}{\partial \dot{q}^2} \in \mathbb{R}^{n_q, n_q}$. The vector $f_a(q, \dot{q}, t) \in \mathbb{R}^{n_q}$ contains the applied forces, e.g., gravity and Coriolis force.

1.2.2 Constrained Planar Multibody Systems

In constrained mechanical systems the free motion of the systems is restricted. The constraints result either from direct physical contact of two bodies of the system or from an interconnection of the bodies, e.g., joints. One distinguishes between holonomic and non-holonomic constraints.
Constraints of the form $g_i(q, t) = 0$ for $i = 1, \ldots, n_\lambda$, where the constraints depend on the position coordinates and n_λ is the number of constraints, are called *holonomic*. They can be divided into *rheonomic*, i.e., time dependent constraints and *scleronomic*, i.e., constraints independent of time. *Nonholonomic* constraints are of the form $g_i(q, \dot{q}, t) = 0$. They depend in addition on the velocity coordinates. They are also divided into rheonomic and scleronomic constraints. An example for a nonholonomic system is a disc rolling in a plane without slipping ("pure" rolling), see [26]. In constrained mechanical systems a special class of force, the so called *constraint force* must be taken into account.

Holonomic Constraints

If a number of holonomic constraints is considered, then the equations of motion have the following form

$$\begin{aligned} M(q)\ddot{q} =& f_a(q, \dot{q}, t) - f_c(q, \lambda, t), \\ 0 =& g(q, t), \end{aligned} \tag{1.12}$$

with $g(q, t) \in \mathbb{R}^{n_\lambda}$ describing the n_λ constraints. The vector $f_c(q, \lambda, t) \in \mathbb{R}^{n_q}$ describes the additional forces acting on the system, the so called *generalized constraint forces*. By d'Alembert's principle it can be shown that $f_c(q, \lambda, t) = G(q, t)^T \lambda$ with the *constraint matrix* $G(q, t) := \frac{d}{dq} g(q, t) \in \mathbb{R}^{n_\lambda, n_q}$ and n_λ unknown parameters $\lambda \in \mathbb{R}^{n_\lambda}$, see [8]. Therefore, the equations of motion of a holonomically constrained system can be written as

$$\begin{aligned} M(q)\ddot{q} =& f_a(q, \dot{q}, t) - G(q, t)^T \lambda, \\ 0 =& g(q, t). \end{aligned} \tag{1.13}$$

Nonholonomic Constraints

If nonholonomic constraints are considered in the system, then the equations of motion read

$$\begin{aligned} M(q)\ddot{q} =& f_a(q,\dot{q},t) - f_c(q,\dot{q},\lambda,t), \\ 0 =& g(q,\dot{q},t). \end{aligned} \tag{1.14}$$

In this case the constraint forces $f_c(q,\dot{q},\lambda,t)$ cannot be further simplified. In the following we will restrict to the case of holonomically constrained systems.

Constrained Systems with Friction

If friction between two bodies is considered in the system, then the generalized applied forces f_a include the friction forces which, in general, depend on the generalized constraint forces. This means that the generalized applied forces f_a are coupled to the constraint forces via the parameter λ. Then, the equations of motion read

$$\begin{aligned} M(q)\ddot{q} =& f_a(q,\dot{q},\lambda,t) - G^T(q,t)\lambda, \\ 0 =& g(q,t). \end{aligned} \tag{1.15}$$

We only consider dry friction or Coulomb friction, i.e., the friction force is assumed to be proportional to the normal force on the surface between the two bodies in the contact point, see [8].

Systems with Dynamical Force Elements

When dynamical force elements like electro-magnetic forces, control devices or hydraulic components influence the mechanical system, then additional first order differential equations are added to the equations of motion, see [8].
We give an example to illustrate the structure of the equations of motion.

Example 1.2.2. *[8] Spring and damper in series*
Consider the system in Figure 1.2 consisting of a mass m_1 and a spring and damper in series. To establish the equations of motion of such a system under gravity we assume a nonzero mass m_2. This yields

$$\begin{aligned} m_1\ddot{x} &= m_1 g - k(x-r), \\ m_2\ddot{r} &= m_2 g - d\dot{r} + k(x-r). \end{aligned}$$

Here, g denotes the gravity. If we set m_2 to zero we obtain

$$\begin{aligned} m_1\ddot{x} &= m_1 g - k(x-r), \\ \dot{r} &= \frac{k}{d}(x-r). \end{aligned}$$

The example illustrates the form of the equations of motion for constrained mechanical systems with dynamical force elements. Written in a more generalized form they read

$$\begin{aligned} M(q)\ddot{q} &= f_a(q,\dot{q},r,\lambda,t) - G^T(q,t)\lambda, \\ \dot{r} &= f_d(q,\dot{q},r,\lambda,t), \\ 0 &= g(q,t). \end{aligned} \tag{1.16}$$

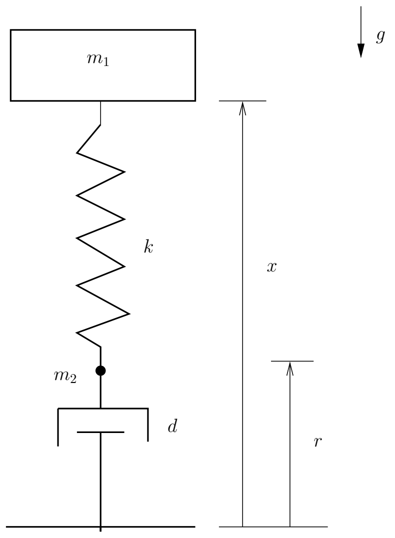

Figure 1.2: Spring and damper in series

Here, the vector $f_d(q,\dot{q},r,\lambda,t) \in \mathbb{R}^{n_r}$ describes the dynamical forces and the vector $r \in \mathbb{R}^{n_r}$ describes the dynamic force elements, where n_r is the number of coordinates describing such elements.

Contact Problems

We now consider contact problems between two rigid bodies. In contact problems we have to consider additional algebraic equations besides the constraint conditions $g(q,s,t) = 0$. Then, the equations of motion of multibody systems with contact conditions are given by

$$\begin{aligned} M(q)\ddot{q} &= f_a(q,\dot{q},s,\lambda,t) - G^T(q,s,t)\lambda, \\ 0 &= h(q,s,t), \\ 0 &= g(q,s,t). \end{aligned} \tag{1.17}$$

Here, the algebraic equations $0 = h(q,s,t) \in \mathbb{R}^{n_s}$ determine uniquely the contact point coordinates $s \in \mathbb{R}^{n_s}$ as function of the position coordinates q. The constraint matrix $G(q,s,t)$ can be obtained from the constraints

$$\begin{aligned} g(q,s,t) &= 0, \\ h(q,s,t) &= 0, \end{aligned}$$

and is given by $G(q,s,t) = [\frac{\partial g}{\partial q} - \frac{\partial g}{\partial s}(\frac{\partial h}{\partial s})^{-1}\frac{\partial h}{\partial q}](q,s,t)$ (see [3], Lemma 2.2.1), where $\frac{\partial h}{\partial s}$ is assumed to be nonsingular.

1.2.3 The Equations of Motion for Complex MBS Models

If several different constraints and elements as described in Section 1.2.2 are considered in the system, then the equations of motion have the following complex form, see [3].

$$\begin{aligned}
M(q)\ddot{q} &= f_a(q,\dot{q},r,w,s,\lambda,t) - G^T(q,s,t)\lambda,\\
\dot{r} &= f_d(q,\dot{q},r,w,s,\lambda,t),\\
0 &= d(q,\dot{q},r,w,s,\lambda,t),\\
0 &= h(q,s,t),\\
0 &= g(q,s,t).
\end{aligned} \tag{1.18}$$

Here, $f_a(q,\dot{q},r,w,s,\lambda,t) \in \mathbb{R}^{n_q}$ are again the applied and gyroscopic forces and $G(q,s,t) \in \mathbb{R}^{n_\lambda,n_q}$ is the constraint matrix with the associated constraint forces $G^T(q,s,t)\lambda$. We assume that the constraint matrix given by $G(q,s,t) = [\frac{\partial g}{\partial q} - \frac{\partial g}{\partial s}(\frac{\partial h}{\partial s})^{-1}\frac{\partial h}{\partial q}](q,s,t)$ has full row rank, to avoid redundant constraints (the so-called *Grübler condition*), see [10]. The vector $r \in \mathbb{R}^{n_r}$ describes the dynamic force elements and $f_d(q,\dot{q},r,w,s,\lambda,t) \in \mathbb{R}^{n_r}$ are the dynamical forces. Sometimes, force laws and constraints may be formulated more conveniently using auxiliary variables $w \in \mathbb{R}^{n_w}$, which are implicitly defined by the nonlinear equation $0 = d(q,\dot{q},r,w,s,\lambda,t)$ with nonsingular Jacobian $\frac{\partial d}{\partial w}$. In addition, we assume that the Jacobian $\frac{\partial h}{\partial s}$ is nonsingular, such that the equation $0 = h(q,s,t)$ determines $s = s(q,t)$.

In the following we will also need the corresponding first order system to the system above. Order reduction by introducing new variables $v \in \mathbb{R}^{n_q}$ for the velocities leads to the first order system

$$\begin{aligned}
\dot{q} &= v,\\
M(q)\dot{v} &= f_a(q,v,r,w,s,\lambda,t) - G^T(q,s,t)\lambda,\\
\dot{r} &= f_d(q,v,r,w,s,\lambda,t),\\
0 &= d(q,v,r,w,s,\lambda,t),\\
0 &= h(q,s,t),\\
0 &= g(q,s,t).
\end{aligned} \tag{1.19}$$

Chapter 2

BDF Methods for First Order Higher Index DAEs

Our aim is to solve higher order differential-algebraic systems of higher index. As higher order differential-algebraic equations are often solved by transforming to systems of first order equations and then applying methods for the first order systems, we will consider first order DAEs of higher index in this chapter. In the first part of this chapter we give some preliminary definitions and results, which are important to characterize the behavior of DAEs and focus on the so-called semi-explicit form. Then, we introduce the concepts of backward difference formulas (BDF) to solve ordinary differential equations (ODEs) and extend this approach to the numerical solution of DAEs. We consider BDF methods as numerical methods to solve higher index DAEs and give some convergence results for these systems. We describe the problems which arise for higher index systems.

2.1 Preliminaries for First Order Differential-Algebraic Systems

In this section we will consider different types of first order differential-algebraic systems, namely linear differential-algebraic equations with constant or variable coefficients, nonlinear differential-algebraic equations and semi-explicit differential-algebraic equations as subclass of nonlinear systems.
Linear differential-algebraic equations with variable coefficients are of the form

$$E(t)\dot{x}(t) = A(t)x(t) + f(t), \tag{2.1}$$

where $t \in \mathbb{I} \subseteq \mathbb{R}$ and

$$E, A \in \mathcal{C}(\mathbb{I}, \mathbb{R}^{n,m}), \quad f \in \mathcal{C}(\mathbb{I}, \mathbb{R}^{n}), \tag{2.2}$$

together with an initial condition

$$x(t_0) = x_0, \tag{2.3}$$

with $t_0 \in \mathbb{I}$ and $x_0 \in \mathbb{R}^m$.

For the simpler case of linear systems with constant coefficients system (2.1) simplifies to

$$E\dot{x}(t) = Ax(t) + f(t), \quad t \in \mathbb{I}, \tag{2.4}$$

where $E, A \in \mathbb{R}^{n,m}$ and $f \in \mathcal{C}(\mathbb{I}, \mathbb{R}^n)$.
In the most general form we consider nonlinear problems

$$F(t, x(t), \dot{x}(t)) = 0, \tag{2.5}$$

where $F \in \mathcal{C}(\mathbb{I} \times \mathbb{R}^m \times \mathbb{R}^m, \mathbb{R}^n)$ together with an initial condition

$$x(t_0) = x_0, \quad t_0 \in \mathbb{I}, \quad x_0 \in \mathbb{R}^m. \tag{2.6}$$

Definition 2.1.1. *[17]*
A function $x : \mathbb{I} \to \mathbb{R}^m$ is called a solution *of* (2.5)*, if $x \in \mathcal{C}^1(\mathbb{I}, \mathbb{R}^m)$ and x satisfies* (2.5) *pointwise.*
It is called solution of the initial value problem (2.5)*,* (2.6)*, if x is a solution of* (2.5) *and x satisfies the initial condition* (2.6)*.*
An initial condition (2.6) *is called* consistent*, if the corresponding initial value problem is solvable, i.e., has at least one solution.*

Throughout this work we always assume that the differential-algebraic systems we consider are solvable. For a detailed analysis of linear differential-algebraic systems and conditions for solvability, see [18, 19].
The behavior of differential-algebraic systems in numerical calculations is directly related to a property called index. There are different concepts of assigning an index to a DAE. We will focus on two different concepts of index, the differentiation index [15] and the strangeness index [19, 17].

Definition 2.1.2. *[15]*
The first order differential-algebraic system $F(t, x(t), \dot{x}(t)) = 0$ has differentiation index ν*, if ν is the minimal number of differentiations*

$$\begin{aligned} F(t, x(t), \dot{x}(t)) &= 0, \\ \frac{d}{dt} F(t, x(t), \dot{x}(t)) &= 0, \\ &\vdots \\ \frac{d^\nu}{dt^\nu} F(t, x(t), \dot{x}(t)) &= 0, \end{aligned} \tag{2.7}$$

such that (2.7) *allows to extract an explicit ODE $\dot{x}(t) = \varphi(t, x(t))$ by algebraic means (the so called underlying ODE).*

In the following we will refer to the differentiation index as *d-index*. Differential-algebraic equations with a d-index higher than 1 are called *higher index problems.*

Example 2.1.3. *Linear Equations with Constant Coefficients*
Consider the linear constant coefficient system

$$\dot{z}_2 + z_1 = \delta_1, \tag{2.8a}$$
$$\dot{z}_3 + z_2 = \delta_2, \tag{2.8b}$$
$$z_3 = \delta_3.$$

If we differentiate the first equation once with respect to t, the second equation twice and the third equation three times we get

$$
\begin{aligned}
\ddot{z}_2 + \dot{z}_1 &= \dot{\delta}_1, \\
z_3^{(3)} + \ddot{z}_2 &= \ddot{\delta}_2, \\
z_3^{(3)} &= \delta_3^{(3)}.
\end{aligned}
$$

Thus, we can extract an ODE

$$
\dot{z}_1 = \dot{\delta}_1 - \ddot{\delta}_2 + \delta_3^{(3)}. \tag{2.8c}
$$

The equations (2.8) *form the underlying ODE as described in Definition 2.1.2. This means that the system has d-index 3.*

Next, we want to introduce the strangeness index [17, 23]. First, we consider a linear differential-algebraic equation (2.1) and derive a condensed form via local and global equivalence transformations, which allow to extract some characteristic quantities.

Definition 2.1.4. *[23] Let (E_i, A_i), $i = 1, 2$, be two pairs of matrix functions with $E_i, A_i \in \mathcal{C}(\mathbb{I}, \mathbb{R}^{n,m})$. These pairs are called* (globally) equivalent *if there exist $P \in \mathcal{C}(\mathbb{I}, \mathbb{R}^{n,n})$, $Q \in \mathcal{C}^1(\mathbb{I}, \mathbb{R}^{m,m})$ with P, Q pointwise nonsingular, such that*

$$
\begin{aligned}
E_2 &= P E_1 Q, \\
A_2 &= P A_1 Q - P E_1 \dot{Q}.
\end{aligned} \tag{2.9}
$$

We get the corresponding local equivalence by choosing $\dot{Q}(t)$ independently of $Q(t)$ at a fixed point $t \in \mathbb{I}$.

Definition 2.1.5. *[23] Two pairs of matrices (E_i, A_i), $i = 1, 2$ with $E_i, A_i \in \mathbb{R}^{n,m}$ are called* (locally) equivalent *if there exist matrices $P \in \mathbb{R}^{n,n}$, $Q, S \in \mathbb{R}^{m,m}$ with P and Q nonsingular such that*

$$
\begin{aligned}
E_2 &= P E_1 Q, \\
A_2 &= P A_1 Q - P E_1 S.
\end{aligned} \tag{2.10}
$$

Using the local equivalence transformation (2.10), we obtain the following local invariants and condensed form for the pair of matrices (E, A).

Theorem 2.1.6. *Let $E, A \in \mathbb{R}^{n,m}$ and*

(a) T basis of kernel E,
(b) Z basis of corange E = kernel E^T,
(c) T' basis of cokernel E = range E^T,
(d) V basis of corange $(Z^T A T)$,

Then, the quantities

(a) r = rank E *(rank),*
(b) a = rank $(Z^T A T)$ *(algebraic part),*

(c) $s = \operatorname{rank}\,(V^T Z^T A T')$ *(strangeness),*

(d) $d = r - s$ *(differential part),*

(e) $u = n - r - a - s$ *(left undetermined part),*

(f) $v = m - r - a$ *(right undetermined part)*

are invariant under (2.10) *and* (E, A) *is equivalent to the condensed form*

$$\begin{array}{c} s \\ d \\ a \\ s \\ u \end{array} \left(\begin{bmatrix} I_s & 0 & 0 & 0 \\ 0 & I_d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & I_a & 0 \\ I_s & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right). \tag{2.11}$$

Here, the last column has width v.

Proof. See [23]. □

If we apply the results for the local condensed form (2.11) to equation (2.1) then we obtain functions $r, a, s : \mathbb{I} \to \mathbb{N}_0$ and we assume that these quantities are constant, i.e.,

$$r(t) \equiv r, \quad a(t) \equiv a, \quad s(t) \equiv s \quad \text{on } \mathbb{I}. \tag{2.12}$$

Then we get the following global condensed form.

Theorem 2.1.7. *Let* E, A *in* (2.1) *be sufficiently smooth and let* (2.12) *hold. Then the pair* (E, A) *is equivalent to a pair of matrix functions of the form*

$$\begin{array}{c} s \\ d \\ a \\ s \\ u \end{array} \left(\begin{bmatrix} I_s & 0 & 0 & 0 \\ 0 & I_d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & A_{12} & 0 & A_{14} \\ 0 & 0 & 0 & A_{24} \\ 0 & 0 & I_a & 0 \\ I_s & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right). \tag{2.13}$$

Here, the columns have dimensions s, d, a *and* v.

Proof. See [23]. □

Writing down the equations that belong to the matrix pair (2.13), we get

$$\begin{aligned} \dot{x}_1 &= A_{12}(t)x_2 + A_{14}(t)x_4 + f_1(t), && \text{(2.14a)} \\ \dot{x}_2 &= A_{24}(t)x_4 + f_2(t), \\ 0 &= x_3 + f_3(t), \\ 0 &= x_1 + f_4(t), && \text{(2.14b)} \\ 0 &= f_5(t). \end{aligned}$$

For the resulting system, obtained from (2.14) by differentiating equation (2.14b) and inserting in (2.14a), we can again compute characteristic values r, a, s, d, u, v and the condensed form (2.14) and we can then proceed inductively. This procedure leads to a sequence of pairs of matrix functions (E_i, A_i), $i \in \mathbb{N}_0$, starting with $(E_0, A_0) = (E, A)$. The corresponding characteristic quantities r_i, a_i, s_i are uniquely determined under the regularity assumptions (2.12) and the sequence $(r_i)_i$ strictly decreases unless s_i becomes 0. The value μ with $\mu = min\{i \in \mathbb{N}_0 | s_i = 0\}$ is called the *strangeness index (s-index)* of (2.1).

We can differentiate (2.1) with sufficiently smooth coefficient functions to obtain an *inflated* DAE

$$M_l(t)\dot{z}_l = N_l(t)z_l + g_l(t), \tag{2.15}$$

with

$$\begin{aligned}(M_l)_{i,j} &= \binom{i}{j}E^{(i-j)} - \binom{i}{j+1}A^{(i-j-1)}, \quad i,j = 0,\dots,l,\\ (N_l)_{i,j} &= \begin{cases} A^{(i)} & \text{for } i = 0,\dots,l, j = 0,\\ 0 & \text{else,}\end{cases}\\ (z_l)_i &= x^{(i)}, \quad i = 0,\dots,l,\\ (g_l)_i &= f^{(i)}, \quad i = 0,\dots,l,\end{aligned} \tag{2.16}$$

where $E^{(i)}$ denotes the i-th derivative of E with respect to t and we use the convention that $\binom{i}{j} = 0$ for $i < 0$, $j < 0$ or $j > i$.

Theorem 2.1.8. *Let the strangeness-index μ of (E, A) in (2.1) be well defined. Then there exist numbers a, d and v such that the inflated pair (M_μ, N_μ) associated with (E, A) has the following properties.*

1. *For all $t \in \mathbb{I}$ we have $rank M_\mu(t) = (\mu + 1)n - a - v$. This implies the existence of a smooth matrix function Z with orthonormal columns and size $((\mu + 1)n, a - v)$ and pointwise maximal rank satisfying $Z^T M_\mu = 0$.*

2. *For all $t \in \mathbb{I}$ we have $rank Z^T N_\mu[I_m 0 \dots 0]^T = a$ and without loss of generality Z can be partitioned as $[Z_2, Z_3]$, with Z_2 of size $((\mu + 1)n, a)$ and Z_3 of size $((\mu+1)n, v)$, such that $\hat{A}_2 = Z_2^T N_\mu[I_m 0 \dots 0]^T$ has full row rank a and that $Z_3^T N_\mu[I_m 0 \dots 0]^T = 0$. Furthermore, there exists a smooth matrix function T_2 with orthonormal columns and size (m, d), $d = m - a$ and pointwise maximal rank satisfying $\hat{A}_2 T_2 = 0$.*

3. *For all $t \in \mathbb{I}$ we have $rank E(t)T_2(t) = d$. This implies the existence of a smooth matrix function Z_1 with orthonormal columns and size (n, d) so that $\hat{E}_1 = Z_1^T E$ has constant rank d.*

Furthermore, system (2.1) has the same solution set as the s-index 0 system

$$\begin{bmatrix}\hat{E}_1(t)\\ 0\\ 0\end{bmatrix}\dot{x} = \begin{bmatrix}\hat{A}_1(t)\\ \hat{A}_2(t)\\ 0\end{bmatrix}x + \begin{bmatrix}\hat{f}_1(t)\\ \hat{f}_2(t)\\ \hat{f}_3(t)\end{bmatrix}, \tag{2.17}$$

where $\hat{A}_1 = Z_1^T A$, $\hat{f}_1 = Z_1^T f$, $\hat{f}_i = Z_i^T g_\mu$ for $i = 2, 3$.

Proof. See [23]. □

Consider now the general nonlinear problem (2.5). In the nonlinear case we pose Theorem 2.1.8 as hypothesis. To do this, we first assume that all functions are sufficiently smooth. We introduce a nonlinear derivative array of the form

$$F_l(t, x(t), \dot{x}(t), \dots, x^{(l+1)}(t)) = 0, \tag{2.18}$$

which is obtained by successive differentiation of the form

$$F_l(t, x(t), \dot{x}(t), \ldots, x^{(l+1)}(t)) = \begin{bmatrix} F(t, x(t), \dot{x}(t)) \\ \frac{d}{dt}F(t, x(t), \dot{x}(t)) \\ \vdots \\ \frac{d^l}{dt^l}F(t, x(t), \dot{x}(t)) \end{bmatrix}. \tag{2.19}$$

Partial derivatives of F_l are denoted by subscripts, i.e.,

$$F_{l;x} := \frac{\partial}{\partial x}F_l, \quad F_{l;\dot{x},\ldots,x^{(l+1)}} := \left[\frac{\partial}{\partial \dot{x}}F_l \ldots \frac{\partial}{\partial x^{(l+1)}}F_l\right],$$

and we denote by

$$\mathbb{L}_\mu := \{z_\mu \in \mathbb{I} \times \mathbb{R}^m \times \mathbb{R}^m \times \ldots \times \mathbb{R}^m | F_\mu(z_\mu) = 0\} \tag{2.20}$$

the solution set of the derivative array F_μ for some integer μ.

Hypothesis 2.1.9. *[21]*
Consider the general system of nonlinear differential-algebraic equations (2.5). *There exist integers μ, r, a, d and v such that $\mathbb{L}_\mu$ is not empty, and the following properties hold:*

1. *The set $\mathbb{L}_\mu \subseteq \mathbb{R}^{(\mu+2)m+1}$ forms a manifold of dimension $(\mu+2)m+1-r$.*
2. *We have*
$$rank F_{\mu;x,\dot{x},\ldots,x^{(\mu+1)}} = r$$
on $\mathbb{L}_\mu$.
3. *We have*
$$corank F_{\mu;x,\dot{x},\ldots,x^{(\mu+1)}} - corank F_{\mu-1;x,\dot{x},\ldots,x^{(\mu)}} = v$$
on $\mathbb{L}_\mu$. (The corank is the dimension of the corange and we use the convention that $corank F_{-1;x} = 0$.)
4. *We have*
$$rank F_{\mu;\dot{x},\ldots,x^{(\mu+1)}} = r - a$$
on $\mathbb{L}_\mu$ such that there are smooth full rank matrix functions Z_2 and T_2 defined on $\mathbb{L}_\mu$ of size $((\mu+1)n, a)$ and $(m, m-a)$, respectively, satisfying
$$Z_2^T F_{\mu;\dot{x},\ldots,x^{(\mu+1)}} = 0, \quad rank Z_2^T F_{\mu;x} = a, \quad Z_2^T F_{\mu;x} T_2 = 0$$
on $\mathbb{L}_\mu$.
5. *We have $rank F_{\dot{x}} T_2 = d = n - a - v$ on $\mathbb{L}_\mu$.*

When the nonlinear DAE (2.5) satisfy Hypothesis 2.1.9, then we call the least possible μ the *strangeness index* (s-index) of (2.5). A DAE with s-index $\mu = 0$ is called *strangeness-free.*
Theorem 2.1.8 holds for linear systems of the form (2.1) and Hypothesis 2.1.9 holds for general nonlinear systems (2.5). In the following we will restrict ourselves in both cases to regular quadratic systems with $m = n$ and $v = 0$.
The relation between d-index and s-index for first order systems is given in [19]. In [19] it has been shown that if both the d-index ν and the s-index μ are well-defined, we have $\nu = 0$ if $\mu = 0$ and $a = 0$ or $\nu = \mu + 1$ otherwise.

2.2 Semi-explicit Differential-Algebraic Systems

A frequently discussed subclass of nonlinear systems are semi-explicit systems. In this section we focus on differential-algebraic equations of d-index 1, 2 and 3 in semi-explicit form.

2.2.1 Semi-explicit Systems of d-Index 1

Consider the semi-explicit differential-algebraic system

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)), \\ 0 &= g(t, y(t), z(t)), \end{aligned} \tag{2.21}$$

with initial values $y(t_0) = y_0$ and $z(t_0) = z_0$, $y_0 \in \mathbb{R}^{n_y}$, $z_0 \in \mathbb{R}^{n_z}$ and sufficiently smooth functions $f : [t_0, t_0 + T] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ and $g : [t_0, t_0 + T] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$. The first equation is already in the form of an ordinary differential equation. If we now differentiate the constraints $0 = g(t, y(t), z(t))$ with respect to t we get

$$\begin{aligned} 0 = \frac{d}{dt} g(t, y, z) &= g_t(t, y, z) + g_y(t, y, z)\dot{y}(t) + g_z(t, y, z)\dot{z}(t) \\ &= g_t(t, y, z) + [g_y f](t, y, z) + g_z(t, y, z)\dot{z}(t). \end{aligned} \tag{2.22}$$

Here, partial derivatives are denoted by subscripts, i.e., $g_z := \frac{\partial}{\partial z} g(t, y, z)$ and for two functions f and g we define

$$[gf](t, y, z) := g(t, y(t), z(t)) f(t, y(t), z(t)).$$

Assume that $g_z(t, y, z)$ is nonsingular in a neighborhood of the solution of (2.21), then by multiplying (2.22) with g_z^{-1} we get

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)), \\ \dot{z}(t) &= -[g_z^{-1} g_t](t, y(t), z(t)) - [g_z^{-1} g_y f](t, y(t), z(t)). \end{aligned}$$

This is an ordinary differential equation, thus (2.21) has d-index 1.
Initial values y_0, z_0 for problem (2.21) are consistent if they satisfy the constraint equation $g(t_0, y_0, z_0) = 0$.

2.2.2 Semi-explicit Systems of d-Index 2

Consider the semi-explicit differential-algebraic system

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)), \\ 0 &= g(t, y(t)), \end{aligned} \tag{2.23}$$

with initial values $y(t_0) = y_0$, $z(t_0) = z_0$, $y_0 \in \mathbb{R}^{n_y}$, $z_0 \in \mathbb{R}^{n_z}$ and sufficiently smooth functions $f : [t_0, t_0 + T] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ and $g : [t_0, t_0 + T] \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_z}$. If we differentiate the constraints $0 = g(t, y(t))$ with respect to t, we get

$$0 = \frac{d}{dt} g(t, y) = g_t(t, y) + g_y(t, y)\dot{y}(t) = g_t(t, y) + [g_y f](t, y, z). \tag{2.24}$$

Assume that $[g_y f_z](t, y, z)$ is nonsingular in a neighborhood of the solution of (2.23). If we replace the constraints $0 = g(t, y(t))$ in (2.23) by the derivative (2.24), then we get a DAE in the form (2.21), which can be transformed into an ordinary differential equation by a second differentiation of the constraints. Therefore, the system (2.23) has d-index 2.
Initial values y_0, z_0 for problem (2.23) are consistent if they satisfy the conditions $g(t_0, y_0) = 0$ and $g_t(t_0, y_0) + g_y(t_0, y_0)f(t_0, y_0, z_0) = 0$.

2.2.3 Semi-explicit Systems of d-Index 3

Consider the semi-explicit differential-algebraic system

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)), \\ \dot{z}(t) &= k(t, y(t), z(t), u(t)), \\ 0 &= g(t, y(t)), \end{aligned} \tag{2.25}$$

with initial values $y(t_0) = y_0$, $z(t_0) = z_0$, $u(t_0) = u_0$, $y_0 \in \mathbb{R}^{n_y}$, $z_0 \in \mathbb{R}^{n_z}$, $u_0 \in \mathbb{R}^{n_u}$ and sufficiently smooth functions $f : [t_0, t_0 + T] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$, $k : [t_0, t_0 + T] \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_z}$ and $g : [t_0, t_0 + T] \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_u}$. We assume that $n_u \leq min(n_z, n_y)$ to avoid overdetermined systems.

If we differentiate the constraints $0 = g(t, y(t))$ twice with respect to t we get for the first derivative

$$0 = \frac{d}{dt} g(t, y) = g_t(t, y) + g_y(t, y)\dot{y}(t) = g_t(t, y) + [g_y f](t, y, z), \tag{2.26}$$

and for the second derivative

$$\begin{aligned} 0 &= \frac{d^2}{dt^2} g(t, y) = \frac{d}{dt}(g_t(t, y)) + \frac{d}{dt}(g_y(t, y)\dot{y}(t)) \\ &= g_{tt}(t, y) + g_{ty}(t, y)\dot{y}(t) + \frac{d}{dt}(g_y(t, y))\dot{y}(t) + g_y(t, y)\ddot{y}(t). \end{aligned} \tag{2.27}$$

The term $\frac{d}{dt}(g_y(t, y))$ leads to a product of a tensor of third order $g_{yy}(t, y)$, and a vector $\dot{y}(t)$

$$\frac{d}{dt}(g_y(t, y)) = g_{yy}(t, y)\dot{y}(t) + g_{yt}(t, y), \tag{2.28}$$

which has to be further multiplied by the vector $\dot{y}(t)$. For the vector-vector multiplication we use the notation of the scalar product $(\dot{y}(t), \dot{y}(t))$. Thus we get

$$\begin{aligned} 0 &= \frac{d^2}{dt^2} g(t, y) \\ &= g_{tt}(t, y) + [g_{ty} f](t, y, z) + [g_{yt} f](t, y, z) + [g_{yy}(f, f)](t, y, z) \\ &\quad + [g_y f_t](t, y, z) + [g_y f_y f](t, y, z) + [g_y f_z k](t, y, z, u). \end{aligned} \tag{2.29}$$

Assume that

$$[g_y f_z k_u](t, y, z, u) \quad \text{is nonsingular} \tag{2.30}$$

in a neighborhood of the solution of (2.25). If we replace the constraints
$0 = g(t, y(t))$ in (2.25) by the second derivative (2.29), then we get a DAE in

the form (2.21), which can be transformed into an ordinary differential equation by another differentiation of the constraints. Therefore, three differentiations of the constraints lead to an ordinary differential equation, thus the system (2.25) has d-index 3.
Initial values y_0, z_0, u_0 for problem (2.25) are consistent if they satisfy the three conditions

$$\begin{aligned}
0 &= g(t_0, y_0),\\
0 &= g_t(t_0, y_0) + g_y(t_0, y_0) f(t_0, y_0, z_0),\\
0 &= g_{tt}(t_0, y_0) + [g_{ty} f](t_0, y_0, z_0) + [g_{yt} f](t_0, y_0, z_0) + [g_{yy}(f, f)]\,(t_0, y_0, z_0)\\
&\quad + [g_y f_t](t_0, y_0, z_0)\,[g_y f_y f](\,t_0, y_0, z_0) + [g_y f_z k]\,(t_0, y_0, z_0, u_0).
\end{aligned}$$

2.3 BDF Methods for the Numerical Solution of First Order ODEs

In this section we discuss the concepts of the backward differentiation formulas (BDF) to solve initial value problems for ordinary differential equations in order to extend this approach to the numerical solution of DAEs. An introduction to BDF methods as described in this section is given in [33].
Consider the initial value problem

$$\begin{aligned}
\dot{y}(t) &= f(t, y(t)), \quad t \in [t_0, t_0 + T],\\
y(t_0) &= y_0.
\end{aligned} \tag{2.31}$$

The basic idea for deriving BDF methods is to differentiate the polynomial $p(t)$ which interpolates values $y_{n-k}, y_{n-k+1}, \dots, y_n$ of y and to set the derivative of $p(t)$ at t_n to $f(t_n, y_n)$.
Assume that the approximations $y_{n-k}, y_{n-k+1}, \dots, y_{n-1}$ to the exact solution $y(t)$ of (2.31) at $t_{n-k}, \dots, t_{n-1}$ are known. In order to derive a formula for y_n we consider the polynomial $p(t)$ of order k, which interpolates the values $\{(t_i, y_i) | i = n-k, \dots, n\}$, i.e.,

$$p(t_{n-i}) = y_{n-i}, \quad i = 0, \dots, k. \tag{2.32}$$

This polynomial can be expressed in terms of backward differences, see [33], as

$$p(t) = p(t_n + sh) = \sum_{j=0}^{k} (-1)^j \binom{-s}{j} \nabla^j y_n. \tag{2.33}$$

Here, we use a constant stepsize h and the backward differences are defined as follows.

Definition 2.3.1. *[33]*
The backward differences $\nabla^j y_n$ are recursively defined by

$$\begin{aligned}
\nabla^0 y_n &= y_n,\\
\nabla^{j+1} y_n &= \nabla^j y_n - \nabla^j y_{n-1} \text{ for } j \geq 0.
\end{aligned}$$

The unknown value y_n will now be determined in such a way that the polynomial $p(t)$ satisfies the differential equation (2.31) at t_n, i.e.,

$$\dot{p}(t_n) = f(t_n, y_n), \tag{2.34}$$

with

$$\begin{aligned} \left.\frac{dp(t)}{dt}\right|_{t=t_n} &= \frac{1}{h} \left.\frac{dp(t_n + sh)}{ds}\right|_{s=0} \\ &= \frac{1}{h}\sum_{j=0}^{k}(-1)^j \frac{d}{ds}\left.\binom{-s}{j}\right|_{s=0} \nabla^j y_n. \end{aligned}$$

This yields the implicit formula

$$\sum_{j=0}^{k} \delta_j \nabla^j y_n = hf(t_n, y_n), \tag{2.35}$$

with the coefficients

$$\delta_j := (-1)^j \frac{d}{ds}\left.\binom{-s}{j}\right|_{s=0}.$$

Using the definition of the binomial coefficients

$$(-1)^j \binom{-s}{j} = \frac{1}{j!} s(s+1)\ldots(s+j-1) \text{ for } j > 0$$

and $\binom{-s}{0} = 1$, the coefficients δ_j are obtained by

$$\delta_0 = 0, \quad \delta_j = \frac{1}{j} \text{ for } j \geq 1.$$

Therefore, we can write formula (2.35) as

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_n = hf(t_n, y_n). \tag{2.36}$$

These multistep formulas are known as k-step *Backward Differentiation Formulas* (BDF). The BDF formulas can also be written in a scaled form as

$$\sum_{i=0}^{k} \alpha_i y_{n-i} = hf(t_n, y_n). \tag{2.37}$$

In Table 2.1 the coefficients of the BDF methods up to order $k = 6$ are given. In [33] some characteristic properties of BDF methods are discussed. In particular BDF-methods are $A(\alpha)$-stable for $k \leq 6$ and the order of consistency is $p = k$. For the concept of stability and $A(\alpha)$-stability see also [33]. If we consider variable stepsizes $h_n = t_n - t_{n-1}$ the BDF methods can be derived analogous to the constant stepsize case. The interpolation polynomial (2.33) is now presented by use of divided differences for the non-equidistant grid.

Table 2.1: BDF coefficients

α_i	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
$k=1$	1	-1					
$k=2$	$\frac{3}{2}$	-2	$\frac{1}{2}$				
$k=3$	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$			
$k=4$	$\frac{25}{12}$	-4	3	$-\frac{4}{3}$	$\frac{1}{4}$		
$k=5$	$\frac{137}{60}$	-5	5	$-\frac{10}{3}$	$\frac{5}{4}$	$-\frac{1}{5}$	
$k=6$	$\frac{147}{60}$	-6	$\frac{15}{2}$	$-\frac{20}{3}$	$\frac{15}{4}$	$-\frac{6}{5}$	$\frac{1}{6}$

Definition 2.3.2. *[4]*
For a function y the divided differences *are defined recursively by*

$$\begin{aligned} y[t_n] &:= y_n, \\ y[t_n, t_{n-1}, \dots, t_{n-k}] &:= \frac{y[t_n, \dots, t_{n-k+1}] - y[t_{n-1}, \dots, t_{n-k}]}{t_n - t_{n-k}}. \end{aligned}$$

Using the divided differences presentation the interpolation polynomial $p(t)$ reads

$$p(t) = \sum_{j=0}^{k} \prod_{i=0}^{j-1} (t - t_{n-i}) y[t_n, \dots t_{n-j}]. \tag{2.38}$$

The unknown value y_n is again determined by requiring that

$$\dot{p}(t_n) = f(t_n, y_n). \tag{2.39}$$

The variable stepsize BDF method is then given by

$$\sum_{j=1}^{k} \prod_{i=1}^{j-1} (t_n - t_{n-i}) y[t_n, \dots, t_{n-j}] = f(t_n, y_n), \tag{2.40}$$

or in a scaled form by

$$\sum_{i=0}^{k} \alpha_{n,i} y_{n-i} = h_n f(t_n, y_n). \tag{2.41}$$

The coefficients $\alpha_{n,i}$ now depend on the past and current stepsizes.
To compute the new iterate y_n at step n during the integration we have to solve equation (2.37) in the case of constant stepsizes or equation (2.41) if using variable stepsizes. This results in a system of nonlinear equations, which are normally solved by means of a modified Newton method. This means that in

each integration step of a BDF method we have to solve a nonlinear system of equations of the form

$$G(y_n) = \sum_{i=0}^{k} \alpha_{n,i} y_{n-i} - h_n f(t_n, y_n) = 0. \tag{2.42}$$

2.4 BDF Methods for the Numerical Solution of First Order DAEs

We now extend the approach given in Section 2.3 to solve the initial value problem for differential-algebraic systems. In the following we will recall some of the results for the numerical solution of different types of first order DAEs by BDF methods. For the proofs of the following theorems we refer to [4, 5, 12, 17].

2.4.1 Linear Constant Coefficient Systems

We first consider the linear constant coefficient system (2.4). Here, we only consider the quadratic case with $m = n$. A BDF method with constant stepsize h applied to (2.4) has the form

$$E\frac{1}{h}\sum_{i=0}^{k} \alpha_i x_{n-i} = Ax_n + f(t_n), \tag{2.43}$$

with given starting values $x_0, \dots, x_{k-1}$, where x_i denotes the numerical approximation to $x(t_i)$ with $t_i = t_0 + ih$.
At each integration step we have to solve a nonlinear system of equations of the form

$$G(x_n) = E\frac{1}{h}\sum_{i=0}^{k} \alpha_i x_{n-i} - Ax_n - f(t_n) = 0. \tag{2.44}$$

Theorem 2.4.1. *The k-step constant stepsize BDF method* (2.43) *for $k < 7$ applied to the linear constant coefficient DAE system* (2.4) *of d-index ν is convergent of order $O(h^k)$ after $(\nu - 1)k + 1$ steps.*

Proof. See [4], Theorem 3.1.1. □

We can observe that for higher index systems ($\nu \geq 2$) the numerical solution converges in an interval bounded away from the initial time.

The BDF method with variable stepsize applied to (2.4) has the form

$$E\frac{1}{h_n}\sum_{i=0}^{k} \alpha_{i,n} x_{n-i} = Ax_n + f(t_n), \tag{2.45}$$

with given starting values $x_0, \dots, x_{k-1}$ and the current stepsize h_n.

Theorem 2.4.2. *If a k-step variable stepsize BDF method* (2.45) *with $k < 7$ is applied to the constant coefficient system* (2.4) *with d-index ν and the ratio of adjacent stepsizes is bounded by $\frac{h_{k+1}}{h_k} = 1 + O(h)$, as $h \to 0$, then the global error in the numerical solution is $O(h_{max}^q)$, where $q = min(k, k - \nu + 2)$.*

Proof. See [12], Theorem 2.2. □

Theorem 2.4.2 states a problem which arises in the numerical solution of higher index DAEs by BDF methods. If a system of d-index $\nu = 3$ is numerically solved by a BDF method with $k = 1$, i.e., the implicit Euler method with variable stepsizes, the solution is expected to contain $O(1)$ errors. If we consider problems of d-index $\nu \geq 3$, then we have to expect a reduction of the convergence order by at least one.

2.4.2 Linear Variable Coefficient Systems

We now consider a linear time-dependent system (2.1). A BDF method for system (2.1) is given by

$$E(t_n)\frac{1}{h}\sum_{i=0}^{k}\alpha_i x_{n-i} = A(t_n)x_n + f(t_n), \tag{2.46}$$

with given starting values $x_0, \ldots, x_{k-1}$. At each integration step we have to solve a nonlinear system of equations of the form

$$G(x_n) = E(t_n)\frac{1}{h}\sum_{i=0}^{k}\alpha_i x_{n-i} - A(t_n)x_n - f(t_n) = 0. \tag{2.47}$$

For the variable coefficient case we have the following convergence result.

Theorem 2.4.3. *Let the s-index μ be well defined for the pair $(E(t), A(t))$ of sufficiently smooth matrix functions, with s-index $\mu = 0$. Let the matrix pencil $(E(t), A(t))$ be regular for all $t \in [t_0, t_0 + T]$, then the constant stepsize BDF* (2.46) *for $k < 7$ applied to system* (2.1) *is convergent of order k.*

Proof. See [17], Theorem 41. □

2.4.3 Nonlinear Systems

Consider the general nonlinear system (2.5). The k-step BDF method with constant stepsize $h = t_n - t_{n-1}$ applied to the nonlinear DAE (2.5) has the form

$$F(t_n, x_n, \frac{1}{h}\sum_{i=0}^{k}\alpha_i x_{n-i}) = 0, \tag{2.48}$$

for given starting values $x_0, \ldots, x_{k-1}$.
In the following we restrict ourselves to the subclass of semi-explicit systems and give convergence results for semi-explicit systems of d-index $\nu \in \{1, 2, 3\}$.
First, we consider the semi-explicit system of d-index 1

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)), \\ 0 &= g(t, y(t), z(t)), \end{aligned} \tag{2.49}$$

as described in Section 2.2.1.
A k-step BDF method applied to (2.49) is given by

$$\begin{aligned} \sum_{i=0}^{k}\alpha_i y_{n-i} &= hf(t_n, y_n, z_n), \\ 0 &= g(t_n, y_n, z_n), \end{aligned} \tag{2.50}$$

with given starting values $y_0, \dots, y_{k-1}$ and $z_0, \dots, z_{k-1}$. Here, g can be formally solved for z_n in terms of y_n and t_n using the implicit function theorem, i.e., $z_n = \tilde{g}(t_n, y_n)$, and then z_n can be inserted into formula (2.50). We get the same convergence results as would be obtained if applying the BDF method to the ODE $\dot{y} = f(t, y, \tilde{g}(t, y))$. Therefore, BDF methods applied to semi-explicit d-index one DAEs are stable and convergent to the same order of accuracy as for ODEs.
Furthermore, if variable stepsize BDF methods are implemented in such a way that the method is stable for standard ODEs, then the k-step BDF method with variable stepsize for $k < 7$ is convergent for nonlinear DAEs of d-index one, see [4].

While the fixed and variable stepsize BDF methods converge for problems of d-index 1, for higher index problems with d-index greater than 1 even the constant stepsize BDF method can fail. Some reasons for this behavior are that the formulas are not accurate when the stepsize changes and that the error estimates used in the codes to control the stepsize are not realistic. For details see [4].
To deal with higher index problems we consider the semi-explicit system of d-index 2

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)), \\ 0 &= g(t, y(t)), \end{aligned} \tag{2.51}$$

and the semi-explicit system of d-index 3

$$\begin{aligned} \dot{y}(t) &= f(t, y(t), z(t)), \\ \dot{z}(t) &= k(t, y(t), z(t), u(t)), \\ 0 &= g(t, y(t)), \end{aligned} \tag{2.52}$$

with consistent initial values $y(t_0) = y_0$, $z(t_0) = z_0$, $u(t_0) = u_0$, as described in Section 2.2.2 and 2.2.3.
A BDF approximation of the d-index 2 system (2.51) is given by

$$\begin{aligned} \sum_{i=0}^{k} \alpha_i y_{n-i} &= h f(t_n, y_n, z_n), \\ 0 &= g(t_n, y_n), \end{aligned} \tag{2.53}$$

and for the d-index 3 system (2.52) by

$$\begin{aligned} \sum_{i=0}^{k} \alpha_i y_{n-i} &= h f(t_n, y_n, z_n), \\ \sum_{i=0}^{k} \alpha_i z_{n-i} &= h k(t_n, y_n, z_n, u_n), \\ 0 &= g(t_n, y_n), \end{aligned} \tag{2.54}$$

with a number of starting values $y_0, \dots, y_{k-1}$, $z_0, \dots, z_{k-1}$ and $u_0, \dots, u_{k-1}$.

Definition 2.4.4. *[5]*
The starting values $y_0, \dots, y_{k-1}$, $z_0, \dots, z_{k-1}$ *and* $u_0, \dots, u_{k-1}$ *with* $y_j \in \mathbb{R}^{n_y}$,

$z_j \in \mathbb{R}^{n_z}$ *and* $u_j \in \mathbb{R}^{n_u}$ *for* $j = 0, \dots, k-1$ *are said to be* numerically consistent to order $k+1$ *if there exists a solution to the systems* (2.51),(2.52)*, such that*

$$\begin{aligned} \|y_j - y(t_j)\| &= O(h^{k+1}), \\ \|g(t_j, y_j)\| &= O(h^{k+1}) \end{aligned}$$

for the d-index 2 system (2.51) *and*

$$\begin{aligned} \|y_j - y(t_j)\| &= O(h^{k+1}), \\ \|z_j - z(t_j)\| &= O(h^{k+1}), \\ \|g(t_j, y_j)\| &= O(h^{k+2}) \end{aligned}$$

for the d-index 3 system (2.52) *with* $j = 0, \dots, k-1$*.*

We get the following convergence results for the d-index 2 and d-index 3 problem.

Theorem 2.4.5. *There exists a numerical solution of the d-index 2 system* (2.51) *by the k-step BDF* (2.53) *with constant stepsize h for* $k < 7$ *which converges globally with k-th order accuracy to a solution of* (2.51) *if the starting values are numerically consistent to order* $k+1$*.*

Proof. See [5], Theorem 1. □

Theorem 2.4.6. *There exists a numerical solution of the d-index 3 system* (2.52) *by the k-step BDF* (2.54) *with constant stepsize h for* $k < 7$ *which converges globally with k-th order accuracy to a solution of* (2.52) *after* $k+1$ *steps if the starting values are numerically consistent to order* $k+1$*, and the algebraic equations at each step are solved to* $O(h^{k+2})$ *accuracy for* $k \leq 2$*, and* $O(h^{k+3})$ *accuracy for* $k = 1$*.*

Proof. See [4], Theorem 3.2.3. □

In the case of variable stepsizes the k-step variable stepsize BDF has order $k - \nu + 2$ for d-index ν DAEs in semi-explicit form, provided the stepsize sequence is stable for ODEs, see [4].

We see the following problems when we want to solve higher index DAEs.
Variable-step variable-order BDF methods converge if the d-index does not exceed two and the methods are convergent for ordinary differential equations. If we consider problems of d-index $\nu \geq 3$, then we have to expect a reduction of the convergence order by at least one. For these systems the BDF method with $k = 1$ (implicit Euler method) in general fails to converge in the first $\nu - 2$ steps after a change of stepsize. When the implicit Euler method is used, numerical solutions containing O(1) errors will occur after every step where such a change of stepsize is made. As in almost all codes the integration process starts with the first order formula, the initial point may be regarded as one of those positions, where the stepsize is changed (from 0 to a positive value). Altogether, methods for solving index 3 DAEs based on BDFs suffer a loss of accuracy whenever there is a change of stepsize or a change of order in the method.
Another difficulty concerns stability for general higher index DAEs. There are d-index 2 and d-index 3 DAEs for which the implicit Euler method is unstable, see [4].
The following example is taken from [1] and illustrates the behavior of the implicit Euler method applied to a d-index 3 problem in semi-explicit form.

Example 2.4.7. *Consider the following second order system*

$$\begin{aligned} \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} &= 2 \begin{bmatrix} y_2 \\ -y_1 \end{bmatrix} + \lambda \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \\ 0 &= y_1^2 + y_2^2 - 1. \end{aligned} \tag{2.55}$$

A corresponding first order system reads

$$\begin{aligned} \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} &= \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \\ \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} &= \begin{bmatrix} 2y_2 + y_1\lambda \\ -2y_1 + y_2\lambda \end{bmatrix}, \\ 0 &= y_1^2 + y_2^2 - 1. \end{aligned}$$

This is a semi-explicit system of d-index 3 as described in Section 2.2.3, with $n_y = 2$, $n_u = 1$, $n_z = 2$, $k(t, y, z, \lambda) = [2y_2 + y_1\lambda, -2y_1 + y_2\lambda]^T$, $f(t, y, z) = [z_1, z_2]^T$ *and* $g(y) = y_1^2 + y_2^2 - 1$ *with* $y = [y_1, y_2]^T$, $z = [z_1, z_2]^T$, $u = \lambda$. *Note that* $(\partial^2 k/\partial\lambda^2) \equiv 0$ *and that* $\partial g/\partial y \cdot \partial f/\partial z \cdot \partial k/\partial\lambda = 2(y_1^2 + y_2^2) \equiv 2$ *is nonsingular for all t and all y, which lie in the solution manifold. Therefore, the DAE is of d-index 3.*
The equations model a particle on a circular track subject to a tangential force. The analytic solution to this problem is given by

$$\begin{aligned} y_1(t) &= \sin t^2, \\ y_2(t) &= \cos t^2, \\ z_1(t) &= 2t \cos t^2, \\ z_2(t) &= -2t \sin t^2, \\ \lambda(t) &= -4t^2. \end{aligned}$$

If the initial point is taken to be $t_0 = 1$, *then consistent values are*
$y_0 = [\sin 1, \cos 1]^T$, $z_0 = [2\cos 1, -2\sin 1]^T$.
Solving the problem by the implicit Euler method, one obtains $O(h)$ *errors in* $y = [y_1, y_2]^T$ *and* $z = [z_1, z_2]^T$. *Table 2.2 shows the absolute error of the algebraic variable* λ, *when the test problem was solved using the implicit Euler method with stepsize* $h = 0.005$ *and* $h = 0.01$. $O(1)$ *errors are displayed in bold face. We observe* $O(1)$ *errors in the algebraic variable* λ *at step 1, where the stepsize changes from* 0 *to* h. *Table 2.3 shows the absolute error of the algebraic variable when using the implicit Euler method with variable stepsizes. Here, we observe* $O(1)$ *errors, whenever the stepsize is changed.*

There are different approaches to overcome these problems.
Arévalo and Löstedt [1] developed an algorithm for correcting the numerical values after a change of stepsize for d-index 3 DAEs in semi-explicit form. They introduce a correction mechanism to increase the accuracy of the algebraic variables from $O(1)$ to $O(h)$ after a change of stepsize or a change of order in the BDF method.
In chapter 3 we discuss a method introduced by Sand [27] to directly solve the higher order system, without transforming to first order systems. This approach is shown to have a better solution behavior than the ordinary BDF methods, when we consider changes in stepsize and order.

	$\lvert\lambda(t_j) - \lambda_j\rvert$	
t_j	$h = 0.0050$	$h = 0.0100$
1.0050	**2.0400**	
1.0100	0.0409	**2.0810**
1.0150	0.0419	
1.0200	0.0429	0.0836
1.0300	0.0451	0.0877
1.0400	0.0474	0.0921
1.0500	0.0497	0.0967

Table 2.2: Constant stepsize implicit Euler method

Step no.	Stepsize $\times 10^{-3}$	$\lvert\lambda(t_j) - \lambda_j\rvert$ (impl. Euler)
1	1.000	**2.0080**
2	1.000	0.0080
3	0.200	**8.0303**
4	0.040	**8.0348**
5	0.008	**8.0357**
6	0.008	0.0001
7	0.016	**1.0047**
8	0.032	**1.0048**
9	0.064	**1.0052**
10	0.064	0.0006

Table 2.3: Variable stepsize implicit Euler method

Chapter 3

BDF Methods for Second Order Higher Index DAEs

In this chapter we describe numerical methods to solve second order differential-algebraic equations of higher index. We will focus on the case of second order DAEs with d-index 3 as they arise in the equations of motion of multibody systems.
In the first part we give some preliminary definitions and show the difficulties which arise if we solve higher order systems by simply transforming to first order systems and then applying the numerical method. In the second part we introduce modified BDF methods for the numerical solution of higher order higher index DAEs and concentrate on second order systems of d-index 3.

3.1 Preliminaries for Second Order Differential-Algebraic Systems

In this section we extend the definitions of the d-index and the s-index given in Chapter 2 to fit to the case of higher order systems. For higher order DAEs we have the following definition for the d-index.

Definition 3.1.1. *The d-th order differential-algebraic system*

$$F(t, x(t), \dot{x}(t), \ldots, x^{(d)}(t)) = 0$$

has d-index ν, if the corresponding first order system

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= x_3, \\ &\vdots \\ \dot{x}_{d-1} &= x_d, \\ F(t, x_1, x_2, \ldots, x_d, \dot{x}_d) &= 0, \end{aligned}$$

with $[x_1, x_2, \ldots, x_d] := [x, \dot{x}, \ldots, x^{(d-1)}]$, has d-index ν.

The concept of the s-index for linear higher order DAEs is introduced in [24, 30]. Here, we restrict to the case of linear second order DAEs with variable coefficients of the form

$$A_2(t)\ddot{x}(t) + A_1(t)\dot{x}(t) + A_0(t)x(t) = f(t), \quad t \in \mathbb{I}, \tag{3.1}$$

where $A_i \in \mathcal{C}(\mathbb{I}, \mathbb{C}^{m,n})$ for $i = 0, 1, 2$ and $f(t) \in \mathcal{C}(\mathbb{I}, \mathbb{C}^m)$ is sufficiently smooth. In [24, 30] local and global condensed forms are derived similar to the strategies used in the first order case as described in Section 2.1. First, a local condensed form is derived. For constant matrices $A_2, A_1, A_0 \in \mathbb{C}^{m,n}$ the matrix triple (A_2, A_1, A_0) is equivalent to the matrix triple $(\hat{A}_2, \hat{A}_1, \hat{A}_0)$ of the following form

$$\left(\begin{bmatrix} I_{s^{(0,1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I_{s^{(1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{s^{(0,2)}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{d^{(2)}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \right.$$

$$\begin{bmatrix} 0 & 0 & * & * & 0 & 0 & * & * \\ 0 & 0 & * & * & 0 & 0 & * & * \\ 0 & 0 & * & * & 0 & 0 & * & * \\ 0 & 0 & * & * & 0 & 0 & * & * \\ 0 & 0 & 0 & * & I_{s^{(0,1)}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{d^{(1)}} & 0 & 0 \\ I_{s^{(0,1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I_{s^{(1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.2}$$

$$\begin{array}{c c}
\left[\begin{array}{cccccccc}
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & 0 & 0 & 0 & 0 & 0 & I_a & 0 \\
0 & 0 & 0 & 0 & I_{s^{(0,1)}} & 0 & 0 & 0 \\
0 & 0 & I_{s^{(0,2)}} & 0 & 0 & 0 & 0 & 0 \\
I_{s^{(0,1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]\Bigg) &
\begin{array}{c}
s^{(0,1,2)} \\ s^{(1,2)} \\ s^{(0,2)} \\ d^{(2)} \\ s^{(0,1)} \\ d^{(1)} \\ s^{(0,1,2)} \\ s^{(1,2)} \\ a \\ s^{(0,1)} \\ s^{(0,2)} \\ s^{(0,1,2)} \\ v
\end{array}
\end{array}.$$

For triples $(A_2(t), A_1(t), A_0(t))$ of matrix-valued functions we can derive the local condensed form (3.2) at any fixed value $\hat{t} \in \mathbb{I}$ and then we can calculate the characteristic quantities for $(A_2(\hat{t}), A_1(\hat{t}), A_0(\hat{t}))$, so that we obtain functions

$$r, a, s^{(0,1,2)}, s^{(0,1)}, d^{(1)}, s^{(1,2)}, s^{(0,2)}, d^{(2)}, u, v : \mathbb{I} \to \mathbb{N}_0, \tag{3.3}$$

where $r = rank(A_2(\hat{t}))$ and $u = n - r - s^{(0,1)} - d^{(1)} - a$. Under the regularity assumptions

$$\begin{aligned}
&r(t) \equiv r, a(t) \equiv a, s^{(0,1,2)}(t) \equiv s^{(0,1,2)}, s^{(0,1)}(t) \equiv s^{(0,1)}, \\
&d^{(1)}(t) \equiv d^{(1)}, s^{(1,2)}(t) \equiv s^{(1,2)}, s^{(0,2)}(t) \equiv s^{(0,2)}
\end{aligned} \tag{3.4}$$

we can obtain a global condensed form for triples of matrix-valued functions via global equivalence transformations. In [24] it is shown that the triple $(A_2(t), A_1(t), A_0(t))$ is globally equivalent to $(\hat{A}_2(t), \hat{A}_1(t), \hat{A}_0(t))$ of the condensed form

$$\Bigg(\left[\begin{array}{cccccccc}
I_{s^{(0,1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & I_{s^{(1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & I_{s^{(0,2)}} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & I_{d^{(2)}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right],$$

$$
\begin{bmatrix}
0 & 0 & * & * & 0 & 0 & * & * \\
0 & 0 & * & * & 0 & 0 & * & * \\
0 & 0 & * & * & 0 & 0 & * & * \\
0 & 0 & * & * & 0 & 0 & * & * \\
0 & 0 & 0 & 0 & I_{s^{(0,1)}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I_{d^{(1)}} & 0 & 0 \\
I_{s^{(0,1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & I_{s^{(1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \tag{3.5}
$$

$$
\left.\begin{bmatrix}
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & * & 0 & * & 0 & * & 0 & * \\
0 & 0 & 0 & 0 & 0 & 0 & I_a & 0 \\
0 & 0 & 0 & 0 & I_{s^{(0,1)}} & 0 & 0 & 0 \\
0 & 0 & I_{s^{(0,2)}} & 0 & 0 & 0 & 0 & 0 \\
I_{s^{(0,1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}\right)
\qquad
\begin{matrix}
s^{(0,1,2)} \\ s^{(1,2)} \\ s^{(0,2)} \\ d^{(2)} \\ s^{(0,1)} \\ d^{(1)} \\ s^{(0,1,2)} \\ s^{(1,2)} \\ a \\ s^{(0,1)} \\ s^{(0,2)} \\ s^{(0,1,2)} \\ v
\end{matrix} .
$$

If we consider the associated system of DAEs

$$
\hat{A}_2(t)\ddot{y}(t) + \hat{A}_1(t)\dot{y}(t) + \hat{A}_0(t)y(t) = \hat{f}(t), \tag{3.6}
$$

then this system can be transformed into an equivalent second order system of DAEs

$$
A_2^{<1>}(t)\ddot{y}(t) + A_1^{<1>}(t)\dot{y}(t) + A_0^{<1>}(t)y(t) = f^{<1>}(t), \tag{3.7}
$$

with $(A_2^{<1>}(t), A_1^{<1>}(t), A_0^{<1>}(t); f^{<1>})$ being of the following form

$$
\left(\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & I_{d^{(2)}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & * & * & 0 & 0 & 0 & * \\
0 & * & * & * & 0 & * & 0 & * \\
0 & 0 & * & * & 0 & 0 & 0 & * \\
0 & 0 & * & * & 0 & 0 & 0 & * \\
0 & 0 & 0 & * & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I_{d^{(1)}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & I_{s^{(1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},\right.
$$

$$\left.\begin{bmatrix} 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 & I_a & 0 \\ 0 & 0 & 0 & 0 & I_{s^{(0,1)}} & 0 & 0 & 0 \\ 0 & 0 & I_{s^{(0,2)}} & 0 & 0 & 0 & 0 & 0 \\ I_{s^{(0,1,2)}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} ; \begin{bmatrix} \hat{f}_1(t) - \ddot{\hat{f}}_{12}(t) \\ \hat{f}_2(t) - \dot{\hat{f}}_8(t) \\ \hat{f}_3(t) - \ddot{\hat{f}}_{11}(t) \\ \hat{f}_4(t) \\ \hat{f}_5(t) - \dot{\hat{f}}_{10}(t) \\ \hat{f}_6(t) \\ \hat{f}_7(t) - \dot{\hat{f}}_{12}(t) \\ \hat{f}_8(t) \\ \hat{f}_9(t) \\ \hat{f}_{10}(t) \\ \hat{f}_{11}(t) \\ \hat{f}_{12}(t) \\ \hat{f}_{13}(t) \end{bmatrix} \right) . \tag{3.8}$$

The index reduction procedure can then be continued as follows. For the triple $(A_2^{<1>}(t), A_1^{<1>}(t), A_0^{<1>}(t))$ in (3.8), we can again transform to the condensed form (3.5) and then pass it to the form (3.8). Thus, we obtain a sequence of triples of matrix functions $(A_2^{<i>}(t), A_1^{<i>}(t), A_0^{<i>}(t))$, $i \in \mathbb{N}_0$, where $(A_2^{<0>}(t), A_1^{<0>}(t), A_0^{<0>}(t)) = (A_2(t), A_1(t), A_0(t))$. After a finite number of steps μ, the strangeness $s^{(0,1,2)}_{<\mu>}$, $s^{(0,2)}_{<\mu>}$, $s^{(0,1)}_{<\mu>}$ and $s^{(1,2)}_{<\mu>}$ corresponding to $(A_2^{<\mu>}(t), A_1^{<\mu>}(t), A_0^{<\mu>}(t))$ vanish. We call μ the *strangeness-index* or *s-index* of the second order system of DAEs (3.1) and we call the final equivalent second order system of DAEs *strangeness-free.*

3.2 Problems in the Order Reduction

Usually second order systems are turned into first order systems by introducing new variables for the first order derivatives. This classical transformation of a second order system of DAEs to first order may lead to certain difficulties. We consider again the linear second order system (3.1). If the degree of differentiability of the right-hand side $f(t)$ in system (3.1) is limited, then the transformation to first order may be mathematically incorrect, i.e., there may not exist any continuous solution to the resulting first order system, whereas there exist continuous solutions to the original second order system. The reason for this behavior is the coupling between the differential and the algebraic equations in the system.

Example 3.2.1. *[24]*
Consider the linear second order constant coefficient DAE

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \ddot{x}(t) + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \dot{x}(t) + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} x(t) = f(t), \quad t \in \mathbb{I}, \tag{3.9}$$

where $x(t) = [x_1(t), x_2(t)]^T$*, and* $f(t) = [f_1(t), f_2(t)]^T$*. System* (3.9) *has the unique solution*

$$\begin{cases} x_1(t) = f_2(t), \\ x_2(t) = f_1(t) - \dot{f}_2(t) - \ddot{f}_2(t). \end{cases} \tag{3.10}$$

Using the classical transformation to first order with

$$v(t) = [v_1(t), v_2(t)]^T = [\dot{x}_1(t), \dot{x}_2(t)]^T, \quad y(t) = [v_1(t), v_2(t), x_1(t), x_2(t)]^T,$$

we obtain the first order system

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dot{y}(t) + \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} y(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ 0 \\ 0 \end{bmatrix}, \tag{3.11}$$

which has the unique solution

$$\begin{cases} x_1(t) = f_2(t), \\ x_2(t) = f_1(t) - \dot{f}_2(t) - \ddot{f}_2(t), \\ v_1(t) = \dot{f}_2(t), \\ v_2(t) = \dot{f}_1(t) - \ddot{f}_2(t) - f_2^{(3)}(t). \end{cases} \tag{3.12}$$

Thus, the solution requires the derivatives of f_2 up to order 3, while the solution of the original systems only requires $\ddot{f}_2$.

As we see in Example 3.2.1 the classical approach of introducing the derivatives of the unknown function $x(t)$ as new variables may lead to higher smoothness requirements for the inhomogeneity. Introducing only some derivatives may avoid this difficulty, but in general we do not know which derivatives can be included without difficulties. The condensed form introduced in Section 3.1 for linear higher order systems allows an identification of those second derivatives of variables that can be replaced to obtain a first order system without changing the smoothness requirements.

Example 3.2.2. *[24]*
Consider again the system (3.9) of second order DAEs given in Example 3.2.1. System (3.9) can be equivalently transformed to the following strangeness-free system

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \ddot{\tilde{x}}(t) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \dot{\tilde{x}}(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tilde{x}(t) = \tilde{f}(t), \tag{3.13}$$

where $\tilde{x}(t) = [x_1(t), x_2(t)]^T$, and $\tilde{f}(t) = [f_1(t) - \dot{f}_2(t) - \ddot{f}_2(t), f_2(t)]^T$. The s-index of system (3.9) is $\mu = 2$. In contrast to this the first order system (3.11) can be equivalently transformed to the strangeness-free system

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \dot{\tilde{y}}(t) + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tilde{y}(t) = \begin{bmatrix} f_2(t) \\ f_1(t) - \dot{f}_2(t) - \ddot{f}_2(t) \\ \dot{f}_2(t) \\ \dot{f}_1(t) - \ddot{f}_2(t) - f_2^{(3)}(t) \end{bmatrix},$$

where $\tilde{y}(t) = [x_1(t), x_2(t), \dot{x}_1(t), \dot{x}_2(t)]^T$. The s-index of the first order system (3.11) is $\mu = 3$, which is larger by 1 than the s-index of the original second order system (3.9).

Example 3.2.2 shows that transforming to first order systems may increase the s-index of the system. Thus, if we are able to solve the second order system directly this may give better results than solving the corresponding first order system.

Remark 3.2.3. *This particular problem does not occur when we consider the equations of motion of multibody systems, because when transforming to first order we only introduce new variables for the derivatives of the position coordinates and not for the algebraic variables λ, which are responsible for the higher index of the system. If we would introduce new variables for $\dot{\lambda}$, the arising first order system would have d-index 4. This can be seen if we write the equations of motion as follows*

$$\begin{bmatrix} M & \\ & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \ddot{\lambda} \end{bmatrix} = \begin{bmatrix} f_a(q, \dot{q}, \lambda, t) - G^T(q,t)\lambda \\ g(q,t) \end{bmatrix}.$$

The corresponding first order system reads

$$\begin{aligned} \dot{q} &= v, \\ \dot{\lambda} &= w, \\ \begin{bmatrix} M & \\ & 0 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{w} \end{bmatrix} &= \begin{bmatrix} f_a(q, v, \lambda, t) - G^T(q,t)\lambda \\ g(q,t) \end{bmatrix}. \end{aligned}$$

This system has d-index 4, as we need one more differentiation of the constraints to obtain a representation for $\dot{w}$, than in the case of the normal first order equations of motion.

3.3 BDF Methods for the Numerical Solution of Higher Order DAEs

For the direct numerical solution of higher order differential-algebraic systems we follow the approach given by Sand [27]. In [27] an alternative variable-step version of the implicit Euler method for d'th order DAEs of d-index $\nu \in \{2, \dots, d+1\}$ is introduced and a generalization to modified variable-step variable-order BDF methods is given.

To explain this approach we consider the differential-algebraic initial value problem of order d and d-index $\nu \in \{2, \dots, d+1\}$

$$\begin{aligned} y^{(d)}(t) &= f(t, y(t), \dot{y}(t), \dots, y^{(d-1)}(t), \lambda(t)), \\ 0 &= g(t, y(t), \dot{y}(t), \dots, y^{(d+1-\nu)}(t)), \\ y^{(j)}(t_0) &= \eta_j, \quad j = 0, 1, \dots, d-1. \end{aligned} \tag{3.14}$$

Remark 3.3.1. *The equations of motion of multibody systems with order $d = 2$ and d-index $\nu = 3$ are of the form* (3.14).

Following the classical approach of transforming the higher-order system (3.14) into a first order system leads to

$$\begin{aligned} \dot{y}_0 &= y_1, \\ &\vdots \\ \dot{y}_{d-2} &= y_{d-1}, \\ \dot{y}_{d-1} &= f(t, y_0, y_1, \dots, y_{d-1}, \lambda), \\ 0 &= g(t, y_0, y_1, \dots, y_{d+1-\nu}), \end{aligned} \tag{3.15}$$

with $[y_0, y_1, \dots, y_{d-1}] := [y(t), \dot{y}(t), \dots, y^{(d-1)}(t)]$.
We want to discretize system (3.15) on an interval $[t_0, t_0+T]$ with the non-equidistant grid $t_0 < t_1 \cdots < t_N = t_0 + T$, where $t_n = t_{n-1} + h_n$ for all $n = 1, \dots, N$. Approximating the first order derivatives in (3.15) by the implicit Euler method leads to the implicit Euler approximation

$$\begin{aligned} \frac{y_{j,n} - y_{j,n-1}}{t_n - t_{n-1}} &= y_{j+1,n}, \quad j = 0, 1, \dots, d-2, \\ \frac{y_{d-1,n} - y_{d-1,n-1}}{t_n - t_{n-1}} &= f(t_n, y_{0,n}, y_{1,n}, \dots, y_{d-1,n}, \lambda_n), \\ 0 &= g(t_n, y_{0,n}, y_{1,n}, \dots, y_{d+1-\nu,n}), \end{aligned} \tag{3.16}$$

where $y_{j,0} = \eta_j$ for $j = 0, 1, \dots, d-1$.
Here, $y_{j,n}$ is an approximation to $y^{(j)}(t_n)$ for $j = 0, 1, \dots, d-1$, and λ_n is an approximation to $\lambda(t_n)$.
The problems which arise by following this idea are described above.

We now exchange the equation order reduction and the discretization. This means that the higher order initial value problem (3.14) is first discretized by divided differences and after that the discrete equations are written as a system of equations.
For the divided differences and small stepsize $h_n = t_n - t_{n-1}$ it holds that $k!y[t_n, \dots, t_{n-k}] \approx y^{(k)}(t_n)$. Using this approximation we get

$$d!y[t_n, \dots, t_{n-d}] = f(t_n, \hat{y}_{0,n}, \hat{y}_{1,n}, \dots, \hat{y}_{d-1,n}, \lambda_n) + R, \tag{3.17a}$$
$$0 = g(t_n, \hat{y}_{0,n}, \hat{y}_{1,n}, \dots, \hat{y}_{d+1-\nu,n}), \tag{3.17b}$$

where R denotes a remainder term. Here, $\hat{y}_{j,n}$ is an approximation to $j!y[t_n, t_{n-1}, \dots, t_{n-j}]$ and λ_n is an approximation to $\lambda(t_n)$.

With the relation

$$\begin{aligned} \hat{y}_{j,n} \approx j!y[t_n, \dots, t_{n-j}] &= j! \frac{y[t_n, \dots, t_{n-j+1}] - y[t_{n-1}, \dots, t_{n-j}]}{t_n - t_{n-j}} \\ &= j! \frac{(j-1)!y[t_n, \dots, t_{n-j+1}] - (j-1)!y[t_{n-1}, \dots, t_{n-j}]}{(t_n - t_{n-j})(j-1)!} \\ &\approx j! \frac{\hat{y}_{j-1,n} - \hat{y}_{j-1,n-1}}{(j-1)!(t_n - t_{n-j})} \\ &= \frac{\hat{y}_{j-1,n} - \hat{y}_{j-1,n-1}}{(t_n - t_{n-j})/j}, \end{aligned}$$

we can write (3.17a) as a system of equations and we obtain the discretization

$$\begin{aligned} \frac{\hat{y}_{j,n} - \hat{y}_{j,n-1}}{(t_n - t_{n-1-j})/(j+1)} &= \hat{y}_{j+1,n}, \quad j = 0, 1, \dots, d-2, \\ \frac{\hat{y}_{d-1,n} - \hat{y}_{d-1,n-1}}{(t_n - t_{n-d})/d} &= f(t_n, \hat{y}_{0,n}, \hat{y}_{1,n}, \dots, \hat{y}_{d-1,n}, \lambda_n), \\ 0 &= g(t_n, \hat{y}_{0,n}, \hat{y}_{1,n}, \dots, \hat{y}_{d+1-\nu,n}), \end{aligned} \tag{3.18}$$

where $\hat{y}_{j,0} = \eta_j$ for $j = 0, 1, \dots, d-1$ and $t_m = t_0$ if $m < 0$.

The difference between the approximations (3.16) and (3.18) is that the implicit Euler approximation (3.16) only regards the current stepsize $h_n = t_n - t_{n-1}$ at step n for the approximation of $y_{d,n}$, while the modified Euler approximation (3.18) considers a mean value of the current stepsize h_n and the last $d-1$ stepsizes $h_{n-1} = t_{n-1} - t_{n-2}, \dots, h_{n-d+1} = t_{n-d+1} - t_{n-d}$

$$\frac{1}{d}\sum_{i=0}^{d-1} h_{n-i} = \frac{1}{d}(t_n - t_{n-d}). \tag{3.19}$$

For constant stepsizes the approximations (3.16) and (3.18) are the same as (3.19) equals h_n.
It is interesting to notice that method (3.18) only differs from the implicit Euler approximation (3.16) in the first $d-1$ steps after a change of stepsize. If the d-index is $\nu = d+1 \geq 3$ this is just the case when the implicit Euler method (3.16) fails to converge as we have seen in Chapter 2.

Sand [27] shows that (3.18) has a unique solution under further assumptions and that method (3.18) is convergent for the case $d = \nu - 1 \geq 1$.

Assumption 3.3.2. *[27]*

1. *f and $g^{(\nu-1)}$ are C^1-functions with bounded and Lipschitz-continuous partial derivatives on open sets Ω_1, Ω_2, containing $v_0 = [t_0, \eta_0, \dots, \eta_{d-1}, \lambda_0]$ and $[t_0, \eta_0, \dots, \eta_{d-1}, f(v_0)]$, respectively, where λ_0 is the unique value of $\lambda(t_0)$ (cf. 3. below).*

2. *The initial values $\eta_0, \dots, \eta_{d-1}$ are consistent with the equations*

$$0 = g^{(i)}(t_0, \eta_0, \dots, \eta_{d+1-\nu+i}) \quad \text{for } i = 0, 1, \dots, \nu - 2.$$

3. *There exists a unique solution $\lambda(t_0) = \lambda_0$ to the equation*

$$0 = g^{(\nu-1)}(t_0, \eta_0, \dots, \eta_{d-1}, f(t_0, \eta_0, \dots, \eta_{d-1}, \lambda(t_0))),$$

or a solution $\lambda(t_0) = \lambda_0$ is given as initial value.

4. *The matrix*

$$\frac{\partial}{\partial \lambda(t)} g^{(\nu-1)}(t_0, y_0^{[2]}(t), \dots, y_{d-1}^{[2]}(t), f(t, y_0^{[1]}(t), \dots, y_{d-1}^{[1]}(t), \lambda(t)))$$

is regular with bounded inverse for all $v(t) = [t, y_0^{[1]}(t), ..., y_{d-1}^{[1]}(t), \lambda(t)] \in \Omega_1$, $[t, y_0^{[2]}(t), ..., y_{d-1}^{[2]}(t), f(v(t))] \in \Omega_2$.

Theorem 3.3.3. *Consider system (3.14) with d-index $\nu = d+1$. Assume that the unique solution of (3.14), ensured by Assumption 3.3.2, exists for $t \in [t_0, t_{N-1} + H]$, where $H < \infty$ is an upper bound of the stepsizes $t_i - t_{i-1}$, $i \geq 1$, and that the DAE-solution remains within Ω_1, Ω_2.*
If

1. *$\hat{y}_{j,0} = y^{(j)}(t_0) + O(H)(t_1 - t_0)^{d-j}$ for $j = 0, 1, \dots, d-1$,*

2. *$(t_{n+1} - t_n)/(t_n - t_{n-1}) \in [\gamma, \Gamma]$ for $0 < \gamma \leq \Gamma < \infty$, $n = 1, \dots, N-1$,*

then, for sufficiently small H, (3.18) *has a unique solution satisfying* $\lambda_n = \lambda(t_n) + O(H)$ *for all* t_n*,* $n = 1, 2, \ldots, N$*, and*

$$\begin{aligned}&\|\hat{y}_{j,n} - j!y[t_n, t_{n-1}, \ldots, t_{n-j}]\| = \\ &O(H)(H + t_n - t_0)^{d-j}[1 + K(H + t_n - t_0)exp((K + O(H))(t_n - t_0))],\end{aligned}$$

for $j = 0, 1, \ldots, d-1$*. The constant* $K = d + L_f(1 + ML_g(1 + L_f))$ *depends on the bounds* L_f, L_g *of the partial derivatives of* f *and* $g^{(d)}$ *and on the bound* M *of* $[\partial g^{(d)}/\partial\lambda(t)]^{-1}$ *on* Ω_1, Ω_2 *(cf. Assumption 3.3.2).*
The error bounds for $\hat{y}_{j,n}$*,* $j = 0, 1, \ldots, d-1$*, are also valid if the algebraic constraint is replaced by*

$$g(t_n, \hat{y}_{0,n}) = O(H)\prod_{j=1}^{d}(t_n - t_{n-j}).$$

Proof. See [27], Theorem 3. □

Method (3.18) can be generalized to variable-step variable-order methods based on BDF methods. To derive the variable-step variable-order methods given in [27] we follow the idea that if we apply a BDF method of order k for the estimation of $y^{(j+1)}(t_n)$, then the result should be exact if y is a polynomial of degree at most $k+j$. For $j = 0$ this is achieved by the ordinary variable stepsize BDF method of order k

$$\sum_{i=1}^{k}\prod_{m=1}^{i-1}(t_n - t_{n-m})\hat{y}_0[t_n, t_{n-1}, \ldots, t_{n-i}] = \hat{y}_{1,n}. \tag{3.20}$$

For $j \geq 1$ the BDF method of order k has to be modified in order to produce exact values of $y^{(j+1)}(t_n)$ for polynomials y of degree $k+j$ for $j = 1, 2, \ldots, d-1$. This modified BDF method of order k is given by

$$\sum_{i=1}^{k}\alpha_{i,n}^{[j+1]}\hat{y}_j[t_n, t_{n-1}, \ldots, t_{n-i}] = \hat{y}_{j+1,n}, \quad j = 1, \ldots, d-1, \tag{3.21}$$

where the coefficients $\alpha_{i,n}^{[j+1]}$ are the modified BDF-coefficients for the approximation of $\hat{y}_{j+1,n}$. The coefficients $\alpha_{i,n}^{[j+1]}$ have to satisfy the conditions

$$\sum_{i=1}^{k}\alpha_{i,n}^{[j+1]}\tilde{y}_j[t_n, t_{n-1}, \ldots, t_{n-i}] = \frac{(s+j)!}{(s-1)!}t_n^{s-1} \tag{3.22}$$

for $\tilde{y}_0(t) = t^{s+j}$ and $s = 1, 2, \ldots, k$, $j = 1, 2, \ldots, d-1$, in order to obtain exact solutions for the polynomials $\tilde{y}_0(t) = t^2, \ldots, \tilde{y}_0(t) = t^{k+j}$.

In the case that $\hat{y}_{j,n}$,...,$\hat{y}_{j,n-k}$ have been produced by such a modified BDF method of order k or by formulas of at least this order of accuracy, then the coefficients $\alpha_{i,n}^{[j+1]}$ may be easily obtained from ordinary BDF-coefficients of order k as

$$\alpha_{i,n}^{[j+1]} = \prod_{m=1}^{i-1}(t_n - t_{n-m}), \quad i = 1, 2, \ldots, k-1, \tag{3.23}$$

and for $\tilde{y}_0(t) = t^{k+j}$ they are given by

$$\alpha_{k,n}^{[j+1]} = \left\{ \frac{(k+j)!}{(k-1)!} t_n^{k-1} - \sum_{i=1}^{k-1} \prod_{m=1}^{i-1} (t_n - t_{n-m}) \tilde{y}_j[t_n, \ldots, t_{n-i}] \right\} / \tilde{y}_j[t_n, \ldots, t_{n-k}].$$

3.4 BDF Methods for the Numerical Solution of Second Order DAEs

Now we return to the case of second order DAEs with d-index $\nu = 3$, as they arise in the equations of motion of multibody systems.
The initial value problem (3.14) for $(d, \nu) = (2, 3)$ becomes

$$\begin{aligned} \ddot{y}(t) &= f(t, y(t), \dot{y}(t), \lambda(t)), \\ 0 &= g(t, y(t)), \end{aligned} \tag{3.24}$$

with initial values $y^{(0)}(t_0) = \eta_0$, $y^{(1)}(t_0) = \eta_1$.
The modified BDF method (3.21) reads in this case

$$\sum_{i=1}^{k} \alpha_{i,n}^{[2]} \hat{y}_1[t_n, t_{n-1}, \ldots, t_{n-i}] = \hat{y}_{2,n}, \tag{3.25}$$

where the coefficients $\alpha_{i,n}^{[2]}$ have to satisfy

$$\sum_{i=1}^{k} \alpha_{i,n}^{[2]} \tilde{y}_1[t_n, t_{n-1}, \ldots, t_{n-i}] = \frac{(s+1)!}{(s-1)!} t_n^{s-1} \tag{3.26}$$

for $\tilde{y}_0(t) = t^{s+1}$ and $s = 1, 2, \ldots, k$.
The first order derivatives are again approximated by the ordinary BDF method of order k

$$\hat{y}_{1,n} = \sum_{i=1}^{k} \prod_{m=1}^{i-1} (t_n - t_{n-m}) \hat{y}_0[t_n, t_{n-1}, \ldots, t_{n-i}]. \tag{3.27}$$

Altogether a variable-step variable-order BDF discretization of system (3.24) is given by

$$\begin{aligned} \sum_{i=1}^{k} \alpha_{i,n}^{[2]} \hat{y}_1[t_n, t_{n-1}, \ldots, t_{n-i}] &= f(t_n, \hat{y}_{0,n}, \hat{y}_{1,n}, \lambda_n), \\ 0 &= g(t_n, \hat{y}_{0,n}), \\ \hat{y}_{1,n} &= \sum_{i=1}^{k} \prod_{m=1}^{i-1} (t_n - t_{n-m}) \hat{y}_0[t_n, t_{n-1}, \ldots, t_{n-i}], \end{aligned} \tag{3.28}$$

with the unknowns $\hat{y}_{1,n}$, $\hat{y}_{0,n}$ and λ_n in step n at time t_n and the divided differences

$$\begin{aligned} \hat{y}_0[t_n, t_{n-1}, \ldots, t_{n-i}] &= \sum_{j=0}^{i} \alpha_j (\hat{y}_{0,n-j+1} - \hat{y}_{0,n-j}), \\ \hat{y}_1[t_n, t_{n-1} \ldots, t_{n-i}] &= \sum_{j=0}^{i} \hat{\alpha}_j (\hat{y}_{1,n-j+1} - \hat{y}_{1,n-j}), \end{aligned} \tag{3.29}$$

depending on the current unknowns $\hat{y}_{1,n}$, $\hat{y}_{0,n}$ and on previous values and coefficients α_j, $\hat{\alpha}_j$ depending on $t_n, \ldots, t_{n-i}$.

In the following we want to use these modified BDF methods (up to order $k = 2$) for the numerical solution of the equations of motion of multibody systems. We have to distinguish different formulas for order $k = 1, 2$ depending on the accuracies of the last taken steps. This means that we have to take into account the accuracies of $\hat{y}_{1,n-1}$ and $\hat{y}_{1,n-2}$, as these values are used in the calculation of the coefficients $\alpha_{i,n}^{[2]}$. We say that $\hat{y}_{1,m}$ has the accuracy k_m for $m \in \mathbb{N}$, if $\hat{y}_{1,m}$ was produced by a method of order k_m. In the following BDF(k_{n-2},k_{n-1},k_n) denotes a modified BDF method (3.25) of order k_n, where the previous values $\hat{y}_{1,n-2}$ and $\hat{y}_{1,n-1}$ were produced by methods of order k_{n-2} and k_{n-1}.
For the modified BDF method (3.25) of order $k_n = 1$ we get two different formulas.

BDF(1,1)	$\hat{y}_{1,n-1}$ has the accuracy $k_{n-1} = 1$; used for proceeding a solution of order 1,
BDF(2,1)	$\hat{y}_{1,n-1}$ has the accuracy $k_{n-1} = 2$, i.e., for $y_0(t) = t^2$ the value $\hat{y}_{1,n-1}$ is exact; used when changing from methods of order 2 to methods of order 1 and in the first step.

For the modified BDF methods of order $k_n = 2$ we have to take into account the last two steps. Therefore, we get five different formulas for order $k_n = 2$.

BDF(3,1,2)	$\hat{y}_{1,n-1}$ has the accuracy $k_{n-1} = 1$ and $\hat{y}_{1,n-2}$ has the accuracy $k_{n-2} = 3$, i.e., $\hat{y}_{1,n-2}$ is exact for $y_0(t) = t^2$; used when changing from order 1 to order 2, when the step before has order 3.
BDF(1,2,2)	$\hat{y}_{1,n-1}$ has the accuracy $k_{n-1} = 2$ and $\hat{y}_{1,n-2}$ has the accuracy $k_{n-2} = 1$; used for proceeding methods of order 2 when the step before has order 1,
BDF(2,2,2)	$\hat{y}_{1,n-1}$ has the accuracy $k_{n-1} = 2$ and $\hat{y}_{1,n-2}$ has the accuracy $k_{n-2} = 2$; used for proceeding solutions of order 2,
BDF(1,1,2)	$\hat{y}_{1,n-1}$ has the accuracy $k_{n-1} = 1$ and $\hat{y}_{1,n-2}$ has the accuracy $k_{n-2} = 1$; used for changing from order 1 to order 2,
BDF(2,1,2)	$\hat{y}_{1,n-1}$ has the accuracy $k_{n-1} = 1$ and $\hat{y}_{1,n-2}$ has the accuracy $k_{n-2} = 2$; used for changing from order 1 to order 2 and the step before has order 2.

The coefficients $\alpha_{i,n}^{[2]}$ for these methods are given in Table 3.3. Now we resume with Example 2.4.7 in order to show the advantages of the modified methods.

Example 3.4.1. *Consider again the second order system* (2.55) *given in Example 2.4.7. First, we use the modified BDF method of order* $k = 1$ *with variable stepsizes to solve the second order system. The absolute errors of the algebraic variable* λ *are given in Table 3.1 in comparison with the results obtained by the implicit Euler method, cf. Table 2.3. In Table 3.2 the results for the modified BDF methods of order* $k = 1$ *and* $k = 2$ *are given, in comparison to the results obtained by the ordinary BDF methods of order* 1 *and* 2*. The results are errors in the algebraic variable* λ *for the stepsizes* $h = 0.005$ *and* $h = 0.01$*. We have used second-order formulas except for the first step.*

Step no.	Stepsize $\times 10^{-3}$	$\lvert\lambda(t_j) - \lambda_j\rvert$ (Euler)	(mod. Euler)
1	1.000	**2.0080**	0.0080
2	1.000	0.0080	0.0120
3	0.200	**8.0303**	0.0057
4	0.040	**8.0348**	0.0012
5	0.008	**8.0357**	0.0003
6	0.008	0.0001	0.0001
7	0.016	**1.0047**	0.0002
8	0.032	**1.0048**	0.0004
9	0.064	**1.0052**	0.0007
10	0.064	0.0006	0.0008

Table 3.1: Variable stepsize modified Euler method

	$\lvert\lambda(t_j) - \lambda_j\rvert$, $h = 0.005$		$\lvert\lambda(t_j) - \lambda_j\rvert$, $h = 0.01$	
t_n	(BDF1&2)	(mod.BDF1&2)	(BDF1&2)	(mod.BDF1&2)
1.0050	**2.0400**	0.0402		
1.0100	**4.0190**	0.0010	**2.0810**	0.0809
1.0150	**1.0120**	0.0010		
1.0200	0.0012	0.0009	**4.0350**	0.0041
1.0300	0.0013	0.0009	**1.0280**	0.0041
1.0400	0.0013	0.0009	0.0052	0.0038
1.0500	0.0014	0.0010	0.0054	0.0038

Table 3.2: Modified BDF method of order $k = 1, 2$

In Example 3.4.1 we can see that the modified BDF methods improve the results obtained by the ordinary BDF methods. The high errors which occur whenever the stepsize changes (displayed in bold face), when using ordinary BDF methods do not occur in the modified methods. Similar results are obtained when the order of the method is changed, as the results in Table 3.2 show. When using ordinary BDF methods, large errors occur when the order is changed after the first step, but changing the order of the modified BDF methods does not effect the error. Thus, we obtain better results using the modified BDF methods with variable order and variable stepsizes to solve the second order system, than using ordinary BDF methods to solve the corresponding first order system for higher index DAEs.

method	k	$\alpha_{1,n}^{[2]}$	$\alpha_{2,n}^{[2]}$
BDF(2,1)	1	2	
BDF(1,1)	1	$\frac{2(t_n-t_{n-1})}{t_n-t_{n-2}}$	
BDF(3,1,2)	2	$\frac{2(t_{n-2}-t_n)}{2t_n-t_{n-1}-t_{n-2}}$	$2(2t_n-t_{n-1}-t_{n-2})$
BDF(1,2,2)	2	$\frac{2(3t_n-t_{n-1}-t_{n-2}-t_{n-3})(t_n-t_{n-1})(t_{n-2}-t_{n-3})}{(2t_n-t_{n-1}-t_{n-2})(t_n-t_{n-3})(2t_{n-1}-t_{n-2}-t_{n-3})}-\frac{2(t_{n-1}-t_{n-2})}{2t_{n-1}-t_{n-2}-t_{n-3}}$	$\frac{2(t_n-t_{n-2})(t_{n-1}-t_{n-2})[2(t_n-t_{n-1})(2t_n-t_{n-1}-t_{n-2})-(t_n-t_{n-3})(t_{n-1}-t_{n-2})]}{(2t_{n-1}-t_{n-2}-t_{n-3})(2t_n-t_{n-1}-t_{n-2})(t_n-t_{n-3})}$
BDF(2,2,2)	2	1	$\frac{(t_n-t_{n-2})(t_{n-1}-t_{n-2})[2(2t_n-t_{n-1}-t_{n-2})(t_n-t_{n-1})-(t_{n-1}-t_{n-2})(t_n-t_{n-3})]}{(2t_n-t_{n-1}-t_{n-2})(t_n-t_{n-3})(t_{n-1}-t_{n-2})-(t_n-t_{n-1})(t_{n-1}-t_{n-4})(t_{n-2}-t_{n-3})}$
BDF(1,1,2)	2	$\frac{2(3t_n-t_{n-1}-t_{n-2}-t_{n-3})(t_n-t_{n-1})}{(2t_n-t_{n-1}-t_{n-2})(t_n-t_{n-3})}-\frac{2(2t_n-t_{n-1}-t_{n-2})(t_{n-1}-t_{n-2})}{(t_{n-1}-t_{n-3})(t_n-t_{n-3})}$	$\frac{2(2t_n-t_{n-1}-t_{n-2})(t_n-t_{n-2})(t_{n-1}-t_{n-2})}{(t_{n-1}-t_{n-3})(t_n-t_{n-3})}$
BDF(2,1,2)	2	$\frac{2(t_{n-1}-t_{n-2})(t_{n-2}-t_n)}{(t_n-t_{n-3})(t_{n-1}-t_{n-2})-(t_{n-1}-t_{n-4})(t_{n-2}-t_{n-3})}+\frac{2(t_n-t_{n-1})}{2t_n-t_{n-1}-t_{n-2}}$	$\frac{2(2t_n-t_{n-1}-t_{n-2})(t_n-t_{n-2})(t_{n-1}-t_{n-2})}{(t_n-t_{n-3})(t_{n-1}-t_{n-2})-(t_{n-1}-t_{n-4})(t_{n-2}-t_{n-3})}$

Table 3.3: BDF coefficients of modified formulas

Chapter 4

Implementation of the Modified BDF Methods for Second Order Systems

In this chapter we describe the variable-step variable-order implementation of the modified BDF methods of order $k = 1, 2$ as introduced in Chapter 3 for the numerical solution of the equations of motion of multibody systems. We describe the discretization of the equations of motion as derived in Chapter 1 and explain the strategies to solve the nonlinear systems arising in each integration step by Newton's method and the strategies for the stepsize and order selection.

4.1 Discretization

We consider the complex form of the equations of motion (1.18) in Section 1.2.3:

$$\begin{aligned} M(q)\ddot{q} &= f_a(q,\dot{q},r,w,s,\lambda,t) - G^T(q,s,t)\lambda && \in \mathbb{R}^{n_q},\\ \dot{r} &= f_d(q,\dot{q},r,w,s,\lambda,t) && \in \mathbb{R}^{n_r},\\ 0 &= d(q,\dot{q},r,w,s,\lambda,t) && \in \mathbb{R}^{n_w},\\ 0 &= h(q,s,t) && \in \mathbb{R}^{n_s},\\ 0 &= g(q,s,t) && \in \mathbb{R}^{n_\lambda}, \end{aligned} \tag{4.1}$$

where $M(q)$ is positive definite and $G(q,s,t) = [\frac{\partial g}{\partial q} - \frac{\partial g}{\partial s}(\frac{\partial h}{\partial s})^{-1}\frac{\partial h}{\partial q}](q,s,t)$.

We can determine the d-index of the second order system of the equations of motion using Definition 3.1.1. For system (4.1) the corresponding first order system is given by

$$\begin{aligned}
\dot{q} &= v, \\
M(q)\dot{v} &= f_a(q, v, r, w, s, \lambda, t) - G^T(q, s, t)\lambda, \\
\dot{r} &= f_d(q, v, r, w, s, \lambda, t), \\
0 &= d(q, v, r, w, s, \lambda, t), \\
0 &= h(q, s, t), \\
0 &= g(q, s, t).
\end{aligned} \tag{4.2}$$

Lemma 4.1.1. *The first order equations of motion* (4.2) *form a system of d-index 3 as long as*

$$GM^{-1}\left[-\frac{\partial f_a}{\partial \lambda} + \frac{\partial f_a}{\partial w}\left(\frac{\partial d}{\partial w}\right)^{-1}\frac{\partial d}{\partial \lambda} + G^T\right](q, v, r, s, \lambda, t) \tag{4.3}$$

is non-singular.

Proof. See [3], Lemma 2.4.4. □

Thus, according to Definition 3.1.1 system (4.1) has d-index 3 under the assumptions (4.3).

For the discretization of system (4.1) on the interval $[t_0, t_0 + T]$ we define a non-equidistant grid $t_0 < t_1 < \cdots < t_N = t_0 + T$ with grid points $t_n = t_{n-1} + h_n$. We denote by $y_{i,n}$ an approximation to $y^{(i)}(t_n)$. A discretization of system (4.1) is then given by

$$\begin{aligned}
M(q_{0,n})q_{2,n} &= f_a(q_{0,n}, q_{1,n}, r_{0,n}, w_{0,n}, s_{0,n}, \lambda_{0,n}, t_n) - G^T(q_{0,n}, s_{0,n}, t_n)\lambda_{0,n}, \\
r_{1,n} &= f_d(q_{0,n}, q_{1,n}, r_{0,n}, w_{0,n}, s_{0,n}, \lambda_{0,n}, t_n), \\
0 &= d(q_{0,n}, q_{1,n}, r_{0,n}, w_{0,n}, s_{0,n}, \lambda_{0,n}, t_n), \\
0 &= h(q_{0,n}, s_{0,n}, t_n), \\
0 &= g(q_{0,n}, s_{0,n}, t_n).
\end{aligned} \tag{4.4}$$

Using the modified BDF method (3.25) for the approximation of $q_{2,n}$ and the ordinary BDF method (3.27) for the approximation of $q_{1,n}$ and $r_{1,n}$ we get the following approximation

$$\begin{aligned}
M(q_{0,n})\sum_{i=1}^{k}\alpha_{i,n}^{[2]}q_1[t_n, \ldots, t_{n-i}] &= f_a(q_{0,n}, q_{1,n}, r_{0,n}, w_{0,n}, s_{0,n}, \lambda_{0,n}, t_n) \\
&\quad - G^T(q_{0,n}, s_{0,n}, t_n)\lambda_{0,n}, \\
\sum_{i=1}^{k}\prod_{m=1}^{i-1}(t_n - t_{n-m})r_0[t_n, \ldots, t_{n-i}] &= f_d(q_{0,n}, q_{1,n}, r_{0,n}, w_{0,n}, s_{0,n}, \lambda_{0,n}, t_n), \\
0 &= d(q_{0,n}, q_{1,n}, r_{0,n}, w_{0,n}, s_{0,n}, \lambda_{0,n}, t_n), \\
0 &= h(q_{0,n}, s_{0,n}, t_n), \\
0 &= g(q_{0,n}, s_{0,n}, t_n), \\
q_{1,n} &= \sum_{i=1}^{k}\prod_{m=1}^{i-1}(t_n - t_{n-m})q_0[t_n, \ldots, t_{n-i}]
\end{aligned} \tag{4.5}$$

for given initial values $y_{0,0}$ and $y_{1,0}$, where $y(t) = [q(t), r(t), w(t), s(t), \lambda(t), \dot{q}(t)]^T$. System (4.5) consists of 6 equations of the dimensions n_q, n_r, n_w, n_s, n_λ and n_q in 6 unknown $q_{0,n}$, $r_{0,n}$, $w_{0,n}$, $s_{0,n}$, $\lambda_{0,n}$ and $q_{1,n}$.

Thus, in every step n of the integration procedure we get a nonlinear system of $N = 2n_q + n_r + n_w + n_s + n_\lambda$ equations with N unknown values $x_n = [q_{0,n}, r_{0,n}, w_{0,n}, s_{0,n}, \lambda_{0,n}, q_{1,n}]^T$ at time t_n. We can write this nonlinear system as $F(x_n) = 0$ with

$$F(x_n) := \begin{bmatrix} M \sum_{i=1}^{k} \alpha_{i,n}^{[2]} q_1[t_n, \dots, t_{n-i}] - f_a + G^T \lambda_{0,n} \\ \sum_{i=1}^{k} \prod_{m=1}^{i-1} (t_n - t_{n-m}) r_0[t_n, \dots, t_{n-i}] - f_d \\ d \\ h \\ g \\ q_{1,n} - \sum_{i=1}^{k} \prod_{m=1}^{i-1} (t_n - t_{n-m}) q_0[t_n, \dots, t_{n-i}] \end{bmatrix}. \tag{4.6}$$

For better reading we omit the arguments of the functions in (4.6).

4.2 Nonlinear System Solution

The nonlinear system which arise in Section 4.1 must be solved for x_n at each time step. This is done by means of Newton's method. Consider the nonlinear system

$$F(x_n) = 0, \tag{4.7}$$

with $F : \mathbb{R}^N \to \mathbb{R}^N$.
For system (4.7) the Newton method is given by the iteration

$$x_n^{(m+1)} = x_n^{(m)} - DF(x_n^{(m)})^{-1} F(x_n^{(m)}), \tag{4.8}$$

with a given starting value $x_n^{(0)}$. This can be written in the equivalent form

$$DF(x_n^{(m)}) (\underbrace{x_n^{(m+1)} - x_n^{(m)}}_{:=\delta^{(m+1)}}) = -F(x_n^{(m)}). \tag{4.9}$$

The new iterate $x_n^{(m+1)}$ is then given by

$$x_n^{(m+1)} = x_n^{(m)} + \delta^{(m+1)}. \tag{4.10}$$

This method requires the evaluation of the Jacobian $DF(x_n)$ and the solution of the linear system (4.9) in each step.

For system (4.7) with $F(x_n)$ given by (4.6) the Jacobian $DF(x_n)$ is given by

$$\begin{bmatrix} A_{11} & -f_{a,r_0} & -f_{a,w_0} & -f_{a,s_0} + G_{s_0}^T \lambda_0 & -f_{a,\lambda_0} + G^T & M\alpha_1 - f_{a,q_1} \\ -f_{d,q_0} & A_{22} & -f_{d,w_0} & -f_{d,s_0} & -f_{d,\lambda_0} & -f_{d,q_1} \\ d_{q_0} & d_{,r_0} & d_{w_0} & d_{s_0} & d_{\lambda_0} & d_{q_1} \\ h_{q_0} & 0 & 0 & h_{s_0} & 0 & 0 \\ g_{q_0} & 0 & 0 & g_{s_0} & 0 & 0 \\ A_{61} & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where the terms A_{11}, A_{22} and A_{61} are given by

$$\begin{aligned}
A_{11} &:= M_{q_0} \sum_{i=1}^{k} \alpha_i (q_{1,n-i+1} - q_{1,n-i}) - f_{a,q_0} + G_{q_0}^T \lambda_0, \\
A_{22} &:= \sum_{i=1}^{k} \frac{1}{t_n - t_{n-i}} - f_{d,r_0}, \\
A_{61} &:= -\sum_{i=1}^{k} \frac{1}{t_n - t_{n-i}}.
\end{aligned}$$

Here, we omit the arguments of the functions and the index n and for a function f, f_x denotes the partial derivative of f with respect to x, i.e., $f_x(.) := \frac{\partial}{\partial x} f(.)$.

The initial guess $x_n^{(0)}$ for the Newton method is formed by evaluating the predictor polynomial at t_n. The predictor polynomial $w_n^P(t)$ is the polynomial which interpolates x_{n-1-i} at the last $k+1$ times

$$w_n^P(t_{n-1-i}) = x_{n-1-i}, \quad i = 0, \dots, k \tag{4.11}$$

and is given in terms of the divided differences by

$$w_n^P(t) = \sum_{i=0}^{k} \prod_{j=0}^{i-1} (t - t_{n-1-j}) x[t_{n-1}, \dots, t_{n-1-i}]. \tag{4.12}$$

The predicted value $x_n^{(0)}$ is obtained by evaluating $w_n^P(t)$ at t_n

$$x_n^{(0)} = w_n^P(t_n). \tag{4.13}$$

In the following we describe the strategies used to solve the linear system

$$DF(x_n^{(m)}) \delta^{(m+1)} = -F(x_n^{(m)}). \tag{4.14}$$

We can write $DF(x_n^{(m)})$ simplified as

$$DF(x_n^{(m)}) = \begin{bmatrix} A & Z \\ B & I \end{bmatrix}, \tag{4.15}$$

with matrices

$$A = \begin{bmatrix} A_{11} & -f_{a,r_0} & -f_{a,w_0} & -f_{a,s_0} + G_{s_0}^T \lambda_0 & -f_{a,\lambda_0} + G^T \\ -f_{d,q_0} & A_{22} & -f_{d,w_0} & -f_{d,s_0} & -f_{d,\lambda_0} \\ d_{q_0} & d_{,r_0} & d_{w_0} & d_{s_0} & d_{\lambda_0} \\ h_{q_0} & 0 & 0 & h_{s_0} & 0 \\ g_{q_0} & 0 & 0 & g_{s_0} & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} A_{61} & 0 & 0 & 0 & 0 \end{bmatrix}, \; Z = \begin{bmatrix} M\alpha_1 - f_{a,q_1} \\ -f_{d,q_1} \\ d_{q_1} \\ 0 \\ 0 \end{bmatrix} \text{ and the identity matrix } I.$$

The matrix B can be eliminated using a block Gauss-Elimination step. In detail this is done by the transformation

$$\underbrace{\begin{bmatrix} A & Z \\ B & I \end{bmatrix}\begin{bmatrix} I & 0 \\ -B & I \end{bmatrix}}_{\begin{bmatrix} \tilde{A} & Z \\ 0 & I \end{bmatrix}}\underbrace{\begin{bmatrix} I & 0 \\ -B & I \end{bmatrix}^{-1}\delta^{(m+1)}}_{\begin{bmatrix} d_1^{(m+1)} \\ d_2^{(m+1)} \end{bmatrix}} = -F, \quad = -\begin{bmatrix} F_1 \\ F_2 \end{bmatrix}, \tag{4.16}$$

with $\tilde{A} = A - ZB$. Next, the matrix $\tilde{A}$ is factored into a product of an upper and a lower triangular matrix and a permutation matrix, $\tilde{A} = P^{-1}LU$, using LU decomposition with pivoting. System (4.16) is then solved by the following steps. The second row in (4.16) yields $d_2^{(m+1)} = -F_2$ and solving $Ud_1^{(m+1)} = y$ using backward substitution yields $d_1^{(m+1)}$. Here, y is obtained by solving $Ly = f$ by forward substitution, where $f := P(-F_1 - Zd_2^{(m+1)})$. Thus, we have $d^{(m+1)} = \begin{bmatrix} d_1^{(m+1)} \\ d_2^{(m+1)} \end{bmatrix}$, which yields

$$\delta^{(m+1)} = \begin{bmatrix} I & 0 \\ -B & I \end{bmatrix} d^{(m+1)} \tag{4.17}$$

as solution of the linear system (4.14). The new iterate of the Newton method is then given by $x_n^{(m+1)} = x_n^{(m)} + \delta^{(m+1)}$.

For the termination of the Newton iteration we use the same strategy that is used in the code DASSL and is described in [4]. The Newton iteration is continued until

$$\frac{\rho}{1-\rho}\|x_n^{(m+1)} - x_n^{(m)}\|_{tol} < 0.33, \tag{4.18}$$

where ρ is an estimate of the rate of convergence of the iteration and the norm $\|.\|_{tol}$ is a weighted root mean square norm, where the weights depend on the relative and absolute error tolerances. Precisely, the norm is defined by

$$\|v\|_{tol} = \sqrt{(1/N)\sum_{i=1}^{N}(v_i/WT_i)^2}, \tag{4.19}$$

where

$$WT_i = RTOL_i|v_i| + ATOL_i, \tag{4.20}$$

for $v \in \mathbb{R}^N$. Here, $RTOL$ and $ATOL$ are the desired relative and absolute error tolerances prescribed by the user.

The rate of convergence is estimated whenever two or more iterations have been taken by

$$\rho = \left(\frac{\|x_n^{(m+1)} - x_n^{(m)}\|_{tol}}{\|x_n^{(1)} - x_n^{(0)}\|_{tol}}\right)^{1/m}. \tag{4.21}$$

The reason for choosing condition (4.18) for terminating the iteration is that the iteration error given by $\|x_n^\star - x_n^{(m+1)}\|_{tol}$, will be sufficiently small. Here, $x_n^\star$ is

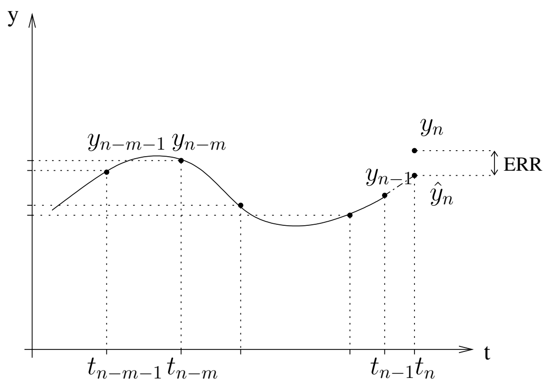

Figure 4.1: Error Estimate by Extrapolation

the solution to equation (4.7). Whenever $\rho > 0.9$ the iteration is considered to have failed. If the condition (4.18) cannot be satisfied within a maximal number of iterations the Newton iteration is considered to be divergent and the step is rejected. In this implementation a maximum of 4 iterations is permitted.

4.3 Error Estimation

To decide if a step is to be accepted or rejected we need an estimate of the error made in each integration step. For this error estimate we use an extrapolation method. The idea is to interpolate the calculated values at times $t_{n-1-m}, \ldots, t_{n-2}, t_{n-1}$ using an interpolation polynomial and extrapolate this polynomial to t_n. The error is then estimated by the difference between the extrapolated value $\hat{y}_n$ and the solution of the Newton iteration y_n, see Figure 4.1.
In detail we consider given calculated values $y_{n-1-m}, y_{n-m}, \ldots, y_{n-1}$. We interpolate these values by an interpolation polynomial $p(t)$ with

$$\begin{aligned} p(t_{n-1-i}) &= y_{n-1-i}, \quad i = 0, \ldots, m, \\ p(t) &= \sum_{i=0}^{m} \prod_{j=0}^{i-1} (t - t_{n-1-j}) y[t_{n-1}, \ldots, t_{n-1-i}]. \end{aligned} \tag{4.22}$$

Extrapolation to t_n gives

$$\hat{y}_n = p(t_n) = \sum_{i=0}^{m} \prod_{j=0}^{i-1} (t_n - t_{n-1-j}) y[t_{n-1}, \ldots, t_{n-1-i}]. \tag{4.23}$$

Thus, the error in step n is estimated by

$$ERR = \|\hat{y}_n - y_n\|_{tol}. \tag{4.24}$$

We choose $m = 3$ if more than four steps have been taken.

4.4 Stepsize and Order Selection

For the stepsize and order selection we choose a similar strategy as used in the code DASSL [4]. The most important aspect is to decide when to accept or reject a step and the selection of the stepsize and order for the next step based on the behavior of the solution. As we have only implemented methods of order $k = 1$ and 2 we use rather simple strategies for raising or lowering the order.
We start with a predefined series of methods in the first four steps. In this initial phase we use the methods BDF(2,1), BDF(3,1,2), BDF(1,2,2) and BDF(2,2,2) successively with a constant stepsize. After that we require at every step that the local error estimate ERR satisfies the condition

$$ERR = \|\hat{y}_n - y_n\|_{tol} \leq 1.0, \tag{4.25}$$

in order to decide if the step is to be accepted or rejected. If condition (4.25) is not satisfied, then the step is rejected.
After the step has been accepted or rejected the code must decide what stepsize is to be used on the next step. We choose the new stepsize $h_{n+1} = rh_n$, so that the error is roughly one half of the desired integration error tolerance. Thus, r is given by

$$r = (2.0EST)^{-1/(k+1)}, \tag{4.26}$$

where EST is given by

$$EST = ERR. \tag{4.27}$$

More precisely, if the step was successful, the stepsize to be used in the next step is chosen by the following strategies. In a consecutive order of successful steps the stepsize is doubled and the order is increased in each step. As $k = 2$ is the maximal order, the order is just increased if the current order is $k = 1$. If the previous step was unsuccessful, i.e., in the first successful step after a failure, we choose the new stepsize as $h_{n+1} = rh_n$, where r is given by (4.26). In the case that $r \geq 2.0$ the stepsize is again doubled $h_{n+1} = 2h_n$. Otherwise, if $1.0 < r < 2.0$ then the stepsize is maintained and if $r \leq 1.0$, then the stepsize is decreased by at least a factor $r = 0.9$ and at most $r = 0.5$. The order is not changed in this case.
If the step has been rejected, i.e., if condition (4.25) was not satisfied, we use the following strategies, where we distinguish two cases. In the first case the Newton iteration has converged and the failure was due to the error test. If it is the first error test failure since the last successful step, then r is determined by (4.26) and multiplied by 0.9

$$r = 0.9(2.0EST)^{-\frac{1}{k+1}}. \tag{4.28}$$

The stepsize is reduced by at least $r = 0.9$ and at most $r = 0.25$ and the order is maintained. The code keeps a count of the number of integration error test failures since the last successful step. After the second error test failure the stepsize is reduced by a factor $r = 0.25$ and the order is maintained. After three consecutive error test failures, the order is reduced to one, and the stepsize is reduced by a factor of $r = 0.25$ on every failure thereafter. In the other case if the Newton iteration failed to converge the stepsize is reduced by a factor $r = 0.25$ and the order is maintained.

When the stepsize is so small that $t + h \approx t$ the code returns with an error message. This minimum stepsize is computed as

$$h_{min} = 4\varepsilon max(|t_n|, |TOUT|), \tag{4.29}$$

where ε is the unit roundoff error, t_n is the current mesh point and $TOUT$ is the user requested output point.
For the initial stepsize we use, if not supplied by the user, the following value

$$h_0 = sign(TOUT - t_0)min(10^{-3}|TOUT - t_0|, \frac{1}{2}\|y_{1,0}\|^{-1}). \tag{4.30}$$

4.5 Interpolation between Mesh Points

For output purposes we employ an interpolant to compute the solution between mesh points. After a successful step we use the polynomial

$$w^I_{n+1}(t) = \sum_{i=0}^{k} \prod_{j=0}^{i-1} (t - t_{n+1-j}) y[t_{n+1}, \ldots, t_{n+1-i}] \tag{4.31}$$

to interpolate at times t between t_n and t_{n+1}. Here, we choose $k = 5$ if more than five steps have been taken and $k = n$, where n is the number of steps taken so far, otherwise.

Chapter 5

Runge-Kutta Methods for First Order Higher Index DAEs

Besides BDF methods there are other frequently used methods for the numerical solution of higher index DAEs. In this chapter we discuss Runge-Kutta methods for the numerical solution of first order higher index differential-algebraic systems.
First, we introduce the Runge-Kutta methods for ordinary differential equations, before extending this approach to DAEs. We present convergence results for semi-explicit DAEs of d-index one, two and three.

5.1 Runge-Kutta Methods for ODEs

Consider the initial value problem

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = \hat{y}_0, \tag{5.1}$$

with a function $f : [t_0, t_0 + T] \times \mathbb{R}^n \to \mathbb{R}^n$, $y : [t_0, t_0 + T] \to \mathbb{R}^n$ and $\hat{y}_0 \in \mathbb{R}^n$.
In the following we can restrict to autonomous systems of the form

$$\dot{y}(t) = f(y(t)), \quad y(t_0) = \hat{y}_0, \tag{5.2}$$

with a function $f : \mathbb{R}^n \to \mathbb{R}^n$, because each non-autonomous system (5.1) can be transfered by the transformation

$$z(t) = \begin{bmatrix} y(t) \\ t \end{bmatrix} \tag{5.3}$$

into an autonomous system

$$\dot{z}(t) = \begin{bmatrix} f(t, y(t)) \\ 1 \end{bmatrix} = g(z), \quad z(t_0) = \begin{bmatrix} y(t_0) \\ t_0 \end{bmatrix}. \tag{5.4}$$

Runge-Kutta methods are one-step methods in contrast to the multistep BDF methods considered in Chapter 2. A one-step method is designed to construct an approximation y_{n+1} of the solution at $t_{n+1} = t_n + h_{n+1}$ only from an approximation y_n of the solution at t_n. Here, h_{n+1} is the current stepsize at step $n+1$. It can change during the integration and does not have to be constant. In the remainder of this chapter we omit the subscript and write only h for the current stepsize.

An s-stage Runge-Kutta method applied to the ODE (5.2) is given by

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i \dot{Y}_{ni}, \quad y_0 = \hat{y}_0, \tag{5.5}$$

where $\dot{Y}_{ni}$ is defined by $\dot{Y}_{ni} = f(Y_{ni})$ with internal stages given by

$$Y_{ni} = y_n + h \sum_{j=1}^{s} a_{ij} \dot{Y}_{nj}, \quad i = 1, \dots, s. \tag{5.6}$$

Here, y_n is an approximation to $y(t_n)$ with $t_n = t_{n-1} + h$ and $\mathcal{A} = [a_{ij}]$, $b = [b_i]$ are the coefficients of the Runge-Kutta method.

This approach is now extended to DAEs, see [13]. Consider the nonlinear DAE

$$F(y(t), \dot{y}(t)) = 0, \quad y(t_0) = \hat{y}_0, \tag{5.7}$$

where $F \in \mathcal{C}(\mathbb{R}^n \times \mathbb{R}^n, \mathbb{R}^n)$. An s-stage Runge-Kutta method applied to (5.7) is given by

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i \dot{Y}_{ni}, \quad y_0 = \hat{y}_0, \tag{5.8a}$$

where

$$F(Y_{ni}, \dot{Y}_{ni}) = 0 \tag{5.8b}$$

and the internal stages are given by

$$Y_{ni} = y_n + h \sum_{j=1}^{s} a_{ij} \dot{Y}_{nj}, \quad i = 1, \dots, s. \tag{5.8c}$$

For further description of the method we define $c_i := \sum_{j=1}^{s} a_{ij}$ for $i = 1, \dots, s$ and the conditions

$$\begin{aligned}
B(p):& \quad \sum_{i=1}^{s} b_i c_i^{k-1} = 1/k, \quad k = 1, \dots, p,\\
C(q):& \quad \sum_{j=1}^{s} a_{ij} c_j^{k-1} = c_i^k/k, \quad k = 1, \dots, q, \quad i = 1, \dots, s, \\
D(r):& \quad \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = b_j/k(1 - c_j^k), \quad k = 1, \dots, r, \quad j = 1, \dots, s.
\end{aligned} \tag{5.9}$$

Definition 5.1.1. *[14]*
A Runge-Kutta method (5.8) *has* (classical) order p *if for sufficiently smooth problems* (5.7) *it holds that*

$$\|y(t_0 + h) - y_1\| \le K h^p,$$

i.e., if the Taylor series for the exact solution $y(t_0 + h)$ and for the numerical solution y_1 coincide up to the term h^p.
If the second condition $C(q)$ in (5.9) *is fulfilled, then q is called the* stage order *of the Runge-Kutta method.*

5.2 Runge-Kutta Methods for DAEs of d-Index 1

We consider the following semi-explicit system of d-index 1

$$\begin{aligned} \dot{y}(t) &= f(y(t), z(t)), \\ 0 &= g(y(t), z(t)), \end{aligned} \tag{5.10}$$

with consistent initial values $y(t_0) = \hat{y}_0$, $z(t_0) = \hat{z}_0$ and properties as described in Section 2.2.1.
The direct application of the s-stage Runge-Kutta method (5.8) to the d-index 1 problem (5.10) yields

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{i=1}^{s} b_i \dot{Y}_{ni}, \quad y_0 = \hat{y}_0, \\ z_{n+1} &= z_n + h\sum_{i=1}^{s} b_i \dot{Z}_{ni}, \quad z_0 = \hat{z}_0, \end{aligned} \tag{5.11a}$$

where

$$\begin{aligned} \dot{Y}_{ni} &= f(Y_{ni}, Z_{ni}), \\ 0 &= g(Y_{ni}, Z_{ni}), \end{aligned} \tag{5.11b}$$

and the internal stages are given by

$$\begin{aligned} Y_{ni} &= y_n + h\sum_{j=1}^{s} a_{ij} \dot{Y}_{nj}, \\ Z_{ni} &= z_n + h\sum_{j=1}^{s} a_{ij} \dot{Z}_{nj}. \end{aligned} \tag{5.11c}$$

Assume that the coefficient matrix $\mathcal{A} = [a_{ij}]$ is nonsingular, then the *stability function* of the Runge-Kutta method is defined as

$$R(\infty) = 1 - b^T \mathcal{A}^{-1}[1, \dots, 1]^T. \tag{5.12}$$

We have the following convergence results for the d-index 1 case.

Theorem 5.2.1. *Suppose that $g_z(y, z)$ has a bounded inverse $\|(g_z(y, z))^{-1}\| \leq M$ in a neighborhood of the solution $[y(t), z(t)]$ of* (5.10) *and that the initial values $[\hat{y}_0, \hat{z}_0]$ are consistent. Consider the Runge-Kutta method* (5.11) *of classical order p, satisfying $C(q)$ in* (5.9) *(i.e., stage order q) with $p \geq q+1$ and having an invertible coefficient matrix $\mathcal{A} = [a_{ij}]$. The y-component of the numerical solution can be interpreted as the numerical result for an ODE. Therefore, we have $y_n - y(t_n) = O(h^p)$. For the z-component it holds that:*

1. *If $b_i = a_{si}$ for all i, then the global error for the z-component satisfies $z_n - z(t_n) = O(h^p)$ for $t_n = nh \le C$, with constant C.*
2. *If $-1 \le R(\infty) < 1$, then $z_n - z(t_n) = O(h^{q+1})$.*
3. *If $R(\infty) = +1$, then $z_n - z(t_n) = O(h^q)$.*
4. *If $|R(\infty)| > 1$, then the numerical solution diverges.*

Proof. See [13], Theorem 3.1. □

5.3 Runge-Kutta Methods for DAEs of d-Index 2

Now we consider the following semi-explicit system of d-index 2

$$\begin{aligned} \dot{y}(t) &= f(y(t), z(t)), \\ 0 &= g(y(t)), \end{aligned} \tag{5.13}$$

with consistent initial values $y(t_0) = \hat{y}_0$, $z(t_0) = \hat{z}_0$ and properties as described in Section 2.2.2.

The s-stage Runge-Kutta method (5.8) applied to the d-index 2 problem (5.13) reads

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{i=1}^{s} b_i \dot{Y}_{n_i}, \quad y_0 = \hat{y}_0, \\ z_{n+1} &= z_n + h\sum_{i=1}^{s} b_i \dot{Z}_{n_i}, \quad z_0 = \hat{z}_0, \end{aligned} \tag{5.14a}$$

where

$$\begin{aligned} \dot{Y}_{n_i} &= f(Y_{n_i}, Z_{n_i}), \\ 0 &= g(Y_{n_i}), \end{aligned} \tag{5.14b}$$

and the internal stages are given by

$$\begin{aligned} Y_{n_i} &= y_n + h\sum_{j=1}^{s} a_{ij} \dot{Y}_{n_j}, \\ Z_{n_i} &= z_n + h\sum_{j=1}^{s} a_{ij} \dot{Z}_{n_j}. \end{aligned} \tag{5.14c}$$

In order to give convergence results for the d-index 2 problem (5.13) we need to introduce the definition of the local error.
The difference between the numerical solution $[y_1, z_1]$ after one step with stepsize h and the exact solution at $t + h$ given by

$$\delta y_h(t) = y_1 - y(t+h), \quad \delta z_h(t) = z_1 - z(t+h), \tag{5.15}$$

is called the *local error*.

Lemma 5.3.1. *Suppose that the Runge-Kutta method* (5.14) *satisfies the conditions $B(p)$ and $C(q)$ (see* (5.9)*) with $p \geq q+1$ and $q \geq 1$. Then the local error is*

$$\begin{aligned} \delta y_h(t) &= O(h^{q+1}), \\ P(t)\delta y_h(t) &= O(h^{q+2}), \\ \delta z_h(t) &= O(h^q). \end{aligned}$$

Here, $P(t)$ is a projection given by

$$P(t) = I - Q(t), \quad Q(t) = (f_z(g_y f_z)^{-1} g_y)(y(t), z(t)). \tag{5.16}$$

Proof. See [13], Lemma 4.3. □

Now we can give the convergence results for the Runge-Kutta method (5.14) applied to the semi-explicit system of d-index 2.

Theorem 5.3.2. *Suppose that $g_y(y)f_z(y,z)$ has a bounded inverse $\|(g_y(y)f_z(y,z))^{-1}\| \leq M$ in a neighborhood of the solution $[y(t), z(t)]$ of* (5.13) *and that the initial values $[\hat{y}_0, \hat{z}_0]$ are consistent. Assume that the Runge-Kutta matrix $\mathcal{A} = [a_{ij}]$ is invertible, $|R(\infty)| < 1$ in* (5.12) *and the local error satisfies*

$$\delta y_h(t) = O(h^r), \quad P(t)\delta y_h(t) = O(h^{r+1})$$

with P(t) given in (5.16)*. Then the method* (5.14) *is convergent of order r, i.e.,*

$$y_n - y(t_n) = O(h^r) \quad \text{for} \quad t_n = nh \leq C,$$

with constant C.
If in addition $\delta y_h(t) = O(h^{r+1})$, then we have $g(y_n) = O(h^{r+1})$.
If the global error of the y-component is $O(h^r)$, $g(y_n) = O(h^{r+1})$ and the local error of the z-component is $O(h^r)$, then we have for the global error

$$z_n - z(t_n) = O(h^r) \quad \text{for} \quad t_n = nh \leq C. \tag{5.17}$$

Proof. See [13], Theorem 4.4. and Theorem 4.6. □

Remark 5.3.3. *Together with Lemma 5.3.1 the convergence order of the y-component is at least $r = q+1$.*

In the case $R(\infty) = \pm 1$ we have the following convergence result.

Theorem 5.3.4. *Suppose that $g_y(y)f_z(y,z)$ has a bounded inverse $\|(g_y(y)f_z(y,z))^{-1}\| \leq M$ in a neighborhood of the solution $[y(t), z(t)]$ of* (5.13) *and that the initial values $[\hat{y}_0, \hat{z}_0]$ are consistent. Assume that the Runge-Kutta matrix $\mathcal{A} = [a_{ij}]$ is invertible, and let conditions $B(p)$ and $C(q)$ in* (5.9) *be satisfied.*

1. *If $R(\infty) = +1$, $q \geq 2$ and $p \geq q$, then $y_n - y(t_n) = O(h^q)$ and $z_n - z(t_n) = O(h^{q-2})$.*

2. *If $R(\infty) = -1$, $q \geq 1$ and $p \geq q+1$, then $y_n - y(t_n) = O(h^{q+1})$ and $z_n - z(t_n) = O(h^{q-1})$.*

Proof. See [13], Theorem 4.9. □

5.4 Runge-Kutta Methods for DAEs of d-Index 3

Finally, we consider the following semi-explicit system of d-index 3

$$\begin{aligned}\dot{y}(t) &= f(y(t), z(t)),\\ \dot{z}(t) &= k(y(t), z(t), u(t)),\\ 0 &= g(y(t)),\end{aligned} \tag{5.18}$$

with consistent initial values $y(t_0) = \hat{y}_0$, $z(t_0) = \hat{z}_0$, $u(t_0) = \hat{u}_0$ and properties as described in Section 2.2.3.
The Runge-Kutta method (5.8) applied to the d-index 3 problem (5.18) reads

$$\begin{aligned}y_{n+1} &= y_n + h\sum_{i=1}^{s} b_i \dot{Y}_{n_i}, \quad y_0 = \hat{y}_0,\\ z_{n+1} &= z_n + h\sum_{i=1}^{s} b_i \dot{Z}_{n_i}, \quad z_0 = \hat{z}_0,\\ u_{n+1} &= u_n + h\sum_{i=1}^{s} b_i \dot{U}_{n_i}, \quad u_0 = \hat{u}_0,\end{aligned} \tag{5.19a}$$

where

$$\begin{aligned}\dot{Y}_{n_i} &= f(Y_{n_i}, Z_{n_i}),\\ \dot{Z}_{n_i} &= k(Y_{n_i}, Z_{n_i}, U_{n_i}),\\ 0 &= g(Y_{n_i}),\end{aligned} \tag{5.19b}$$

and the internal stages are given by

$$\begin{aligned}Y_{n_i} &= y_n + h\sum_{j=1}^{s} a_{ij} \dot{Y}_{n_j},\\ Z_{n_i} &= z_n + h\sum_{j=1}^{s} a_{ij} \dot{Z}_{n_j},\\ U_{n_i} &= u_n + h\sum_{j=1}^{s} a_{ij} \dot{U}_{n_j}.\end{aligned} \tag{5.19c}$$

We have the following convergence results for the d-index 3 case.

Theorem 5.4.1. *Suppose that $g_y(y)f_z(y,z)k_u(y,z,u)$ has a bounded inverse $\|(g_y(y)f_z(y,z)k_u(y,z,u))^{-1}\| \le M$ in a neighborhood of the solution $[y(t), z(t), u(t)]$ of (5.18) and that the initial values $[\hat{y}_0, \hat{z}_0, \hat{u}_0]$ are consistent. Assume that the Runge-Kutta matrix $\mathcal{A} = [a_{ij}]$ is invertible, $|R(\infty)| < 1$ (see (5.12)), and that the conditions $B(p)$, $C(q)$ in (5.9) hold with $p \ge q+1$ and $q \ge 2$. Then we have the following estimates for the global error:*

$$\begin{aligned}y_n - y(t_n) &= O(h^q),\\ z_n - z(t_n) &= O(h^q),\\ u_n - u(t_n) &= O(h^{q-1})\end{aligned} \tag{5.20}$$

for $t_n = nh \le C$, with constant C.

Proof. See [13], Theorem 6.4. $\square$

Theorem 5.4.2. *In addition to the assumptions of Theorem 5.4.1 suppose that* $q \geq 3$ *and the simplifying assumption D(1) in* (5.9). *Then we have for* $nh \leq C$

$$y_n - y(t_n) = O(h^{q+1}). \tag{5.21}$$

Proof. See [13], Theorem 6.7. $\square$

Theorem 5.4.3. *In addition to the assumptions of Theorem 5.4.1 suppose that* $q \geq 4$, $p \geq q+2$ *and the simplifying assumption D(2) in* (5.9) *and* $\sum_{i,j=1}^{s} b_i w_{ij} c_j^{q+1} = 1$. *Then we have for* $nh \leq C$

$$y_n - y(t_n) = O(h^{q+2}). \tag{5.22}$$

Proof. See [13], Theorem 6.8. $\square$

Chapter 6

Runge-Kutta Methods for Second Order DAEs

In this chapter we develop a Runge-Kutta method for the numerical solution of second order differential-algebraic systems. Runge-Kutta methods can be derived by collocation as described in [15, 33]. We follow this approach to derive Runge-Kutta methods for second order equations and apply these methods to the equations of motion of multibody systems.

6.1 Collocation

In this section we will first explain the concept of collocation for the first order initial value problem in ordinary differential equations. In the second part we extend this idea to second order initial value problems for ODEs.

6.1.1 Collocation for First Order Systems

Runge-Kutta methods for first order initial value problems can be seen as collocation methods. The concept of Runge-Kutta methods as collocation methods is introduced in [15, 33].
Consider the initial value problem

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = y_0 \text{ in } [t_0, t_0 + h]. \tag{6.1}$$

A polynomial $w_s(t)$ of degree s is a collocation polynomial for the initial value problem (6.1)if it satisfies the conditions

$$\begin{aligned} w_s(t_0) &= y_0, \\ \dot{w}_s(t_0 + c_i h) &= f(t_0 + c_i h, w_s(t_0 + c_i h)), \quad i = 1, \ldots, s. \end{aligned} \tag{6.2}$$

We can write $\dot{w}_s(t_0 + xh)$ by means of the Lagrange interpolation formula as

$$\dot{w}_s(t_0 + xh) = \sum_{j=1}^{s} \dot{w}_s(t_0 + c_j h) L_j(x), \quad x \in [0, 1], \tag{6.3}$$

with the Lagrange polynomials

$$L_j(x) = \prod_{\substack{l=1 \\ l\neq j}}^{s} \frac{x - c_l}{c_j - c_l}, \quad j = 1, \ldots, s. \tag{6.4}$$

By integration it follows that

$$w_s(t_0 + c_i h) = w_s(t_0) + h \int_0^{c_i} \sum_{j=1}^{s} \dot{w}_s(t_0 + c_j h) L_j(x) dx \tag{6.5}$$

and

$$w_s(t_0 + h) = w_s(t_0) + h \int_0^{1} \sum_{j=1}^{s} \dot{w}_s(t_0 + c_j h) L_j(x) dx. \tag{6.6}$$

With

$$a_{ij} = \int_0^{c_i} L_j(x) dx, \quad b_j = \int_0^{1} L_j(x) dx, \quad i, j = 1, \ldots, s, \tag{6.7}$$

we get for $i = 1, \ldots, s$

$$\begin{aligned} w_s(t_0 + c_i h) &= w_s(t_0) + h \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, w_s(t_0 + c_j h)), \\ w_s(t_0 + h) &= w_s(t_0) + h \sum_{i=1}^{s} b_i f(t_0 + c_i h, w_s(t_0 + c_i h)). \end{aligned} \tag{6.8}$$

Using the formulas (6.8) for each integration step we get the s-stage implicit Runge-Kutta method as

$$\begin{aligned} y_0 &= w_s(t_0), \\ y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i f(t_n + c_i h, Y_{ni}), \\ Y_{ni} &= y_n + h \sum_{j=1}^{s} a_{ij} f(t_n + c_j h, Y_{nj}), \quad i = 1, \ldots, s, \end{aligned} \tag{6.9}$$

where $y_{n+1} = w_s(t_n + h)$ and $Y_{ni} = w_s(t_n + c_i h)$.

6.1.2 Collocation for Second Order Systems

The ideas described in Section 6.1.1 can be extended for the numerical solution of second order systems. We consider the second order initial value problem

$$\begin{aligned} \ddot{y}(t) &= f(t, y(t), \dot{y}(t)), \\ y(t_0) &= y_0, \\ \dot{y}(t_0) &= y_1 \end{aligned} \tag{6.10}$$

in $[t_0, t_0+h]$. Now, we have the two collocation polynomials $w_s(t)$ and $\pi_s(t)$ with

$$\begin{aligned} w_s(t_0) &= y_0, \\ \dot{w}_s(t_0+c_ih) &= \pi_s(t_0+c_ih), \end{aligned} \tag{6.11}$$

$$\begin{aligned} \pi_s(t_0) &= y_1, \\ \dot{\pi}_s(t_0+c_ih) &= f(t_0+c_ih, w_s(t_0+c_ih), \pi_s(t_0+c_ih)). \end{aligned} \tag{6.12}$$

Again we can write $\dot{\pi}_s(t_0+xh)$ and $\dot{w}_s(t_0+xh)$ by means of the Lagrange interpolation formula as

$$\begin{aligned} \dot{\pi}_s(t_0+xh) &= \sum_{j=1}^{s} \dot{\pi}_s(t_0+c_jh)L_j(x), \\ \dot{w}_s(t_0+xh) &= \sum_{j=1}^{s} \dot{w}_s(t_0+c_jh)L_j(x), \quad x \in [0,1]. \end{aligned} \tag{6.13}$$

By integration it follows that

$$\begin{aligned} \pi_s(t_0+c_ih) &= \pi_s(t_0) + h\int_0^{c_i} \sum_{j=1}^{s} \dot{\pi}_s(t_0+c_jh)L_j(x)dx \\ &= \pi_s(t_0) + h\sum_{j=1}^{s} a_{ij} f(t_0+c_jh, w_s(t_0+c_jh), \pi_s(t_0+c_jh)) \end{aligned} \tag{6.14a}$$

and

$$\pi_s(t_0+h) = \pi_s(t_0) + h\sum_{i=1}^{s} b_i f(t_0+c_ih, w_s(t_0+c_ih), \pi_s(t_0+c_ih)). \tag{6.14b}$$

For the second collocation polynomial $w_s(t)$ we get by integration and using the equations (6.11) and (6.14a) the formulations

$$\begin{aligned} w_s(t_0+c_ih) &= w_s(t_0) + h\int_0^{c_i} \sum_{j=1}^{s} \dot{w}_s(t_0+c_jh)L_j(x)dx \\ &= w_s(t_0) + h\sum_{j=1}^{s} a_{ij}\dot{w}_s(t_0+c_jh) \\ &= w_s(t_0) + h\sum_{j=1}^{s} a_{ij}\left[\pi_s(t_0) + h\sum_{l=1}^{s} a_{jl} f(t_0+c_lh, w_s(t_0+c_lh), \pi_s(t_0+c_lh))\right] \\ &= w_s(t_0) + h\sum_{j=1}^{s} a_{ij}\pi_s(t_0) + h^2\sum_{j,l=1}^{s} a_{ij}a_{jl} f(t_0+c_lh, w_s(t_0+c_lh), \pi_s(t_0+c_lh)) \\ &= w_s(t_0) + hc_i\pi_s(t_0) + h^2\sum_{l=1}^{s} \tilde{a}_{il} f(t_0+c_lh, w_s(t_0+c_lh), \pi_s(t_0+c_lh)), \end{aligned} \tag{6.14c}$$

and

$$\begin{aligned}
w_s(t_0+h) &= w_s(t_0) + h\sum_{i=1}^{s} b_i \dot{w}_s(t_0+c_ih) \\
&= w_s(t_0) + h\sum_{i=1}^{s} b_i \pi_s(t_0) + h^2 \sum_{i,j=1}^{s} b_i a_{ij} f(t_0+c_jh, w_s(t_0+c_jh), \pi_s(t_0+c_jh)) \\
&= w_s(t_0) + h\pi_s(t_0) + h^2 \sum_{j=1}^{s} \tilde{b}_j f(t_0+c_jh, w_s(t_0+c_jh), \pi_s(t_0+c_jh)),
\end{aligned} \tag{6.14d}$$

where we have used that $\sum_{i=1}^{s} b_i = 1$ and $\sum_{j=1}^{s} a_{ij} = c_i$ and define

$$\tilde{a}_{il} = \sum_{j=1}^{s} a_{ij}a_{jl} \text{ and } \tilde{b}_j = \sum_{i=1}^{s} b_i a_{ij}, \quad i,j,l = 1,\dots,s. \tag{6.14e}$$

Using the formulas (6.14) we get the s-stage implicit Runge-Kutta method for the second order system (6.10) as

$$\begin{aligned}
y_{n+1} &= y_n + h\dot{y}_n + h^2\sum_{i=1}^{s} \tilde{b}_i f(t_n+c_ih, Y_{ni}, \dot{Y}_{ni}), \\
\dot{y}_{n+1} &= \dot{y}_n + h\sum_{i=1}^{s} b_i f(t_n+c_ih, Y_{ni}, \dot{Y}_{ni}), \\
Y_{ni} &= y_n + hc_i\dot{y}_n + h^2\sum_{j=1}^{s} \tilde{a}_{ij} f(t_n+c_jh, Y_{nj}, \dot{Y}_{nj}), \\
\dot{Y}_{ni} &= \dot{y}_n + h\sum_{j=1}^{s} a_{ij} f(t_n+c_jh, Y_{nj}, \dot{Y}_{nj}),
\end{aligned} \tag{6.15}$$

where $y_{n+1} = w_s(t_n+h)$, $\dot{y}_{n+1} = \pi_s(t_n+h)$, $Y_{ni} = w_s(t_n+c_ih)$, $\dot{Y}_{ni} = \pi_s(t_n+c_ih)$ and $i = 1,\dots,s$.

In the following we give the coefficients for two Runge-Kutta methods of the form (6.15). We choose known Runge-Kutta collocation methods for the approximation of $\dot{y}_{n+1}$ with the coefficients a_{ij}, b_i and c_i and determine $\tilde{a}_{ij}$ and $\tilde{b}_i$ using the formulas (6.14e). Gauss methods and Radau-IIA methods are collocation methods, see [15, 33]. Thus, using the notation

$$\begin{array}{c|cc|cc}
c_1 & a_{11} & a_{12} & \tilde{a}_{11} & \tilde{a}_{12} \\
c_2 & a_{21} & a_{22} & \tilde{a}_{21} & \tilde{a}_{22} \\
\hline
 & b_1 & b_2 & \tilde{b}_1 & \tilde{b}_2
\end{array}$$

we can give the coefficients of two 2-stage Runge-Kutta methods. The coefficients are given in Table 6.1.

Theorem 6.1.1. *The Runge-Kutta method* (6.15) *with* $s = 2$ *and coefficients given in Table 6.1 is convergent of order 4 for sufficiently smooth problems* (6.10), *i.e.,*

$$\|y(t_0+h) - y_1\| = O(h^4).$$

	$\frac{3-\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{3-2\sqrt{3}}{12}$	$\frac{1}{24}$	$\frac{3-2\sqrt{3}}{24}$
Gauss	$\frac{3+\sqrt{3}}{6}$	$\frac{3+2\sqrt{3}}{12}$	$\frac{1}{4}$	$\frac{3+2\sqrt{3}}{24}$	$\frac{1}{24}$
		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3+\sqrt{3}}{12}$	$\frac{3-\sqrt{3}}{12}$
	$\frac{1}{3}$	$\frac{5}{12}$	$\frac{-1}{12}$	$\frac{1}{9}$	$-\frac{1}{18}$
Radau-IIA	1	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0
		$\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0

Table 6.1: Runge-Kutta coefficients

Proof. We examine the Taylor expansion of the exact solution of (6.10) and of the solution obtained by the Runge-Kutta method (6.15) after one step.

$$\begin{aligned} y(t+h) &= y(t) + h\dot{y}(t) + \frac{h^2}{2}\ddot{y}(t) + \frac{h^3}{6}y^{(3)}(t) + O(h^4) \\ &= y(t) + h\dot{y}(t) + \frac{h^2}{2}f(t,y,\dot{y}) + \frac{h^3}{6}\left[f_t(t,y,\dot{y}) + f_y(t,y,\dot{y})\dot{y}\right. \\ &\quad \left. + f_{\dot{y}}(t,y,\dot{y})f(t,y,\dot{y})\right] + O(h^4). \end{aligned} \tag{6.16}$$

$$\begin{aligned} y_1 &= y(t) + h\dot{y}(t) + h^2[\tilde{b}_1 f(t+c_1h, Y_1(t+h), \dot{Y}_1(t+h)) \\ &\qquad + \tilde{b}_2 f(t+c_2h, Y_2(t+h), \dot{Y}_2(t+h))], \\ &= y(t) + h\dot{y}(t) + h^2\tilde{b}_1[f + ha_{11}f_{\dot{y}}f + ha_{12}f_{\dot{y}}f + hc_1f_y\dot{y} + hc_1f_t] \\ &\quad + h^2\tilde{b}_2[f + ha_{21}f_{\dot{y}}f + ha_{22}f_{\dot{y}}f + hc_2f_y\dot{y} + hc_2f_t] + O(h^4). \end{aligned} \tag{6.17}$$

Then it follows that

$$\begin{aligned} y(t+h) - y_1 &= h^2\left[\frac{1}{2}f - \tilde{b}_1f - \tilde{b}_2f\right] + h^3\left[\frac{1}{6}f_t + \frac{1}{6}f_y\dot{y} + \frac{1}{6}f_{\dot{y}}f\right. \\ &\quad - \tilde{b}_1a_{11}f_{\dot{y}}f - \tilde{b}_1a_{12}f_{\dot{y}}f - \tilde{b}_1c_1f_y\dot{y} - \tilde{b}_1c_1f_t - \tilde{b}_2a_{21}f_{\dot{y}}f \\ &\quad \left. - \tilde{b}_2a_{22}f_{\dot{y}}f - \tilde{b}_2c_2f_y\dot{y} - \tilde{b}_2c_2f_t\right] + O(h^4). \end{aligned} \tag{6.18}$$

To obtain order 4 we get the three conditions

$$\begin{aligned} \tilde{b}_1 + \tilde{b}_2 &= \frac{1}{2}, \\ \tilde{b}_1c_1 + \tilde{b}_2c_2 &= \frac{1}{6}, \\ \tilde{b}_1(a_{11}+a_{12}) + \tilde{b}_2(a_{21}+a_{22}) &= \frac{1}{6}. \end{aligned} \tag{6.19}$$

These conditions are satisfied by the coefficients given in Table 6.1. □

Remark 6.1.2. *The s-stage Gauss method has order $p = 2s$ and the s-stage Radau-IIA method has order $p = 2s - 1$, see [33]. Thus, for $s = 2$ the approximation $\dot{y}_n$ to $\dot{y}(t_n)$ in* (6.15) *is of order 4 for the Gauss method and of order 3 for the Radau-IIA method.*

6.2 Application to Second Order DAEs

In this section we describe the application of method (6.15) to second order DAEs of the form

$$\begin{aligned} \ddot{y}(t) &= f(t, y(t), \dot{y}(t), \lambda(t)), \\ 0 &= g(t, y(t)). \end{aligned} \tag{6.20}$$

The Runge-Kutta method (6.15) applied to the second order DAE (6.20) has the form

$$\begin{aligned} y_{n+1} &= y_n + h\dot{y}_n + h^2 \sum_{i=1}^{s} \tilde{b}_i \ddot{Y}_{ni}, \\ \dot{y}_{n+1} &= \dot{y}_n + h \sum_{i=1}^{s} b_i \ddot{Y}_{ni}, \\ \lambda_{n+1} &= \lambda_n + h\dot{\lambda}_n + h^2 \sum_{i=1}^{s} \tilde{b}_i \ddot{\Lambda}_{ni}, \\ \dot{\lambda}_{n+1} &= \dot{\lambda}_n + h \sum_{i=1}^{s} b_i \ddot{\Lambda}_{ni}, \end{aligned} \tag{6.21}$$

where

$$\begin{aligned} \ddot{Y}_{ni} &= f(t_n + c_i h, Y_{ni}, \dot{Y}_{ni}, \Lambda_{ni}), \\ 0 &= g(t_n + c_i h, Y_{ni}) \end{aligned} \tag{6.22}$$

and the internal stages are given by

$$\begin{aligned} Y_{ni} &= y_n + hc_i\dot{y}_n + h^2 \sum_{j=1}^{s} \tilde{a}_{ij} \ddot{Y}_{nj}, \\ \dot{Y}_{ni} &= \dot{y}_n + h \sum_{j=1}^{s} a_{ij} \ddot{Y}_{nj}, \\ \Lambda_{ni} &= \lambda_n + hc_i\dot{\lambda}_n + h^2 \sum_{j=1}^{s} \tilde{a}_{ij} \ddot{\Lambda}_{nj} \end{aligned} \tag{6.23}$$

for $i = 1, \ldots, s$. In the following we will restrict ourselves to Runge-Kutta methods with 2 stages, i.e., $s = 2$ and coefficients given in Table 6.1.

6.3 Application to the Equations of Motion

Now we consider again the equations of motion of multibody systems and use the 2-stage Runge-Kutta method (6.15) in the same way as in Section 6.2 for the numerical solution of the equations of motion. For that we consider the simplified equations of motion

$$\begin{aligned} M\ddot{q} &= f_a(q,\dot{q},\lambda,t) - G^T(q,t)\lambda, \\ 0 &= g(q,t). \end{aligned} \tag{6.24}$$

The Runge-Kutta method (6.15) applied to (6.24) is given by

$$\begin{aligned} q_{n+1} &= q_n + h\dot{q}_n + h^2\sum_{i=1}^{2}\tilde{b}_i\ddot{Q}_{ni}, \\ \dot{q}_{n+1} &= \dot{q}_n + h\sum_{i=1}^{2} b_i\ddot{Q}_{ni}, \\ \lambda_{n+1} &= \lambda_n + h\dot{\lambda}_n + h^2\sum_{i=1}^{2}\tilde{b}_i\ddot{\Lambda}_{ni}, \\ \dot{\lambda}_{n+1} &= \dot{\lambda}_n + h\sum_{i=1}^{2} b_i\ddot{\Lambda}_{ni}, \end{aligned} \tag{6.25a}$$

where for $i = 1, 2$

$$\begin{aligned} M\ddot{Q}_{ni} &= f_a(Q_{ni},\dot{Q}_{ni},\Lambda_{ni},t_n + c_i h) - G^T(Q_{ni},t_n + c_i h)\Lambda_{ni}, \\ 0 &= g(t_n + c_i h, Q_{ni}), \end{aligned} \tag{6.25b}$$

and the internal stages are given by

$$\begin{aligned} Q_{ni} &= q_n + hc_i\dot{q}_n + h^2\sum_{j=1}^{2}\tilde{a}_{ij}\ddot{Q}_{nj}, \\ \dot{Q}_{ni} &= \dot{q}_n + h\sum_{j=1}^{2} a_{ij}\ddot{Q}_{nj}, \\ \Lambda_{ni} &= \lambda_n + hc_i\dot{\lambda}_n + h^2\sum_{j=1}^{2}\tilde{a}_{ij}\ddot{\Lambda}_{nj}. \end{aligned} \tag{6.25c}$$

The equations (6.25b) have to be solved in each integration step to obtain $\ddot{Q}_{ni}$ and $\ddot{\Lambda}_{ni}$ for $i = 1, 2$, which can then be inserted into (6.25a) for the determination of the new values q_{n+1} and λ_{n+1}. The nonlinear system (6.25b) can be formulated with $X_n = [\ddot{Q}_{n1}, \ddot{Q}_{n2}, \ddot{\Lambda}_{n1}, \ddot{\Lambda}_{n2}]^T$ as

$$\begin{aligned} &F(X_n) = \\ &\begin{bmatrix} M\ddot{Q}_{n1} - f_a(Q_{n1},\dot{Q}_{n1},\Lambda_{n1},t_n + c_1 h) + G^T(Q_{n1},t_n + c_1 h)\Lambda_{n1} \\ M\ddot{Q}_{n2} - f_a(Q_{n2},\dot{Q}_{n2},\Lambda_{n2},t_n + c_2 h) + G^T(Q_{n2},t_n + c_2 h)\Lambda_{n2} \\ g(t_n + c_1 h, Q_{n1}) \\ g(t_n + c_2 h, Q_{n2}) \end{bmatrix} = 0. \end{aligned} \tag{6.26}$$

System (6.26) is solved by means of a Newton method as described in Chapter 4. For the Newton method we need the Jacobian of F, which is given by

$$DF(X_n) = \begin{bmatrix} M - T_{11} & -T_{12} & -T_{13} & -T_{14} \\ -T_{21} & M - T_{22} & -T_{23} & -T_{24} \\ g^{(1)}_{\ddot{Q}_{n1}} & g^{(1)}_{\ddot{Q}_{n2}} & 0 & 0 \\ g^{(2)}_{\ddot{Q}_{n1}} & g^{(2)}_{\ddot{Q}_{n2}} & 0 & 0 \end{bmatrix}, \qquad (6.27)$$

where for $i = 1, 2$

$$\begin{aligned} T_{i1} &= f^{(i)}_{a,\ddot{Q}_{n1}} + G^{(i)T}_{\ddot{Q}_{n1}} \Lambda_{ni}, \\ T_{i2} &= f^{(i)}_{a,\ddot{Q}_{n2}} + G^{(i)T}_{\ddot{Q}_{n2}} \Lambda_{ni}, \\ T_{i3} &= f^{(i)}_{a,\ddot{\Lambda}_{n1}} + G^{(i)T} h^2 \tilde{a}_{i1}, \\ T_{i4} &= f^{(i)}_{a,\ddot{\Lambda}_{n2}} + G^{(i)T} h^2 \tilde{a}_{i2}. \end{aligned}$$

Here, $f_a^{(i)} := f_a(t_n + c_i h, Q_{ni}, \dot{Q}_{ni}, \Lambda_{ni})$ and $G^{(i)T} := G^T(Q_{ni}, t_n + c_i h)$.

For the error estimation we use the same extrapolation strategies as used for the modified BDF methods and is described in Section 4.3. Also the stepsize selection is similar to the one used in the BDF methods only with some small modifications as we do not consider Runge-Kutta methods of different order and therefore need no order selection here.

In Chapter 8 we give some numerical results for the application of the Runge-Kutta method (6.25) to the equations of motion of multibody systems.

Other frequently used methods for the numerical solution of higher index DAEs are general linear methods. In a similar way as for the Runge-Kutta methods the concepts of general linear methods can be extended for the numerical solution of second order differential-algebraic systems. As we have not further examine this approach yet, we will given the convergence results for general linear methods applied to first order higher index DAEs and a formulation for general linear methods applied to second order systems in the appendix. Further investigations and the implementation of general linear methods for second order DAEs remain for future work.

Chapter 7

Index Reduction Techniques

In this chapter we describe a number of techniques to reduce the index of a DAE and in particular of the equations of motion of multibody systems, as a high index leads to difficulties in the numerical solution. The DAE is transformed to an analytically equivalent system before applying the numerical method. In the first part of this chapter we describe the index reduction techniques for the first order form of the equations of motion as treated in [11, 22, 32] and in the second part we extend this techniques to the second order form.

7.1 Index Reduction for First Order Systems

The basic idea for the index reduction of a differential-algebraic system is the following. Consider the nonlinear initial value problem

$$F(t, y(t), \dot{y}(t)) = 0, \quad y(t_0) = y_0. \tag{7.1}$$

The analytical transformation for the index reduction is based on the fact that each solution $y(t)$ of (7.1) satisfies also the derivatives

$$\frac{d^r}{dt^r} F(t, y(t), \dot{y}(t)) = 0, \quad (r > 0) \tag{7.2}$$

for sufficiently smooth functions F.
If the DAE (7.1) contains algebraic constraints, than each solution $y(t)$ of (7.1) satisfies the hidden constraints arising in (7.2). If we replace the constraints in (7.1) by their time derivatives, i.e., among others by hidden constraints, then the analytical solution of the problem stays the same, but the newly created analytically equivalent system has in general a lower index than the original system.

We now consider the first order form of the equations of motion as derived in Chapter 1. Introducing new velocity variables v leads to the corresponding first order system

$$
\begin{aligned}
\dot{q} &= v, && (7.3a)\\
M(q)\dot{v} &= f(q, v, \lambda, t) - G^T(q,t)\lambda, && (7.3b)\\
0 &= g(q,t). && (7.3c)
\end{aligned}
$$

The equations (7.3) form a nonlinear system of differential-algebraic equations with d-index 3 if they satisfy the condition that

$$
\begin{gathered}
G(q,t)M^{-1}(q)G^T(q,t) \in \mathbb{R}^{n_\lambda, n_\lambda}\\
\text{is nonsingular for all } q,\ t \text{ which satisfies (7.3c)}
\end{gathered}
\tag{7.4}
$$

Here, we only consider multibody systems that satisfy (7.4) and therefore have d-index 3.

The higher index DAE (7.3) contains hidden constraints, i.e., algebraic constraints, which do not appear explicitly in the original form of the system. The original constraints on position level are given by $g(q,t) = 0$. If we differentiate the constraints once with respect to t, we get the hidden constraints on velocity level

$$
\begin{aligned}
0 &= \frac{d}{dt} g(q,t)\\
&= g_q(q,t)\dot{q} + g_t(q,t)\\
&= G(q,t)\dot{q} + g_t(q,t)\\
&= G(q,t)v + g_t(q,t) =: g^I(q,v,t),
\end{aligned}
\tag{7.5}
$$

and a second differentiation yields the hidden constraints on acceleration level

$$
\begin{aligned}
0 &= \frac{d^2}{dt^2} g(q,t)\\
&= \frac{d}{dt}(G(q,t)v + g_t(q,t))\\
&= \frac{d}{dt}(G(q,t))v + G(q,t)\dot{v} + g_{tt}(q,t) + g_{tq}(q,t)\dot{q}\\
&= 2\frac{d}{dt}G(q,t)v + G(q,t)M^{-1}(q)[f(q,v,\lambda,t) - G^T(q,t)\lambda]\\
&\quad + g_{tt}(q,t)\\
&=: g^{II}(q,v,\lambda,t).
\end{aligned}
\tag{7.6}
$$

A classical method to reduce the d-index of system (7.3) is to replace the algebraic constraints by the hidden constraints. If we replace the constraints (7.3c) by the constraints on velocity level (7.5), then we get the system

$$
\begin{aligned}
\dot{q} &= v,\\
M(q)\dot{v} &= f(q,v,\lambda,t) - G^T(q,t)\lambda,\\
0 &= g^I(q,v,t).
\end{aligned}
\tag{7.7}
$$

This system has d-index 2 and for consistent initial values with $g(q_0, t_0) = 0$ the solution of the initial value problem does not change, see [32]. This formulation is also called the *Index-2-formulation* of the equations of motion.

If we replace the constraints (7.3c) by the constraints on acceleration level (7.6), then we get the following system

$$\begin{aligned}\dot{q} &= v,\\ M(q)\dot{v} &= f(q,v,\lambda,t) - G^T(q,t)\lambda,\\ 0 &= g^{II}(q,v,\lambda,t).\end{aligned} \tag{7.8}$$

This system has d-index 1 and is called the *Index-1-formulation* of the equations of motion, see [32].
The numerical solution of (7.7) satisfies the hidden constraints on velocity level, but in general it does not satisfy the holonomic constraints $g(q,t) = 0$. On the other hand the numerical solution of (7.8) satisfies the hidden constraints on acceleration level, but not the hidden constraints on velocity level or the original constraints on position level. In both cases the numerical solution deviates from the position manifold $\{\eta : g(\eta) = 0\}$ with increasing time (Drift-off effect).

Adding both, the hidden constraints on velocity level (7.5) and the hidden constraints on acceleration level (7.6) to the original equations of motion (7.3) gives the *reduced derivative array* of the equations of motion as

$$\begin{aligned}\dot{q} &= v,\\ M(q)\dot{v} &= f(q,v,\lambda,t) - G^T(q,t)\lambda,\\ 0 &= g(q,t),\\ 0 &= g^I(q,v,t),\\ 0 &= g^{II}(q,v,\lambda,t).\end{aligned} \tag{7.9}$$

7.1.1 Gear-Gupta-Leimkuhler Formulation

If we want to prevent that the solution deviates from the position manifold $\{\eta : g(\eta) = 0\}$ we can consider the original constraints (7.3c) and in addition the hidden constraints on velocity level (7.5) directly in the formulation of the equations of motion. This approach was first given in [11]. We introduce new Lagrange multipliers $\eta \in \mathbb{R}^{n_\lambda}$ to insure that the constraints on velocity level are satisfied. This leads to the following formulation

$$\begin{aligned}\dot{q} &= v - G^T(q,t)\eta,\\ M(q)\dot{v} &= f(q,v,\lambda,t) - G^T(q,t)\lambda,\\ 0 &= g(q,t),\\ 0 &= g^I(q,v,t).\end{aligned} \tag{7.10}$$

This formulation is called the *Gear-Gupta-Leimkuhler formulation (GGL-formulation).*
By introducing the new variables η the dimension of the system is increased by n_λ but the d-index is reduced from 3 to 2. To see that the system has d-index 2 we multiply the second equation in (7.10) from the left by $M^{-1}(q)$. We get a semi-explicit system of d-index 2 as described in Section 2.2.2 with

$y = [q^T, v^T]^T$, $z = [\eta^T, \lambda^T]^T$, where the index 2 condition "$[g_y f_z](t, y, z)$ is regular" is fulfilled. The disadvantage of the GGL-formulation is that we have still a higher index and hidden constraints on acceleration level.

7.1.2 Index Reduction by Minimal Extension

An index reduction technique for DAEs with extra structural information is the index reduction by minimal extension described in [22]. In this case one uses the structure of the system to derive a minimally extended strangeness-free system using the reduced derivative array (7.9). Here, we only consider the index reduction by minimal extension for the equations of motion of multibody systems. For a detailed description and for general systems see [22].

Consider the reduced derivative array (7.9). To obtain the minimally extended strangeness-free system we determine a permutation matrix Π, such that

$$G(q,t)\Pi = [G_1 \quad G_2], \tag{7.11}$$

where G_2 is square and nonsingular. Next we partition

$$\Pi^T q = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}, \quad \Pi^T v = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \tag{7.12}$$

If we now replace every $\dot{p}_2$ by the new variable w_1 and every $\dot{z}_2$ by the new variable w_2, we get the extended system

$$\begin{aligned} \dot{p}_1 &= z_1, \\ w_1 &= z_2, \\ M\Pi \begin{bmatrix} \dot{z}_1 \\ w_2 \end{bmatrix} &= f(p, z, \lambda, t) - G^T(p,t)\lambda, \\ 0 &= g(p,t), \\ 0 &= g^I(p,z,t), \\ 0 &= \tilde{g}^{II}(p,z,\lambda,t), \end{aligned} \tag{7.13}$$

with $p = \Pi \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$, $z = \Pi \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ and

$$\tilde{g}^{II}(p,z,\lambda,t) := 2\frac{d}{dt}(G(p,t))z + G(p,t)M^{-1}\left[f(p,z,\lambda,t) - G^T(p,t)\lambda\right] + g_{tt}(p,t). \tag{7.14}$$

The system (7.13) is strangeness-free, corresponding to a d-index 1, see [22].

7.2 Index Reduction for Second Order Systems

The same techniques as in Section 7.1 can be applied for the index reduction of second order differential-algebraic systems. In this section we give the analogous formulations for the second order equations of motion.

To do this, we consider again the equations of motion of constrained multibody systems in the second order form with d-index 3 as derived in Chapter 1

$$\begin{aligned} M(q)\ddot{q}(t) &= f(q,\dot{q},\lambda,t) - G^T(q,t)\lambda, \\ 0 &= g(q,t). \end{aligned} \tag{7.15}$$

The Index-1-formulation and the Index-2-formulation can be obtained in a similar way as for the first order systems by replacing the original constraints by the constraints on velocity level or by the constraints on acceleration level. We get

$$\begin{aligned} M(q)\ddot{q} &= f(q,\dot{q},\lambda,t) - G^T(q,t)\lambda, \\ 0 &= g^I(q,\dot{q},t), \end{aligned} \tag{7.16}$$

for the Index-2-formulation and

$$\begin{aligned} M(q)\ddot{q} &= f(q,\dot{q},\lambda,t) - G^T(q,t)\lambda, \\ 0 &= g^{II}(q,\dot{q},\lambda,t), \end{aligned} \tag{7.17}$$

for the Index-1-formulation.
To see that system (7.16) has d-index 2 we use Definition 3.1.1. The corresponding first order system is equivalent to (7.7). Thus, according to Definition 3.1.1 system (7.16) has d-index 2. The same holds for the Index-1-formulation (7.17).
The reduced derivative array for the second order system (7.15) is given by

$$\begin{aligned} M(q)\ddot{q}(t) &= f(q,\dot{q},\lambda,t) - G^T(q,t)\lambda, \\ 0 &= g(q,t), \\ 0 &= g^I(q,\dot{q},t), \\ 0 &= g^{II}(q,\dot{q},\lambda,t). \end{aligned} \tag{7.18}$$

7.2.1 Gear-Gupta-Leimkuhler Formulation

Following the ideas in Section 7.1.1 and if we want to guarantee that the solution is maintained then the GGL-formulation for the second order system (7.15) is given by

$$\begin{aligned} M(q)\ddot{q} &= f(q,\dot{q}+G^T(q,t)\eta,\lambda,t) - G^T(q,t)\lambda - M(q)\dot{G}^T(q,t)\eta \\ &\quad - M(q)G^T(q,t)\dot{\eta}, \end{aligned} \tag{7.19a}$$

$$0 = g(q,t), \tag{7.19b}$$

$$0 = g^I(q,\dot{q}+G^T(q,t)\eta,t), \tag{7.19c}$$

with the new algebraic variables η and $g^I(q,\dot{q}+G^T(q,t)\eta,t) = G(q,t)\dot{q} + G(q,t)G^T(q,t)\eta + g_t(q,t)$.

Theorem 7.2.1. *If $M(q)$ is regular and $G(q,t)$ has full rank then the GGL-formulation* (7.19) *forms a d-index 2 system and any solution (q,λ,η) of* (7.19) *has $\eta = 0$ and (q,λ) is a solution of the original equations of motion* (7.15). *Conversely, if (q,λ) is a solution of the equations of motion* (7.15) *then $(q,\lambda,0)$ is a solution of the GGL-formulation* (7.19).

Proof. To determine the d-index of system (7.19) we consider the corresponding first order system

$$\dot{q} = v, \tag{7.20a}$$

$$\begin{aligned} M(q)\dot{v} = f(q, v + G^T(q,t)\eta, \lambda, t) - G^T(q,t)\lambda - M(q)\dot{G}^T(q,t)\eta \\ - M(q)G^T(q,t)\dot{\eta}, \end{aligned} \tag{7.20b}$$

$$0 = g(q,t), \tag{7.20c}$$

$$0 = g^I(q, v + G^T(q,t)\eta, t). \tag{7.20d}$$

Differentiating (7.20c) and (7.20d) once we get

$$\frac{d}{dt}g(q,t) = G(q,t)v + g_t(q,t) = 0, \tag{7.21}$$

and

$$\begin{aligned} \frac{d}{dt}g^I(q, v + G^T(q,t)\eta, t) &= \frac{d}{dt}\left[G(q,t)(v + G^T(q,t)\eta) + g_t(q,t)\right] \\ &= 2\frac{d}{dt}G(q,t)v + G(q,t)\dot{v} + \frac{d}{dt}(G(q,t)G^T(q,t))\eta \\ &\quad + G(q,t)G^T(q,t)\dot{\eta} + g_{tt}(q,t) = 0. \end{aligned} \tag{7.22}$$

Using (7.20b) to replace $\dot{v}$ in (7.22) we get (omitting the dependencies)

$$2\dot{G}v + GM^{-1}f - GM^{-1}G^T\lambda - G\dot{G}^T\eta + \frac{d}{dt}(GG^T)\eta + g_{tt} = 0. \tag{7.23}$$

Since $GM^{-1}G^T$ is nonsingular we can use (7.23) to obtain a representation of λ as function of q, v and η. One more differentiation of (7.21) and (7.23) yields

$$\frac{d^2}{dt^2}g(q,t) = 2\dot{G}v + GM^{-1}f - GM^{-1}G^T\lambda - G\dot{G}^T\eta - GG^T\dot{\eta} + g_{tt} = 0, \tag{7.24}$$

and

$$\begin{aligned} \frac{d^2}{dt^2}g^I(q, v + G^T(q,t)\eta, t) &= 3\ddot{G}v + 3\dot{G}M^{-1}f - 3\dot{G}M^{-1}G^T\lambda - \dot{G}\dot{G}^T\eta \\ &\quad - \dot{G}G^T\dot{\eta} + \ddot{G}G^T\eta + G\frac{d}{dt}(M^{-1}f) - GM^{-1}G^T\dot{\lambda} \\ &\quad - GM^{-1}\dot{G}^T\lambda + g_{ttt} = 0. \end{aligned} \tag{7.25}$$

As GG^T is nonsingular we can obtain explicit representations for $\dot{\lambda}$ and $\dot{\eta}$ using (7.24) and (7.25). Hence, the d-index of the GGL-formulation (7.19) is 2.

To show the equivalence of the two systems we suppose that $[q, \lambda, \eta]$ is a solution of the GGL-formulation (7.19). Differentiating (7.19b) we get

$$\frac{d}{dt}g(q,t) = G(q,t)\dot{q} + g_t(q,t) = 0. \tag{7.26}$$

Subtracting (7.26) from (7.19c) we obtain

$$G(q,t)G^T(q,t)\eta = 0, \tag{7.27}$$

which implies that $\eta = 0$ as $G(q,t)G^T(q,t)$ is nonsingular. Hence, $[q,\lambda]$ is also solution of (7.15). Suppose that $[q,\lambda]$ is a solution of the equations of motion (7.15). Differentiating the constraints in (7.15) we get

$$\frac{d}{dt}g(q,t) = G(q,t)\dot{q} + g_t(q,t) = 0. \tag{7.28}$$

Thus, $[q,\lambda,0]$ is a solution of the GGL-formulation (7.19). □

7.2.2 Index Reduction by Minimal Extension

To obtain the minimally extended strangeness-free system for the second order equations of motion (7.15) we use similar techniques as for the first order system. In the following we assume that M is diagonal. We consider the reduced derivative array (7.18) in the second order form and determine a permutation matrix Π, such that

$$G(q,t)\Pi = [G_1 \quad G_2], \tag{7.29}$$

where G_2 is square and nonsingular. Next we partition

$$\Pi^T q = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}, \quad \Pi^T \dot{q} = \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} \tag{7.30}$$

and replace every occurrence of $\dot{p}_2$ by the new variable $w_1 \in \mathbb{R}^{n_\lambda}$ and every occurrence of $\ddot{p}_2$ by the new variable $w_2 \in \mathbb{R}^{n_\lambda}$. We then get the extended system as

$$\begin{aligned} M\Pi \begin{bmatrix} \ddot{p}_1 \\ w_2 \end{bmatrix} &= f(p,\dot{p},\lambda,t) - G^T(p,t)\lambda && \in \mathbb{R}^{n_q}, \\ 0 &= g(p,t) && \in \mathbb{R}^{n_\lambda}, \\ 0 &= g^I(p,\dot{p},t) && \in \mathbb{R}^{n_\lambda}, \\ 0 &= g^{II}(p,\dot{p},\lambda,t) && \in \mathbb{R}^{n_\lambda}, \end{aligned} \tag{7.31}$$

where $p = \Pi \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$, $\dot{p} = \Pi \begin{bmatrix} \dot{p}_1 \\ w_1 \end{bmatrix}$ and $\ddot{p} = \Pi \begin{bmatrix} \ddot{p}_1 \\ w_2 \end{bmatrix}$ and g^I, g^{II} can be written as

$$g^I(p,\dot{p},t) = G(p,t)\Pi \begin{bmatrix} \dot{p}_1 \\ w_1 \end{bmatrix} + g_t(p,t),$$

$$g^{II}(p,\dot{p},\lambda,t) = 2\frac{d}{dt}G(p,t)\Pi \begin{bmatrix} \dot{p}_1 \\ w_1 \end{bmatrix} + G(p,t)M^{-1}\left[f(p,\dot{p},\lambda,t) - G^T(p,t)\lambda\right] + g_{tt}(p,t).$$

Example 7.2.2. *Consider the following equations of motion*

$$\begin{aligned} \ddot{q}_1 &= 2\lambda q_1, \\ \ddot{q}_2 &= 2\lambda q_2, \\ \ddot{q}_3 &= -\lambda - 1, \\ 0 &= q_1^2 + q_2^2 - q_3. \end{aligned} \tag{7.32}$$

Differentiation of the constraints yields

$$2q_1\dot{q}_1 + 2q_2\dot{q}_2 - \dot{q}_3 = 0, \tag{7.33}$$

and differentiating once more and eliminating the second derivatives with the help of (7.32) *we obtain*

$$2\dot{q}_1^2 + 4\lambda q_1^2 + 4\lambda q_2^2 + 2\dot{q}_2^2 + \lambda + 1 = 0. \tag{7.34}$$

Adding (7.33) *and* (7.34) *to the equations of motion and replacing* $\dot{q}_3$ *by* w_1 *and* $\ddot{q}_3$ *by* w_2*, we obtain the minimally extended strangeness-free system as*

$$\begin{aligned}
\ddot{q}_1 &= 2\lambda q_1, \\
\ddot{q}_2 &= 2\lambda q_2, \\
w_2 &= -\lambda - 1, \\
0 &= q_1^2 + q_2^2 - q_3, \\
0 &= 2q_1\dot{q}_1 + 2q_2\dot{q}_2 - w_1, \\
0 &= 2\dot{q}_1^2 + 4\lambda q_1^2 + 4\lambda q_2^2 + 2\dot{q}_2^2 + \lambda + 1.
\end{aligned} \tag{7.35}$$

A reduced strangeness-free system is achieved by omitting the equations that involve w_1 *and* w_2*, i.e., the third and the fifth equation in* (7.35).

Theorem 7.2.3. *Let* M *be positive definite and diagonal and let* $G(q,t)$ *have full row rank. Then the extended system* (7.31) *has d-index 1 (is strangeness-free).*

Proof. To show that system (7.31) has d-index 1 we consider the corresponding first order system. The first order system corresponding to the reduced derivative array (7.18) is given by

$$\begin{aligned}
\dot{q} &= v, \\
M\Pi \begin{bmatrix} \dot{v}_1 \\ w_2 \end{bmatrix} &= f(q,v,\lambda,t) - G^T(q,t)\lambda, \\
0 &= g(q,t), \\
0 &= g^I(q,v,t), \\
0 &= g^{II}(q,v,\lambda,t).
\end{aligned} \tag{7.36}$$

Multiplying the first equation with Π yields the first order system

$$\begin{aligned}
\dot{q}_1 &= v_1, \\
w_1 &= v_2, \\
M\Pi \begin{bmatrix} \dot{v}_1 \\ w_2 \end{bmatrix} &= f(q,v,\lambda,t) - G^T(q,t)\lambda, \\
0 &= g(q,t), \\
0 &= g^I(q,v,t), \\
0 &= g^{II}(q,v,\lambda,t).
\end{aligned} \tag{7.37}$$

This system is equivalent to system (7.13). In [22] it is shown that (7.13) is strangeness-free. Thus, system (7.31) has d-index 1. □

Chapter 8

Numerical Examples

In this chapter we want to demonstrate for some test problems that the previously discussed methods yield better results for the second order systems, than the numerical solution of the corresponding first order systems. For that we compare the behavior of our modified BDF methods for second order systems with the DAE solver DASSL [4] for the corresponding first order systems. In addition we compare the solutions obtained for the original formulation of the equations of motion, the GGL-formulation and the strangeness-free formulation in the first and second order form, respectively, as formulations of d-index 3, 2 and 1. In the second part we examine the results obtained by the Runge-Kutta method for second order DAEs.

8.1 Examples for Modified BDF Methods

In this section we consider our modified BDF methods and compare the results we obtain with the results obtained by the code DASSL for two examples and different formulations of the equations of motion. DASSL is an integrator to solve first order differential-algebraic systems and is actually designed for the solution of d-index 0 and d-index 1 systems. We will nonetheless use it to solve d-index 3 systems, even though we do not expect reliable solutions in order to show where a code based on ordinary BDF methods fail.

8.1.1 Example 1

In this example taken from [32] we consider a mechanical system consisting of just one body of mass 1 in a two-dimensional space, which is moving on the unit circle. The equations of motion are given by

$$\begin{aligned} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}}_{\ddot{q}} &= \underbrace{\begin{bmatrix} -q_1 - 2q_1\dot{q}_1\dot{q}_2 \\ -\dot{q}_1 + 2q_1 q_2^2 \end{bmatrix}}_{f_a} - \underbrace{\begin{bmatrix} 2q_1 \\ 2q_2 \end{bmatrix}}_{G^T} \lambda, \\ 0 &= \underbrace{q_1^2 + q_2^2 - 1}_{g(q)}, \end{aligned} \tag{8.1}$$

with the initial values

$$t_0 = 0, \quad q(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \dot{q}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \lambda(0) = 0. \tag{8.2}$$

The exact solution of system (8.1) is given by

$$q(t) = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}, \quad \dot{q}(t) = \begin{bmatrix} \cos(t) \\ -\sin(t) \end{bmatrix}, \quad \lambda(t) = \sin(t)\cos(t). \tag{8.3}$$

In the following we give the index-reduced formulations in the second and first order form as described in Chapter 7 in order to compare the performance of our code for problems of different d-index.
The second order GGL-formulation of system (8.1) is given by

$$\begin{aligned} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} &= \begin{bmatrix} -q_1 - 2q_1(\dot{q}_1 + 2q_1\eta)(\dot{q}_2 + 2q_2\eta) \\ -(\dot{q}_1 + 2q_1\eta) + 2q_1q_2^2 \end{bmatrix} - \begin{bmatrix} 2q_1 \\ 2q_2 \end{bmatrix} \lambda \\ &\quad - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\dot{q}_1 \\ 2\dot{q}_2 \end{bmatrix} \eta - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2q_1 \\ 2q_2 \end{bmatrix} \dot{\eta}, \\ 0 &= q_1^2 + q_2^2 - 1, \\ 0 &= 2q_1(\dot{q}_1 + 2q_1\eta) + 2q_2(\dot{q}_2 + 2q_2\eta) \end{aligned} \tag{8.4}$$

and for the second order strangeness-free formulation based on minimal extension as described in Section 7.2.2 we obtain

$$\begin{aligned} \ddot{q}_1 &= -q_1 - 2q_1\dot{q}_1w_1 - 2q_1\lambda, \\ w_2 &= -\dot{q}_1 + 2q_1q_2^2 - 2q_2\lambda, \\ 0 &= q_1^2 + q_2^2 - 1, \\ 0 &= 2q_1\dot{q}_1 + 2q_2w_1, \\ 0 &= 2\dot{q}_1^2 - 2q_1(q_1 + 2q_1\dot{q}_1w_1 + 2q_1\lambda) + 2w_1^2 - 2q_2(\dot{q}_1 - 2q_1q_2^2 + 2q_2\lambda), \end{aligned} \tag{8.5}$$

if $q_2 \neq 0$ and

$$\begin{aligned} w_2 &= -q_1 - 2q_1w_1\dot{q}_2 - 2q_1\lambda, \\ \ddot{q}_2 &= -w_1 + 2q_1q_2^2 - 2q_2\lambda, \\ 0 &= q_1^2 + q_2^2 - 1, \\ 0 &= 2q_1w_1 + 2q_2\dot{q}_2, \\ 0 &= 2w_1^2 - 2q_1(q_1 + 2q_1w_1\dot{q}_2 + 2q_1\lambda) + 2\dot{q}_2^2 - 2q_2(w_1 - 2q_1q_2^2 + 2q_2\lambda), \end{aligned} \tag{8.6}$$

if $q_1 \neq 0$.

Order reduction by introducing new variables for the velocities yields the corresponding first order form of system (8.1)

$$\begin{aligned} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} &= \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} &= \begin{bmatrix} -q_1 - 2q_1v_1v_2 \\ -v_1 + 2q_1q_2^2 \end{bmatrix} - \begin{bmatrix} 2q_1 \\ 2q_2 \end{bmatrix} \lambda, \\ 0 &= q_1^2 + q_2^2 - 1. \end{aligned} \tag{8.7}$$

For system (8.7) we obtain the first order GGL-formulation as

$$\begin{aligned}
\begin{bmatrix}\dot q_1\\ \dot q_2\end{bmatrix} &= \begin{bmatrix}v_1\\ v_2\end{bmatrix} - \begin{bmatrix}2q_1\\ 2q_2\end{bmatrix}\eta,\\
\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\dot v_1\\ \dot v_2\end{bmatrix} &= \begin{bmatrix}-q_1-2q_1v_1v_2\\ -v_1+2q_1q_2^2\end{bmatrix} - \begin{bmatrix}2q_1\\ 2q_2\end{bmatrix}\lambda,\\
0 &= q_1^2+q_2^2-1,\\
0 &= 2q_1v_1+2q_2v_2
\end{aligned}\tag{8.8}$$

and the first order strangeness-free formulation of (8.7) based on minimal extension as described in Section 7.1.2 is given by

$$\begin{aligned}
\dot q_1 &= v_1,\\
w_1 &= v_2,\\
\dot v_1 &= -q_1-2q_1v_1v_2-2q_1\lambda,\\
w_2 &= -v_1+2q_1q_2^2-2q_2\lambda,\\
0 &= q_1^2+q_2^2-1,\\
0 &= 2q_1v_1+2q_2v_2,\\
0 &= 2v_1^2-2q_1(q_1+2q_1v_1v_2+2q_1\lambda)+2v_2^2-2q_2(v_1-2q_1q_2^2+2q_2\lambda)
\end{aligned}\tag{8.9}$$

if $q_2 \neq 0$. Omitting the equations that involve w_1 and w_2, i.e., the second and the forth equation in (8.9) we get the reduced strangeness-free formulation of (8.7), if $q_2 \neq 0$ as

$$\begin{aligned}
\dot q_1 &= v_1,\\
\dot v_1 &= -q_1-2q_1v_1v_2-2q_1\lambda,\\
0 &= q_1^2+q_2^2-1,\\
0 &= 2q_1v_1+2q_2v_2,\\
0 &= 2v_1^2-2q_1(q_1+2q_1v_1v_2+2q_1\lambda)+2v_2^2-2q_2(v_1-2q_1q_2^2+2q_2\lambda).
\end{aligned}\tag{8.10}$$

In the other case that $q_1 \neq 0$ we get the following reduced strangeness-free formulation of (8.7)

$$\begin{aligned}
\dot q_2 &= v_2,\\
\dot v_2 &= -v_1+2q_1q_2^2-2q_2\lambda,\\
0 &= q_1^2+q_2^2-1,\\
0 &= 2q_1v_1+2q_2v_2,\\
0 &= 2v_1^2-2q_1(q_1+2q_1v_1v_2+2q_1\lambda)+2v_2^2-2q_2(v_1-2q_1q_2^2+2q_2\lambda).
\end{aligned}\tag{8.11}$$

In the following the second order systems are solved by our modified BDF methods of order 1 and 2, while the first order systems are solved by DASSL, each using the consistent initial values (8.2). The integration is performed on the interval $[t_0, t_0+T]$, where $t_0 = 0$ and $T = 1$. Table 8.1 displays the results for the original equations of motion (8.1) and (8.7) in the first and second order form.
On the left side the results for the original second order equations of motion (8.1) obtained with the modified BDF methods are displayed and on the right side

	System (8.1), mod. BDF			System (8.7), DASSL		
time	Absolute errors		No.	Absolute errors		No.
t	$\|q_1(t_j) - q_{1,j}\|$	$\|\lambda(t_j) - \lambda_j\|$	Steps	$\|q_1(t_j) - q_{1,j}\|$	$\|\lambda(t_j) - \lambda_j\|$	Steps
0.1	$3.21 \cdot 10^{-8}$	$9.46 \cdot 10^{-4}$	13	$6.90 \cdot 10^{-5}$	$2.27 \cdot 10^{-1}$	17
0.2	$1.60 \cdot 10^{-6}$	$6.38 \cdot 10^{-4}$	14	$6.81 \cdot 10^{-7}$	$9.93 \cdot 10^{-2}$	23
0.3	$1.40 \cdot 10^{-5}$	$2.73 \cdot 10^{-3}$	15	$7.29 \cdot 10^{-5}$	$3.93 \cdot 10^{-2}$	29
0.4	$3.41 \cdot 10^{-5}$	$2.64 \cdot 10^{-3}$	16	$3.59 \cdot 10^{-4}$	$8.60 \cdot 10^{-2}$	35
0.5	$6.98 \cdot 10^{-5}$	$7.34 \cdot 10^{-4}$	17	$8.62 \cdot 10^{-4}$	$1.59 \cdot 10^{-3}$	39
0.6	$1.25 \cdot 10^{-4}$	$3.45 \cdot 10^{-3}$	18	$1.42 \cdot 10^{-3}$	$2.91 \cdot 10^{-3}$	43
0.7	$1.91 \cdot 10^{-4}$	$2.00 \cdot 10^{-3}$	21	$1.88 \cdot 10^{-3}$	$3.66 \cdot 10^{-2}$	47
0.8	$2.61 \cdot 10^{-4}$	$1.64 \cdot 10^{-3}$	23	$2.27 \cdot 10^{-3}$	$2.63 \cdot 10^{-2}$	51
0.9	$3.31 \cdot 10^{-4}$	$3.76 \cdot 10^{-3}$	25	$2.62 \cdot 10^{-3}$	$5.88 \cdot 10^{-2}$	55
1.0	$3.96 \cdot 10^{-4}$	$6.13 \cdot 10^{-3}$	26	$2.90 \cdot 10^{-3}$	$3.19 \cdot 10^{-2}$	59

Table 8.1: Results for original equations of motion

	System (8.4), mod. BDF			System (8.8), DASSL		
time	Absolute errors		No.	Absolute errors		No.
t	$\|q_1(t_j) - q_{1,j}\|$	$\|\lambda(t_j) - \lambda_j\|$	Steps	$\|q_1(t_j) - q_{1,j}\|$	$\|\lambda(t_j) - \lambda_j\|$	Steps
0.1	$3.80 \cdot 10^{-5}$	$8.46 \cdot 10^{-1}$	10	$5.40 \cdot 10^{-6}$	$6.25 \cdot 10^{-5}$	51
0.2	$3.04 \cdot 10^{-3}$	$6.47 \cdot 10^{0}$	11	$2.29 \cdot 10^{-5}$	$1.50 \cdot 10^{-4}$	93
0.3	$1.10 \cdot 10^{-2}$	$2.18 \cdot 10^{-1}$	12	$5.20 \cdot 10^{-5}$	$2.68 \cdot 10^{-4}$	136
0.4	$2.65 \cdot 10^{-2}$	$1.06 \cdot 10^{0}$	14	$9.20 \cdot 10^{-5}$	$4.34 \cdot 10^{-4}$	178
0.5	$5.52 \cdot 10^{-2}$	$2.53 \cdot 10^{0}$	17	$1.43 \cdot 10^{-4}$	$6.66 \cdot 10^{-4}$	221
0.6	$1.06 \cdot 10^{-1}$	$5.04 \cdot 10^{0}$	21	$2.02 \cdot 10^{-4}$	$9.86 \cdot 10^{-4}$	263
0.7	$1.99 \cdot 10^{-1}$	$1.07 \cdot 10^{1}$	34	$2.70 \cdot 10^{-4}$	$1.41 \cdot 10^{-3}$	306
0.8	$2.70 \cdot 10^{-1}$	$3.27 \cdot 10^{1}$	64	$3.42 \cdot 10^{-4}$	$1.94 \cdot 10^{-3}$	349
0.9	$3.50 \cdot 10^{-1}$	$3.65 \cdot 10^{1}$	89	$4.13 \cdot 10^{-4}$	$2.54 \cdot 10^{-3}$	391
1.0	$1.27 \cdot 10^{0}$	$3.71 \cdot 10^{1}$	107	$4.74 \cdot 10^{-4}$	$3.11 \cdot 10^{-3}$	433

Table 8.2: Results for GGL-formulations

	System (8.5), mod. BDF			System (8.10), DASSL		
time	Absolute errors		No.	Absolute errors		No.
t	$\|q_1(t_j) - q_{1,j}\|$	$\|\lambda(t_j) - \lambda_j\|$	Steps	$\|q_1(t_j) - q_{1,j}\|$	$\|\lambda(t_j) - \lambda_j\|$	Steps
0.1	$2.65 \cdot 10^{-8}$	$3.17 \cdot 10^{-4}$	13	$3.37 \cdot 10^{-5}$	$3.75 \cdot 10^{-5}$	10
0.2	$1.41 \cdot 10^{-6}$	$1.37 \cdot 10^{-3}$	14	$1.15 \cdot 10^{-4}$	$1.21 \cdot 10^{-4}$	13
0.3	$7.24 \cdot 10^{-6}$	$4.37 \cdot 10^{-4}$	16	$1.85 \cdot 10^{-4}$	$1.66 \cdot 10^{-4}$	16
0.4	$1.85 \cdot 10^{-5}$	$2.08 \cdot 10^{-3}$	17	$1.91 \cdot 10^{-4}$	$1.34 \cdot 10^{-4}$	18
0.5	$4.04 \cdot 10^{-5}$	$8.01 \cdot 10^{-4}$	19	$1.55 \cdot 10^{-4}$	$1.42 \cdot 10^{-5}$	20
0.6	$7.05 \cdot 10^{-5}$	$2.50 \cdot 10^{-3}$	21	$1.26 \cdot 10^{-4}$	$1.81 \cdot 10^{-4}$	22
0.7	$1.11 \cdot 10^{-4}$	$1.47 \cdot 10^{-3}$	23	$1.22 \cdot 10^{-4}$	$8.78 \cdot 10^{-5}$	24
0.8	$1.55 \cdot 10^{-4}$	$1.67 \cdot 10^{-3}$	25	$1.34 \cdot 10^{-4}$	$1.44 \cdot 10^{-4}$	26
0.9	$2.00 \cdot 10^{-4}$	$2.90 \cdot 10^{-3}$	27	$1.40 \cdot 10^{-4}$	$1.39 \cdot 10^{-3}$	28
1.0	$2.38 \cdot 10^{-4}$	$3.07 \cdot 10^{-3}$	29	$1.12 \cdot 10^{-4}$	$9.37 \cdot 10^{-4}$	30

Table 8.3: Results for strangeness-free formulations

the results for the corresponding first order system (8.7) solved with DASSL. The results are the absolute errors in the variables q_1 and λ at different interpolated time steps together with the number of steps taken so far. In Table 8.2 the corresponding results for the GGL-formulations (8.4) and (8.8) in the first and second order form are displayed and in Table 8.3 the results for the strangeness-free formulations (8.5) and (8.10) based on minimal extension.

We can observe that the modified BDF methods work well for the original equations of motion (8.1) with order 2 and d-index 3. The results for the modified BDF methods given in Table 8.1 are obtained using the tolerances $RTOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$ and $ATOL = [10^{-4}, 10^{-4}, 10^{-2}, 10^{-2}, 10^{-2}]$ for the variables $[q_1, q_2, \lambda, \dot{q}_1, \dot{q}_2]$. We can observe that the modified BDF methods fulfill the desired error tolerances. The absolute errors in the position coordinate q_1 are of order 10^{-4} (in the beginning of the integration even better as the integration is started with smaller stepsizes), while we observe a reduction of the accuracy of about a factor of 10 in the algebraic variable λ. The modified BDF methods require 26 steps to reach the end of the integration interval.
As expected, DASSL applied to the original equations of motion in the first order form does not yield satisfactory results for the algebraic variable λ due to the higher index. The results for DASSL were computed with the tolerances $ATOL = [10^{-4}, 10^{-4}, 10^{-2}, 10^{-2}, 1]$ and $RTOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$ for the variables $[q_1, q_2, v_1, v_2, \lambda]$. Here, we have to choose the absolute tolerance 1 for the algebraic variable λ, as the code could not converge otherwise, due to the higher index. DASSL cannot reach the desired error tolerances for q_1 and requires 59 steps to reach the end of the interval, i.e., twice as much steps as the modified BDF methods. The results for the position coordinate q_1 and the algebraic variable λ have both a reduced accuracy of factor 10 when compared to the results obtained with the modified BDF methods.
The solution behavior for the algebraic variable is illustrated in Figure 8.1 and the absolute errors for the modified BDF methods and for DASSL are shown in Figure 8.2. We can see, that the solution obtained with the modified BDF methods seems to be a good approximation to the exact solution, while the solution obtain with DASSL does not coincide with the exact solution. We observe high errors for the results obtained with DASSL, while the errors for the modified BDF methods are small.

For the GGL-formulation the modified BDF methods work not so well as for the original equations of motion. The results in Table 8.2 were computed with the tolerances $ATOL = [10^{-3}, 10^{-3}, 10^{-2}, 1, 1, 1, 1]$ and $RTOL = [10^{-2}, 10^{-2}, 10^{-1}, 1, 1, 1, 1]$ for the variables $[q_1, q_2, \eta, \lambda, \dot{q}_1, \dot{q}_2, \dot{\eta}]$. We can observe high errors in the algebraic variable λ and the solution for the position coordinates q_1 deviates form the exact solution with increasing time due to the influence of the algebraic variable. The modified BDF methods require 107 steps to reach the end of the integration interval. We choose the above tolerances in order to obtain the best results. Using stricter tolerances causes the code to exit, because the Newton iteration fails to converge repeatedly. The reason for this behavior is a very badly scaled and ill-conditioned iteration matrix, which is close to singular for small stepsizes. Also small perturbations in the data might be the reason for this behavior, as the derivative of the algebraic variable η occurs in the second order GGL-formulation. Further investigations of the behavior of the second or-

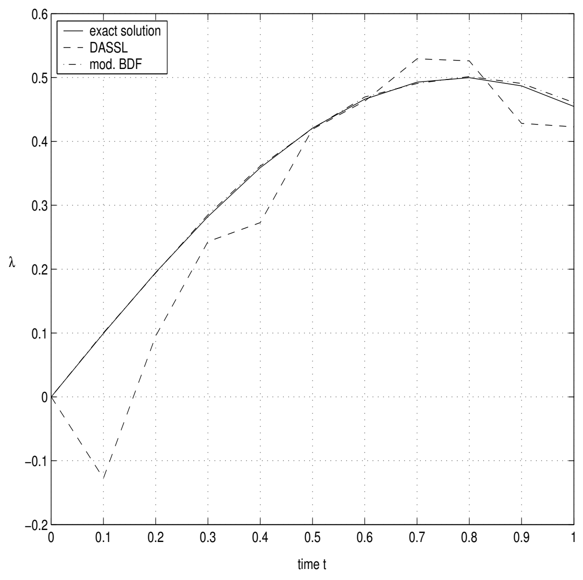

Figure 8.1: Original equations of motion, solutions for λ

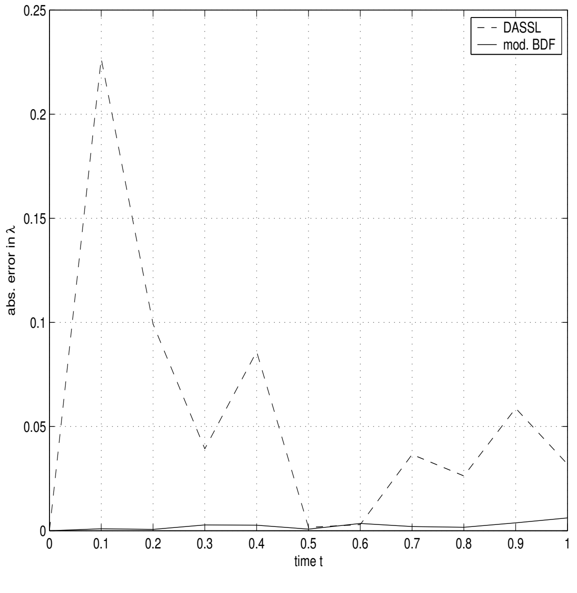

Figure 8.2: Original equations of motion, absolute errors in λ

der GGL-formulation under small perturbations are needed to understand this behavior.
Using DASSL we can obtain better results using the tolerances $ATOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$ and $RTOL = ATOL$ for the variables $[q_1, q_2, v_1, v_2, \lambda, \eta]$, but we can also observe difficulties in this case. For the position coordinates we observe an error behavior of 10^{-4}, but the accuracy for the algebraic variable λ is still beneath the accuracy of the other position coordinates and beneath the desired error tolerance. DASSL requires very small stepsizes to obtain these results, as we can see at the large number of steps taken. These difficulties arise because the first order GGL-formulation is of d-index 2, i.e., still of higher index.

For the strangeness-free formulations the modified BDF methods and DASSL yield similar results, as both codes reach almost the same accuracy of about 10^{-4} for the position coordinates q_1 and require almost the same number of steps. To obtain the results in Table 8.3 we have used the tolerances $RTOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$ and $ATOL = [10^{-4}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-2}, 10^{-2}]$ for the modified BDF methods with variables $[q_1, q_2, w_1, w_2, \lambda, \dot{q}_1]$ and $ATOL = [10^{-4}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-2}]$, $RTOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$ for DASSL with variables $[q_1, q_2, v_1, v_2, \lambda]$. We can see that the results obtained with DASSL fulfill the desired error tolerances for both, the position variable q_1 and the algebraic variable λ. This behavior is just as we have expected for the strangeness-free formulation. For the results obtained with the modified BDF methods we can observe a reduction of the accuracy in the algebraic variable of a factor of 10. The results for the modified BDF methods are almost the same as the results obtained for the original second order equations of motion displayed in Table 8.1. We can see no improve in the accuracy for the strangeness-free formulation. This is due to the fact that the modified BDF methods were designed for the numerical solution of differential-algebraic equations of order 2 and d-index $\nu \in \{2, 3\}$ and thus yield already good results for the original equations of motion.

We chose a rather simple implementation of the modified BDF methods. Using better error estimates and stepsize selection strategies will improve the results. Further, we have only implemented methods up to order 2. Higher order methods may also lead to better results.
The problem of singular or ill-conditioned iteration matrices is treated in [25]. Scaling the rows of the iteration matrix corresponding to the algebraic variables by the factor $1/h$ as suggested in [25] may further improve the results.

8.1.2 Example 2

The second example is a planar vertical model of a 2-D truck with nonlinear suspension introduced in [31]. The dimension and the bad scaling of the model are typical for problems in vehicle dynamics. Figure 8.3 shows the model of the truck. It forms a multibody system with seven bodies, linear and nonlinear suspension elements, and a kinematic joint. The DAE consists of 12 unknowns, i.e., 11 position coordinates and one Lagrange multiplier function. The truck proceeds with constant speed v_0 and external forces are the driveway excitation $u(t)$ and the gravitational force. As initial values for $t_0 = 0$ we took

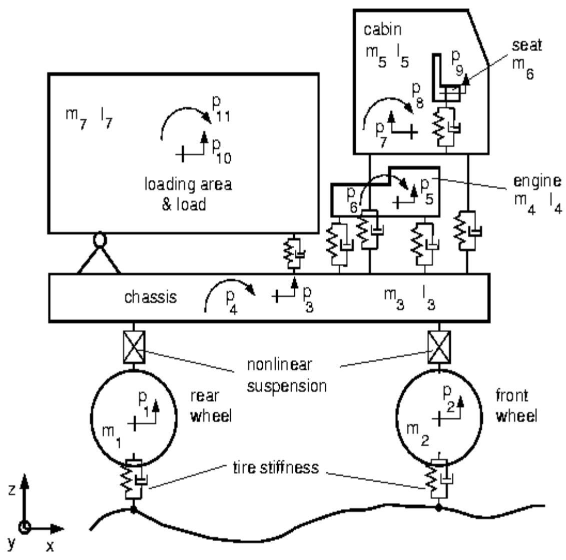

Figure 8.3: Truck

$y_0 = [0, \ldots, 0]^T$.
To obtain the numerical results we consider the time interval $[t_0, t_0+T]$ with $T = 20$. The equations of motion in the second order form are solved by the modified BDF methods using the tolerances $ATOL = [10^{-4}, ..., 10^{-4}, 10^{-2}, 10^{-3}, ..., 10^{-3}]$ and $RTOL = [10^{-3}, ..., 10^{-3}, 1, 10^{-2}, ..., 10^{-2}]$ for the variables $[p_1, ..., p_{11}, \lambda,$ $\dot{p}_1, ..., \dot{p}_{11}]$ and the corresponding first order system is solved with DASSL using the tolerances $ATOL = [10^{-4}, ..., 10^{-4}, 10^{-2}]$ and $RTOL = [10^{-3}, ..., 10^{-3}, 1]$ (the last variable is the algebraic variable λ). While the original system used in the modified BDF methods is of dimension 12×12 the first order system used in the code DASSL is of dimension 23×23, i.e., twice as large as the second order system.
Figure 8.4 shows the vertical motion of the drivers seat (position coordinate p_9) as response to the excitation $u(t)$ as a result obtained by our modified BDF methods and Figure 8.6 shows the results for the algebraic variable λ. The results for the corresponding first order system solved with DASSL are displayed in Figure 8.5 and Figure 8.7, where we have restricted the maximal stepsize to be used as 0.1.

We do not know the exact solution of the system, but we can expect that the solution behavior is related to the excitation $u(t)$. We can see that the results we obtain for the coordinate p_9 with the modified BDF methods and with DASSL seem to be similar and are the direct response to the excitation $u(t)$. On the other hand the results obtained for the algebraic variable λ are quite different. If the system goes back to its equilibrium state after excitation we can expect λ to approach zero. We can observe a decreasing solution for λ obtained with the modified BDF methods, while we observe oscillations around a constant value in the solution obtained with DASSL. Thus, we can deduce that the solution

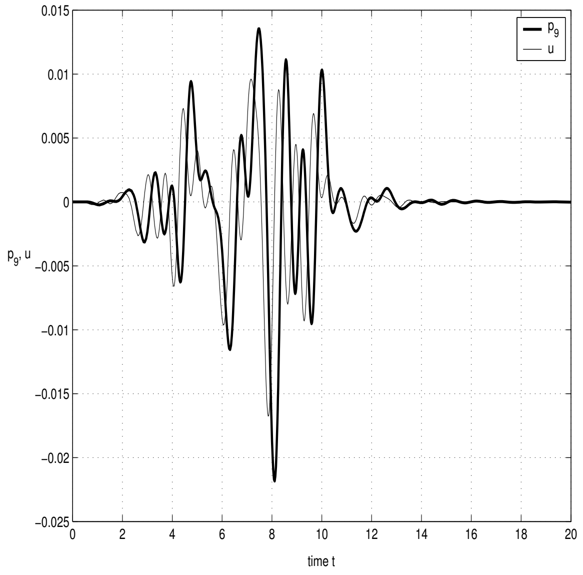

Figure 8.4: Solution p_9 for Truck solved with mod. BDF

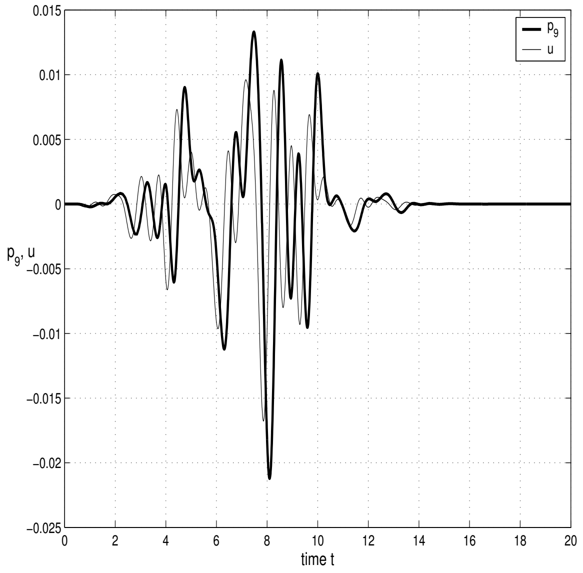

Figure 8.5: Solution p_9 for Truck solved with DASSL

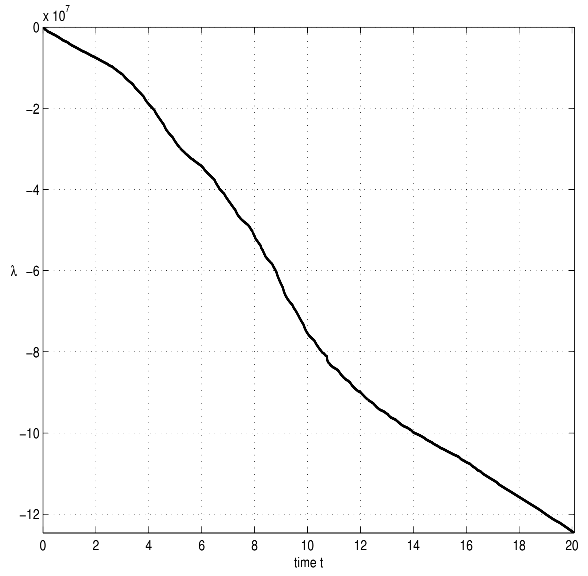

Figure 8.6: Solution λ for Truck solved with mod. BDF

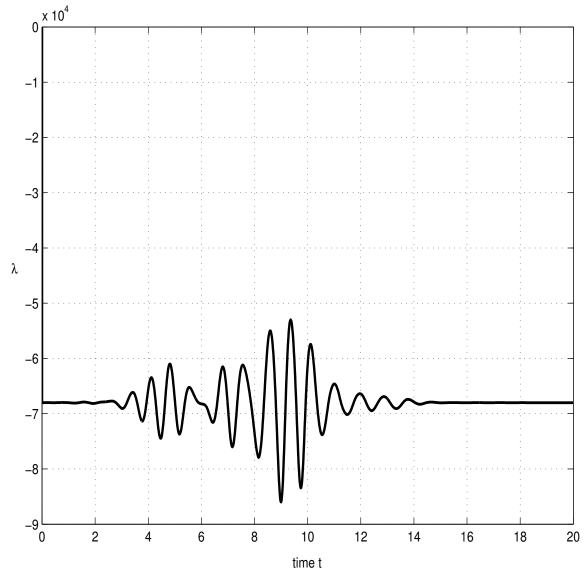

Figure 8.7: Solution λ for Truck solved with DASSL

behavior for the solution obtained with the modified BDF methods is closer to the expected behavior.
We know that DASSL has problems to solve this example due to the higher index and therefore we cannot expect to obtain reliable results for λ. The oscillations in the algebraic variable λ in Figure 8.7, which do not occur in the case of the modified BDF methods are an indicator for this. The tolerances for DASSL are chosen such that we obtain the best results for the position coordinates not regarding the accuracy of λ, otherwise the code could not converge. While the modified BDF methods need 526 steps to reach the end of the integration interval, DASSL requires 1268 steps, if we do not restrict the maximal stepsize, i.e., again twice as much as the modified BDF methods similar to the example before.

8.2 Example for Runge-Kutta Method

To examine the performance of the Runge-Kutta method given in Chapter 6 we apply it to the example given in Section 8.1.1. The original second order equations of motion (8.1) are integrated with the 2-stage Runge-Kutta method (6.15) with Radau-IIA coefficients given in Table 6.1 on the interval $[t_0, t_0 + T]$ with $t_0 = 0$, $T = 1$ and the consistent initial values (8.2). We use the tolerances $RTOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]^T$ and $ATOL = [10^{-4}, 10^{-4}, 10^{-2}, 10^{-2}, 10^{-2}, 10^{-1}]^T$ for the variables $[q_1, q_2, \lambda, \dot{q}_1, \dot{q}_2, \dot{\lambda}]$ corresponding to the tolerances used in the BDF methods.
In Table 8.4 the absolute errors for the variables q_1, q_2 and λ at certain interpolated points are displayed together with the number of steps taken and in Table 8.5 the corresponding results for the derivatives $\dot{q}_1$, $\dot{q}_2$ and $\dot{\lambda}$. The Runge-Kutta method needs 24 steps to reach the end of the integration interval, almost the same number of steps as the modified BDF methods.
We can observe absolute errors of order 10^{-6} for the variables q_1 and q_2 and absolute errors of order 10^{-4} in the algebraic variable λ. Again we can observe a reduction of the order of accuracy in the algebraic variable of about a factor 100. For the derivatives we can observe absolute errors of order 10^{-5} in $\dot{q}_1$ and $\dot{q}_2$ and of order 10^{-2} in $\dot{\lambda}$. Here, we can see an even greater reduction of the accuracy in $\dot{\lambda}$.

We can observe that we obtain better results with the Runge-Kutta method than using the modified BDF methods. This is due to the fact that the Runge-Kutta method is of order 4, while we have only implemented BDF methods of order 1 and 2. Therefore we can reach a higher accuracy using the Runge-Kutta method than using the modified BDF methods.
Nevertheless, we observe some problems when using the 2-stage Runge-Kutta method to solve certain test problems. Often the code fails due to a singular or ill-conditioned iteration matrix, that causes the Newton iteration to fail to converge repeatedly.
To see why this problem occurs so frequently, we write the nonlinear system

time	No.	Absolute errors		
t	Steps	$\lvert q_1(t_j) - q_{1,j}\rvert$	$\lvert q_2(t_j) - q_{2,j}\rvert$	$\lvert\lambda(t_j) - \lambda_j\rvert$
0.1	12	$4.08 \cdot 10^{-9}$	$3.75 \cdot 10^{-10}$	$2.32 \cdot 10^{-4}$
0.2	13	$1.57 \cdot 10^{-7}$	$2.98 \cdot 10^{-8}$	$9.71 \cdot 10^{-4}$
0.3	14	$9.82 \cdot 10^{-7}$	$2.85 \cdot 10^{-7}$	$1.87 \cdot 10^{-3}$
0.4	15	$1.82 \cdot 10^{-6}$	$7.66 \cdot 10^{-7}$	$8.83 \cdot 10^{-4}$
0.5	17	$3.00 \cdot 10^{-6}$	$1.64 \cdot 10^{-6}$	$5.04 \cdot 10^{-4}$
0.6	19	$4.18 \cdot 10^{-6}$	$2.86 \cdot 10^{-6}$	$2.64 \cdot 10^{-4}$
0.7	20	$5.36 \cdot 10^{-6}$	$4.51 \cdot 10^{-6}$	$5.92 \cdot 10^{-4}$
0.8	21	$6.55 \cdot 10^{-6}$	$6.74 \cdot 10^{-6}$	$1.38 \cdot 10^{-3}$
0.9	22	$7.67 \cdot 10^{-6}$	$9.65 \cdot 10^{-6}$	$1.14 \cdot 10^{-3}$
1.0	24	$8.62 \cdot 10^{-6}$	$1.34 \cdot 10^{-5}$	$1.03 \cdot 10^{-3}$

Table 8.4: Results for Runge-Kutta method for q_1, q_2, λ

time	No.	Absolute errors		
t	Steps	$\lvert\dot{q}_1(t_j) - \dot{q}_{1,j}\rvert$	$\lvert\dot{q}_2(t_j) - \dot{q}_{2,j}\rvert$	$\lvert\dot{\lambda}(t_j) - \dot{\lambda}_j\rvert$
0.1	12	$4.71 \cdot 10^{-7}$	$3.04 \cdot 10^{-6}$	$2.02 \cdot 10^{-2}$
0.2	13	$8.16 \cdot 10^{-6}$	$2.50 \cdot 10^{-5}$	$4.05 \cdot 10^{-2}$
0.3	14	$2.92 \cdot 10^{-5}$	$5.66 \cdot 10^{-5}$	$7.45 \cdot 10^{-2}$
0.4	15	$9.12 \cdot 10^{-6}$	$1.80 \cdot 10^{-5}$	$1.33 \cdot 10^{-1}$
0.5	17	$1.24 \cdot 10^{-5}$	$9.96 \cdot 10^{-6}$	$9.57 \cdot 10^{-2}$
0.6	19	$1.28 \cdot 10^{-5}$	$1.29 \cdot 10^{-5}$	$4.03 \cdot 10^{-2}$
0.7	20	$2.25 \cdot 10^{-5}$	$7.03 \cdot 10^{-6}$	$3.06 \cdot 10^{-2}$
0.8	21	$3.80 \cdot 10^{-5}$	$1.49 \cdot 10^{-6}$	$5.22 \cdot 10^{-2}$
0.9	22	$3.33 \cdot 10^{-5}$	$1.92 \cdot 10^{-5}$	$7.99 \cdot 10^{-2}$
1.0	24	$2.35 \cdot 10^{-5}$	$3.31 \cdot 10^{-5}$	$8.54 \cdot 10^{-2}$

Table 8.5: Results for Runge-Kutta method for $\dot{q}_1, \dot{q}_2, \dot{\lambda}$

(6.26) simplified as

$$F(X_n) = \begin{bmatrix} M\ddot{Q}_{n1} - f(Q_{n1}, \dot{Q}_{n1}, \Lambda_{n1}, t_n + c_1 h) \\ M\ddot{Q}_{n2} - f(Q_{n2}, \dot{Q}_{n2}, \Lambda_{n2}, t_n + c_2 h) \\ g(t_n + c_1 h, Q_{n1}) \\ g(t_n + c_2 h, Q_{n2}) \end{bmatrix} = 0. \tag{8.12}$$

Then the Jacobian has the form

$$DF(X_n) = \begin{bmatrix} M - h^2 f_{11} & -h^2 f_{12} & -h^2 \tilde{a}_{11} f^{(1)}_{\Lambda_{n1}} & -h^2 \tilde{a}_{12} f^{(1)}_{\lambda_{n2}} \\ -h^2 f_{21} & M - h^2 f_{22} & -h^2 \tilde{a}_{21} f^{(2)}_{\Lambda_{n1}} & -h^2 \tilde{a}_{22} f^{(2)}_{\Lambda_{n2}} \\ h^2 \tilde{a}_{11} g^{(1)}_{Q_{n1}} & h^2 \tilde{a}_{12} g^{(1)}_{Q_{n2}} & 0 & 0 \\ h^2 \tilde{a}_{21} g^{(2)}_{Q_{n1}} & h^2 \tilde{a}_{22} g^{(2)}_{Q_{n2}} & 0 & 0 \end{bmatrix}, \tag{8.13}$$

with

$$\begin{aligned} f_{11} &= \tilde{a}_{11} f^{(1)}_{Q_{n1}} - \frac{1}{h} a_{11} f^{(1)}_{\dot{Q}_{n1}}, \\ f_{12} &= \tilde{a}_{12} f^{(1)}_{Q_{n2}} - \frac{1}{h} a_{12} f^{(1)}_{\dot{Q}_{n2}}, \\ f_{21} &= \tilde{a}_{21} f^{(2)}_{Q_{n1}} - \frac{1}{h} a_{21} f^{(2)}_{\dot{Q}_{n1}}, \\ f_{22} &= \tilde{a}_{22} f^{(2)}_{Q_{n2}} - \frac{1}{h} a_{22} f^{(2)}_{\dot{Q}_{n2}}. \end{aligned}$$

We can see that (8.13) is close to singular for small stepsizes h. To solve this problem it is proposed in [25] to scale the rows of the iteration matrix, which belong to the algebraic variables by the factor $\frac{1}{h}$. In our case this factor would be $\frac{1}{h^2}$ as h^2 occurs in the iteration matrix (8.13). To investigate if this approach improve the results is left for future work.

Summary and Future Work

This work deals with the numerical solution of second order differential-algebraic equations (DAEs). We have examined different approaches to solve second order differential-algebraic systems with higher index numerically and compared this approach with the solutions of corresponding first order systems, where we have paid special attention to the equations of motion of multibody systems.
First of all, we have summarized the convergence results for the numerical solution of first order differential-algebraic systems solved with BDF methods and Runge-Kutta methods. We have seen that we get convergence problems and a reduction in the convergence order when solving higher index problems, especially when solving first order DAEs of higher index with BDF methods. The main problem when using BDF methods is that these methods suffer a loss of accuracy whenever there is a change of stepsize or change of order in the methods. The first order BDF method (implicit Euler method) even fail to converge in the first steps after a change of stepsize for higher index problems. This behavior of BDF methods is responsible for instabilities arising in the numerical solutions and for a reduction of the convergence order for higher index DAEs.
Further, we have shown some problems which arise in the order reduction of higher order DAEs. The classical transformation to first order systems by introducing new variables for the derivatives may lead to certain difficulties when applied to DAEs. When we analyze the procedure of turning a differential-algebraic system into a first order system this approach may lead to solutions that have different smoothness requirements and the resulting first order system can have a higher index than the original system. The transformation to first order can even be mathematically incorrect, i.e., the resulting first order system may not have a solution, while the original system is solvable.
Because of these difficulties it is advantageous to solve second order differential-algebraic systems directly, without previously transforming to first order systems.

We have examined modified BDF methods for the numerical solution of second order DAEs of d-index 3, especially of the second order equations of motion of multibody systems. We have implemented variable-step variable-order modified BDF methods up to order 2, following similar stepsize selection strategies as used in the code DASSL. Additionally, we have developed a Runge-Kutta method based on collocation for the numerical solution of second order systems and we have implemented a 2-stage Runge-Kutta method for the solution of the second order equations of motion.

We have introduced the concept of index reduction for higher index equations.

By index reduction the index of a DAE is decreased using certain techniques and analytical equivalence transformations in order to obtain better numerical results. We have considered the GGL-formulation and a strangeness-free formulation derived by index reduction by minimal extension, both, in the first and second order form.
Index reduction usually increases the dimension of the system as index reduction is done by adding extra equations (the derivatives of the constraints) to the equations of motion. Moreover, index reduction is an additional step in the integration procedure and we require extra assumptions and additional information about the structure of the equations.

Altogether we can summarize the advantages of solving second order systems directly as follows. First of all, we do not need to transform to first order systems and thus we may avoid to increase the index of the DAEs unintentionally or impose stronger smoothness requirements on the systems as described above. In addition we can avoid to increase (almost double) the dimension of the systems. Particularly when simulating complex mechanical systems in aircrafts or vehicle systems we have already large systems in the second order formulation. Further, we can avoid extra index reductions and instabilities in the numerical solutions, which arise when solving first order higher index DAEs. We can save additional steps in the integration procedure and have no increase in the dimension of the system or in the index, which is costly in computational time and effort as well as memory space.

We have seen that the modified BDF methods work well for the original equations of motion of multibody systems. For these systems of order 2 and d-index 3 we can obtain better results than the ordinary BDF methods (DASSL), because these have problems due to the higher index. For the strangeness-free formulations based on minimal extension, both, the modified BDF methods and DASSL yield similar results. Further, we have observed that our modified BDF methods fail for the GGL-formulation, while using DASSL we can obtain better results, though DASSL requires very small stepsize to fulfill the desired tolerances. The behavior of the modified BDF methods for the GGL-formulation might be explained due to small perturbations in the data, which lead to instabilities in the numerical solution. Further investigations of the behavior of the second order GGL-formulation under small perturbations is left for future work. It also remains to derive higher order modified BDF methods for the solution of second order systems. We may improve the performance of our codes by using better error estimates and stepsize selection strategies and for the Runge-Kutta method a better error estimation and stepsize selection may be achieved using embedded Runge-Kutta methods. The 2-stage Runge-Kutta method yields satisfactory results. We have observed an $O(h^4)$ behavior for the position coordinates q and a $O(h^3)$ behavior for the velocities $\dot{q}$. Only for the algebraic variables we observe a reduction in the order of accuracy.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der numerischen Lösung von differentiell-algebraischen Gleichungen von zweiter Ordnung und höherem Index. Wir betrachten verschiedene Ansätze zur Lösung von Systemen zweiter Ordnung und vergleichen diese mit der numerischen Lösung der zugehörigen Systeme erster Ordnung.
Differentiell-algebraische Gleichungen zweiter Ordnung treten häufig in technischen Anwendungen und in der Modellierung und Simulation von dynamischen Systemen unter Zwangsbedingungen auf. Ein spezielles Beispiel ist die Simulation von mechanischen Mehrkörpersystemen. Die Bewegungsgleichungen von Mehrkörpersystemen, die in der numerischen Simulation verwendet werden sind Gleichungen von zweiter Ordnung und Differentiationsindex 3 und gehören damit zu den Problemen von höherem Index. In dieser Arbeit haben wir uns speziell mit dieser Art von Gleichungen beschäftigt und sie für die numerische Integration herangezogen.

Der übliche Weg zur Lösung von gewöhnlichen Differentialgleichungen und differentiell-algebraischen Gleichungen höherer Ordnung ist die Transformation in ein System von Differentialgleichungen erster Ordnung durch Einführen neuer Variablen für höhere Ableitungen, d.h. für Ableitungen größer oder gleich zweiter Ordnung, um dann dieses System mit bewährten numerischen Methoden zu lösen. Deswegen haben wir zunächst die wichtigsten Konvergenz-Sätze und Resultate zur numerischen Lösung von differentiell-algebraischen Gleichungen erster Ordnung mit Hilfe von BDF Verfahren und Runge-Kutta Verfahren zusammengefaßt. Wir haben gesehen, daß Konvergenzprobleme auftreten, wenn wir Systeme höheren Index lösen wollen. Die in der numerischen Simulation häufig verwendeten BDF Verfahren reagieren mit verminderter Genauigkeit auf Änderungen in der Schrittweite oder in der Ordnung des BDF Verfahrens. Besonders das implizite Euler Verfahren (BDF Verfahren erster Ordnung) kann in den ersten Schritten nach einer Schrittweiten- oder Ordnungsänderung nicht konvergieren. Für Systeme von höherem Index müssen wir daher mit eine Reduktion der Konvergenzordnung und Instabilitäten in der numerischen Lösung rechnen.

Weiter zeigen wir Probleme auf, die bei der Ordnungsreduktion von Systemen höherer Ordnung auftreten können. Bei der Transformation zu Systemen erster Ordnung tritt bei DAEs ein Problem auf, daß bei gewöhnlichen Differentialgleichungen nicht zu beachten war. Durch die Transformation in ein System erster Ordnung kann sich der Index der DAE erhöhen und Lösungen mit veränderten und stärkeren Glattheitsbedingungen auftreten. Das resultierende System erster Ordnung kann im schlechtesten Fall sogar keine Lösung besitzen, während das System höherer Ordnung lösbar ist. Des Weiteren können Instabilitäten in der numerischen Lösung auftreten. Um diese Schwierigkeiten zu umgehen, ist es wünschenswert Systeme höherer Ordnung direkt zu lösen, ohne den Umweg über Systeme erste Ordnung gehen zu müssen.

Der Schwerpunkt dieser Arbeit liegt in der Untersuchung von modifizierten BDF Verfahren für die numerische Lösung von DAEs zweiter Ordnung und d-Index 3, insbesondere von den Bewegungsgleichungen von Mehrkörpersystemen. Dazu haben wir diese modifizierten BDF Verfahren bis zur Ordnung 2 zusammen mit einer Schrittweiten- und Ordnungssteuerung implementiert und für die numerische Lösung der Bewegungsgleichungen verwendet.
Zusätzlich entwickeln wir ein Runge-Kutta Verfahren basierend auf Kollokationsmethoden zur Lösung von DAEs zweiter Ordnung und wenden dieses ebenfalls auf die Bewegungsgleichungen an. Bei der Implementierung des Runge-Kutta Verfahrens beschränken wir uns auf ein Verfahren zweiter Stufe.
Wir geben eine kurze Einführung über Verfahren zur Reduktion des Index für die Bewegungsgleichungen von Mehrkörpersystemen von erster und zweiter Ordnung. Diese Verfahren werden verwendet, um durch verschiedene Techniken und analytische Äquivalenztransformationen Systeme niedrigeren Index zu erhalten und damit bessere numerische Resultate zu erzielen. Hierbei betrachten wir speziell die GGL-Formulierung vom Differentiationsindex 2 und die strangeness-freie Formulierung basierend auf minimaler Erweiterung vom Differentiationsindex 1, jeweils für Gleichungen von erster und zweiter Ordnung. Durch diese Index-Reduktion wird im allgemeinen die Dimension des Systems erhöht, da Index-Reduktion meist durch Hinzunahme von zusätzlichen Gleichungen realisiert wird und man benötigt oft zusätzliche Bedingungen an die DAE.
Die Vorteile einer direkten Lösung von Systemen zweiter Ordnung können wie folgt zusammengefaßt werden.
Zunächst können wir uns die Transformation in ein System erster Ordnung sparen und mit sehr viel kleineren Systemen arbeiten. Gerade bei der Simulation von komplexen mechanischen Systemen erhält man sehr schnell sehr große Systeme. Weiter vermeiden wir es den Index des Systems durch die Transformation nach erster Ordnung unbeabsichtigt zu erhöhen und können uns außerdem zum Teil den Schritt der Index-Reduktion sparen, da die vorgestellten Methoden auch für Systeme höherer Ordnung gute Ergebnisse liefern. Insgesamt können wir damit zusätzliche Schritte im Lösungsprozess vermeiden und behalten die Größe des Systems bei, wodurch Rechenzeit und Speicherbedarf reduziert wird.
Die modifizierten BDF Verfahren liefern gute Ergebnisse für die Bewegungsgleichungen in zweiter Ordnung. Für diese Systeme erhalten wir sogar bessere Resultate als bei der Lösung des zugehörigen Systems erster Ordnung mit gewöhnlichen BDF Verfahren (DASSL), da diese aufgrund des hohen Index nicht für alle Variablen gute Ergebnisse liefern können (insbesondere nicht für die algebraischen Variablen). Für die indexreduzierten Systeme erhalten wir ganz unterschiedliche Resultate. Die strangeness-freien Formulierungen basierend auf minimaler Erweiterung liefern für beide Verfahren, die modifizierten und die gewöhnlichen BDF Verfahren, gute Ergebnisse. Dagegen versagen die modifizierten BDF Verfahren für die GGL-Formulierung, während DASSL, trotz sehr kleiner Schrittweiten, um die geforderten Toleranzen zu erfüllen, gute Ergebnisse liefert.

Das Verhalten für die GGL-Formulierung kann auf kleine Störungen in den Daten zurückzuführen sein, die zu Instabilitäten in der numerischen Lösung führen. Die Untersuchung der Auswirkung von Störungen in den Daten auf das Verhalten der modifizierten BDF Verfahren ist zu untersuchen.
Weiter bleibt die Herleitung von modifizierten BDF Verfahren höherer Ordnung offen.
Das zweistufige Runge-Kutta Verfahren liefert ebenfalls zufriedenstellende Ergebnisse. Wir haben gesehen, daß die Genauigkeiten für die Ortskoordinaten q bei $O(h^4)$ und für die Geschwindigkeiten $\dot{q}$ bei $O(h^3)$ liegen. Nur für die algebraischen Variablen erhalten wir eine Reduktion in der Genauigkeit.

Bibliography

[1] C. Arévalo, P. Lötstedt. *Improving the accuracy of BDF methods for index 3 differential-algebraic equations*, BIT, Vol. 35, pp. 297-308, 1994.

[2] V. I. Arnold. *Mathematical Methods of Classical Mechanics*, Springer, 1978.

[3] M. Arnold, V. Mehrmann, A. Steinbrecher. *Index reduction in industrial multibody system simulation*, Technical Report IB 532-01-01, DLR German Aerospace Center, Institute of Aeroelasticity, Vehicle System Dynamics Group, 2001.

[4] K. E. Brenan, S. L. Campbell, L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Classics in Applied Mathematics, 1996.

[5] K. E. Brenan, B. E. Engquist. *Backward differentiation approximations of nonlinear differential/algebraic systems*, Mathematics of Computation, Vol. 51, pp. 659-676, 1988.

[6] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta and General Linear Methods*, John Wiley & Sons Ltd., 1987.

[7] P. Chartier. *General linear methods for differential-algebraic equations of index one and two*, Institut de recherche en Informatique et systèmes aléatoires, Campus de Beaulieu, Publication interne no. 711, 1993.

[8] E. Eich-Soellner, C. Führer. *Numerical Methods in Multibody Dynamics*, B.G. Teubner, Stuttgart, 1998.

[9] C. Führer, B. J. Leimkuhler. *Numerical solution of differential-algebraic equations for constrained mechanical motion*, Numerische Mathematik, Vol. 59, pp. 55-69, 1991.

[10] C. Führer. *Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen - Theorie, numerische Ansätze und Anwendungen*, TU München, 1988.

[11] C. W. Gear, B. Leimkuhler, G. K. Gupta. *Automatic integration of Euler-Lagrange equations with constraints*, Journal of Computional and Applied Mathematics, Vol. 12/13, pp. 77-90, 1985.

[12] C. W. Gear, L. R. Petzold. *Singular implicit ordinary differential equations and constraints*, Report No. UIUCDCS-R-82-1110, Department of Computer Science, University of Illinois at Urbana-Champaign, 1982.

[13] E. Hairer, C. Lubich, M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer, Berlin, 1989.

[14] E. Hairer, S.P. Norsett, G. Wanner. *Solving Ordinary Differential Equations I - Nonstiff Problems*, Springer, 2. Edition, Berlin Heidelberg, 1993.

[15] E. Hairer, G. Wanner. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*, Springer, 1991.

[16] G. Hamel. *Theoretische Mechanik*, Springer, 1967.

[17] P. Kunkel, V. Mehrmann. *Analysis und Numerik linearer differentiell-algebraischer Gleichungen*, "Wissenschaftliches Rechnen - Eine Einführung in das Scientific Computing", Herzberger (Hrsg.), Akademie Verlag, Berlin, pp. 233-278, 1995.

[18] P. Kunkel, V. Mehrmann. *Canonical forms for linear differential-algebraic equations with variable coefficients*, Journal of Computational and Applied Mathematics, Vol. 56, pp. 225-251, 1994.

[19] P. Kunkel, V. Mehrmann. *Local and global invariants of linear differential-algebraic equations and their relation*, Electronic Transactions on Numerical Analysis, Vol. 4, pp. 138-157, 1996.

[20] P. Kunkel, V. Mehrmann. *Regular solutions of nonlinear differential-algebraic equations and their numerical determination*, Numerische Mathematik, Vol. 79, pp. 581-600, 1998.

[21] P. Kunkel, V. Mehrmann. *Analysis of over- and underdetermined nonlinear differential-algebraic systems with application to nonlinear control problems*, Mathematics of Control, Signals and Systems, Vol. 14, pp. 233-256, 2001.

[22] P. Kunkel, V. Mehrmann. *Index reduction for differential-algebraic equations by minimal extension*, Zeitschrift für Angewandte Mathematik und Mechanik, Vol. 84, pp. 579-597, 2004.

[23] P. Kunkel, V. Mehrmann, W. Rath. *Analysis and numerical solution of control problems in descriptor form*, Mathematics of Control, Signals and Systems, Vol. 14, pp. 29-61, 2001.

[24] V. Mehrmann, C. Shi. *Analysis of higher order linear differential-algebraic systems*, Institute of Mathematics, Technische Universität Berlin, Preprint 17-2004, 2004.

[25] R.L. Petzold, P. Lötstedt. *Numerical solution of nonlinear differential equations with algebraic constraints II: practical implications*, SIAM Journal on Scientific and Statistic Computing, Vol. 7, No. 3, pp. 720-733, 1986.

[26] R. Roberson, R. Schwertassek. *Dynamics of Multibody Systems*, Springer, 1988.

[27] J. Sand. *On implicit Euler for high-order high-index DAEs*, Applied Numerical Mathematics, Vol. 42, pp. 411-424, 2002.

[28] S. Schneider. *Convergence of general linear methods on differential-algebraic systems of index 3*, BIT, Vol. 37, No. 2, pp. 424-441, 1997.

[29] S. Schulz. *General linear methods for linear DAEs*, Humboldt Universität zu Berlin, Institut für Mathematik, Preprint 03-10, 2003.

[30] C. Shi. *Linear differential-algebraic equations of higher-order and the regularity or singularity of matrix polynomials*, Ph.D. Thesis, Institute of Mathematics, Technische Universität Berlin, 2004.

[31] B. Simeon, F. Grupp, C. Führer, P. Rentrop. *A nonlinear truck model and its treatment as a multibody system*, Journal of Computional and Applied Mathematics, Vol. 50, pp. 523-532, 1994.

[32] A. Steinbrecher. *Regularization of nonlinear equations of motion of multibody systems by index reduction with preseving the solution manifold.* Technical Report 742-2002, Institute of Mathematics, Technische Universität Berlin, 2002.

[33] K. Strehmel, R. Weiner. *Numerik gewöhnlicher Differentialgleichungen.* Teubner Studienbücher Mathematik, Stuttgart 1995.

Appendix A

Appendix

In this part we consider general linear methods for the numerical solution of first order differential-algebraic systems. The concept of general linear methods was first introduced by Butcher [6]. General linear methods include Runge-Kutta methods, linear multistep methods, predictor-corrector methods and one-leg methods, e.g., BDF methods. We will first consider general linear methods applied to initial value problems in ODEs as described in [6], extend this approach to DAEs and then give convergence results for general linear methods applied to initial value problems in semi-explicit differential-algebraic systems of d-index ν with $\nu \in \{1, 2, 3\}$. The convergence results for systems of d-index 1 and 2 are from [7] and the result for systems of d-index 3 are from [28].

General Linear Methods for ODEs

Consider the initial value problem

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = \hat{y}_0, \tag{A.1}$$

with a function $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$.
A k-step s-stage general linear method given by the coefficient matrices $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{k\times k}$, $\mathcal{B} = [b_{ij}] \in \mathbb{R}^{k\times s}$, $\tilde{\mathcal{A}} = [\tilde{a}_{ij}] \in \mathbb{R}^{s\times k}$, $\tilde{\mathcal{B}} = [\tilde{b}_{ij}] \in \mathbb{R}^{s\times s}$ and vector $c = [c_1, \dots, c_s]^T$ applied to the initial value problem (A.1) can be formulated as

$$y_{n+1,i} = \sum_{j=1}^{k} a_{ij} y_{n,j} + h \sum_{j=1}^{s} b_{ij} f(Y_{n,j}, t_n + c_j h), \quad i = 1, \dots, k, \tag{A.2a}$$

$$Y_{n,i} = \sum_{j=1}^{k} \tilde{a}_{ij} y_{n,j} + h \sum_{j=1}^{s} \tilde{b}_{ij} f(Y_{n,j}, t_n + c_j h), \quad i = 1, \dots, s, \tag{A.2b}$$

with $t_n = nh$, where $h > 0$ denotes the stepsize.
One defines the *correct value function* $\mathcal{U}(t, h)$ which takes values in $\mathbb{R}^{mk}$. Each of the k components of $\mathcal{U}(t, h)$ represents a m dimensional function, which relates in some way to the exact solution $y(t)$, i.e., $\mathcal{U}(t, h)$ gives the interpretation of the method. For example, $\mathcal{U}(t_n, h)$ can stand for function values $y(t_n - (i-1)h)$ of the exact solution. See Examples A.0.1 and A.0.2 for two possible correct

value functions. In each step of the integration an approximation y_n to $\mathcal{U}(t_n, h)$ is computed.
In (A.2) $y_{n,i}$ for $i = 1, \dots, k$ denotes the external stages. The vector of external stages $y^{[n]} = [y_{n,1}, \dots, y_{n,k}]^T$ approximates a given correct value function $\mathcal{U}(t_n, h)$ at step n, and contains the necessary information for doing the next step. The internal stages $Y_{n,i}$ are the solutions of the nonlinear system (A.2b) approximating $y(t_n + c_i h)$, $i = 1, \dots, s$, where $y(t)$ is the exact solution of (A.1). It is assumed that an approximation of $y(t_n)$ can be recovered from $\{y_{n,i}, i = 1, \dots, k\}$. At the start of step number $n+1$, k quantities $y_{n,1}, \dots, y_{n,k}$ are available from approximations computed in step n and corresponding quantities $y_{n+1,1}, \dots, y_{n+1,k}$ are evaluated in step number $n+1$.

General linear methods include one-step methods like Runge-Kutta methods and multistep methods. To illustrate this we give two examples.

Example A.0.1. *Runge-Kutta methods*
Consider a Runge-Kutta method as described in Section 5.1 of the form

$$\begin{aligned} \tilde{Y}_{n,i} &= \tilde{y}_n + h \sum_{j=1}^{s} a_{ij} f(\tilde{Y}_{n,j}, t_n + c_j h), \quad i = 1, 2, \dots, s, \\ \tilde{y}_{n+1} &= \tilde{y}_n + h \sum_{j=1}^{s} b_j f(\tilde{Y}_{n,j}, t_n + c_j h). \end{aligned} \tag{A.3}$$

If we identify $y^{[n]}$ with $\tilde{y}_n$ and Y_n with $[\tilde{Y}_{n,1}^T, \dots, \tilde{Y}_{n,s}^T]^T$ then we can write method (A.3) *as a general linear method with $k = 1$ and the coefficient matrices*

$$\mathcal{A} = [1], \tilde{\mathcal{A}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^s, \tilde{\mathcal{B}} = [a_{ij}] \in \mathbb{R}^{s,s}, \mathcal{B} = [b_1, \dots, b_s]^T, c = [c_1, \dots, c_s].$$

The correct value function is given by $\mathcal{U}(t, h) = y(t)$, where $y(t)$ is the exact solution.

Example A.0.2. *Linear multistep methods*
Consider the linear multistep method

$$\sum_{i=0}^{k} \alpha_i \tilde{y}_{n+i} = h \sum_{i=0}^{k} \beta_i f(\tilde{y}_{n+i}).$$

If we identify $y^{[n]}$ with $[\tilde{y}_{n+k-1}^T, \dots, \tilde{y}_n^T]^T$ and Y_n with $[\tilde{y}_{n+k}^T, \dots, \tilde{y}_n^T]^T$, then we can write the method as a general linear method with the coefficient matrices

$$\mathcal{A} = \begin{bmatrix} -\frac{\alpha_{k-1}}{\alpha_k} & \cdots & \cdots & -\frac{\alpha_0}{\alpha_k} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{k,k},$$

$$\mathcal{B} = \begin{bmatrix} \frac{\beta_k}{\alpha_k} & \cdots & \frac{\beta_0}{\alpha_k} \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{k,k+1},$$

$$\tilde{\mathcal{A}} = \begin{bmatrix} -\frac{\alpha_{k-1}}{\alpha_k} & \cdots & \cdots & -\frac{\alpha_0}{\alpha_k} \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{k+1,k},$$

$$\tilde{\mathcal{B}} = \begin{bmatrix} \frac{\beta_k}{\alpha_k} & \cdots & \frac{\beta_0}{\alpha_k} \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{k+1,k+1},$$

$c = [k, k-1, \ldots, 0]^T \in \mathbb{R}^{k+1}$.
Here, $s = k+1$ and the correct value function is given by

$$\mathcal{U}(t,h) = \begin{bmatrix} y(t) \\ y(t-h) \\ \vdots \\ y(t-(k-1)h) \end{bmatrix}.$$

If we regard $[y_{n,1}, y_{n,2}, \ldots, y_{n,k}]^T$ as a point in X^k (X being the vector space on which the original differential equation is defined), then we can write (A.2) in a compact form as

$$\begin{aligned} y^{[n+1]} &= (\mathcal{A} \otimes I_m) y^{[n]} + h(\mathcal{B} \otimes I_m) F(Y_n), \\ Y_n &= (\tilde{\mathcal{A}} \otimes I_m) y^{[n]} + h(\tilde{\mathcal{B}} \otimes I_m) F(Y_n), \end{aligned} \tag{A.4}$$

where m is the dimension of the system, I_m is the m-dimensional identity matrix and

$$F(Y_n) = \begin{bmatrix} f(Y_{n,1}, t_n + c_1 h) \\ \vdots \\ f(Y_{n,s}, t_n + c_s h) \end{bmatrix}, \ y^{[n]} = \begin{bmatrix} y_{n,1} \\ \vdots \\ y_{n,k} \end{bmatrix}, \ Y_n = \begin{bmatrix} Y_{n,1} \\ \vdots \\ Y_{n,s} \end{bmatrix}.$$

In (A.4) the direct product $\mathcal{A} \otimes I$ is defined as $(\mathcal{A} \otimes I)V = W$ for $V = [v_1, v_2, \ldots, v_k] \in X^k$ and $W = [w_1, w_2, \ldots, w_k] \in X^k$ is given by
$w_i = \sum_{j=1}^{k} a_{ij} v_j$, $i = 1, \ldots, k$.
It is convenient to express a general linear method (A.4) in a partitioned form as

$$\mathcal{M} = \begin{bmatrix} \mathcal{A} & \mathcal{B} \\ \tilde{\mathcal{A}} & \tilde{\mathcal{B}} \end{bmatrix}. \tag{A.5}$$

In general we have only the initial value $y(t_0)$ given, so that we need a starting method to obtain an approximation to the initial vector $y^{[0]}$. The order of accuracy of a general linear method is defined relative to this starting method. We now consider the effect of applying the method $\mathcal{M}$ given by (A.5) to the result computed by a starting method $\mathcal{S}$. In the following $\mathcal{S}(h, y(t_0))$ and $\mathcal{M}(h, \mathcal{S}(h, y(t_0)))$ denote the solutions computed after one step with stepsize h using $\mathcal{S}$ and $\mathcal{M}$ respectively.

Definition A.0.3. *[6]*
The order of accuracy *of the general linear method $\mathcal{M}$* relative to *$\mathcal{S}$ is defined*

as the largest integer p, such that

$$\|\mathcal{M}(h, \mathcal{S}(h, y(t_0))) - \mathcal{S}(h, y(t_0 + h))\| \leq Ph^{p+1}$$

for $h < h_0$, where the constants P and h_0 may depend on the problem.
The order of accuracy *of the general linear method $\mathcal{M}$ is the largest p such that there exists a starting method $\mathcal{S}$, such that $\mathcal{M}$ is of order p relative to $\mathcal{S}$.*

Definition A.0.4. *[28]*
The general linear method (A.2) *applied to the initial value problem* (A.1) *has* stage order q $(q + 1 \leq p)$ *if*

$$y(t_n + c_i h) = \sum_{j=1}^{k} \tilde{a}_{ij} y_j(t_n) + h \sum_{j=1}^{s} \tilde{b}_{ij} f(y(t_n + c_j h), t_n + c_j h) + O(h^{q+1}),$$

for $i = 1, \dots, s$.

Definition A.0.5. *[28],[7]*

1. *The* stability matrix *of a general linear method $\mathcal{M}$ is defined by $M(z) = \mathcal{A} + \mathcal{B}(I - z\tilde{\mathcal{B}})^{-1} z\tilde{\mathcal{A}}$ and the limit of this matrix is given by $M(\infty) = \mathcal{A} - \mathcal{B}\tilde{\mathcal{B}}^{-1}\tilde{\mathcal{A}}$.*
   The stability domain S *is defined by*

   $$S := \{z \in \mathbb{C} | \sup_n \|(M(z))^n\| < \infty\}.$$

2. *A general linear method $\mathcal{M}$ is called* stable at the origin *if and only if $0 \in S$.*

3. *A general linear method $\mathcal{M}$ with $\tilde{\mathcal{B}}$ non-singular is called* stable at infinity *if and only if*

   $$\rho(M(\infty)) = \rho(\mathcal{A} - \mathcal{B}\tilde{\mathcal{B}}^{-1}\tilde{\mathcal{A}}) \leq 1$$

   and for all $\lambda \in \sigma(M(\infty))$ with $|\lambda| = 1$ it follows that λ is non-defective. Here, $\rho(M(\infty))$ denotes the spectral radius and $\sigma(M(\infty))$ the spectrum of $M(\infty)$.

4. *A general linear method $\mathcal{M}$ with $\tilde{\mathcal{B}}$ non-singular is called* strictly stable at infinity *if and only if*

   $$\rho(M(\infty)) = \rho(\mathcal{A} - \mathcal{B}\tilde{\mathcal{B}}^{-1}\tilde{\mathcal{A}}) < 1.$$

Definition A.0.6. *A general linear method $\mathcal{M}$ is said to be* zero-stable *if all eigenvalues of $\mathcal{A}$ are on or within the unit circle and eigenvalues on the unit circle are simple.*

Definition A.0.7. *[7]*
A general linear method $\mathcal{M}$ whose correct value function is composed only of values of the exact solution is called stiffly accurate *if for all $l \in \{1, \dots, k\}$ there exists $m \in \{1, \dots, s\}$, such that*

$$a_{l,j} = \tilde{a}_{m,j} \text{ for all } j \in \{1, \dots, k\},$$
$$b_{l,j} = \tilde{b}_{m,j} \text{ for all } j \in \{1, \dots, s\}.$$

General Linear Methods for DAEs of d-Index 1

We now extend the approach given in Section A to the numerical solution of DAEs. First of all, we consider a semi-explicit differential-algebraic system of d-index 1 of the form

$$\begin{aligned} \dot{y}(t) &= f(y(t), z(t)), \\ 0 &= g(y(t), z(t)), \end{aligned} \tag{A.6}$$

with consistent initial values $y(t_0) = \hat{y}_0$, $z(t_0) = \hat{z}_0$ and properties as described in Section 2.2.1. Method (A.4) applied to the d-index 1 problem (A.6) reads

$$\begin{aligned} y^{[n+1]} &= (\mathcal{A} \otimes I_{n_y}) y^{[n]} + h(\mathcal{B} \otimes I_{n_y}) \dot{Y}_n, \\ z^{[n+1]} &= (\mathcal{A} \otimes I_{n_z}) z^{[n]} + h(\mathcal{B} \otimes I_{n_z}) \dot{Z}_n, \end{aligned} \tag{A.7a}$$

with internal stages

$$\begin{aligned} Y_n &= (\tilde{\mathcal{A}} \otimes I_{n_y}) y^{[n]} + h(\tilde{\mathcal{B}} \otimes I_{n_y}) \dot{Y}_n, \\ Z_n &= (\tilde{\mathcal{A}} \otimes I_{n_z}) z^{[n]} + h(\tilde{\mathcal{B}} \otimes I_{n_z}) \dot{Z}_n, \end{aligned} \tag{A.7b}$$

and

$$\begin{aligned} \dot{Y}_n &= f(Y_n, Z_n), \\ 0 &= g(Y_n, Z_n). \end{aligned} \tag{A.7c}$$

The convergence results for general linear methods applied to the semi-explicit d-index 1 system are given by the following theorem.

Theorem A.0.8. *Suppose that for the d-index 1 problem* (A.6) $g_z(y, z)$ *is non-singular in a neighborhood of the solution and that the initial values* $(\hat{y}_0, \hat{z}_0)$ *are consistent. Consider a general linear method* $\mathcal{M}$ *of order* p *and stage order* q *(with* $p \geq q$*), that is stable at the origin and has a non-singular matrix* $\tilde{\mathcal{B}}$*. Suppose that the error of the starting procedure* $\mathcal{S}(h, (y(t_0), z(t_0)))$ *satisfies*

$$\begin{aligned} \mathcal{S}_y(h, \hat{y}_0) - \mathcal{U}_y(t_0, h) &= O(h^p), \\ \mathcal{S}_z(h, \hat{z}_0) - \mathcal{U}_z(t_0, h) &= O(h^{q+1}). \end{aligned}$$

Then the global error of the integration procedure (A.7) *satisfies*

$$\begin{aligned} y^{[n]} - \mathcal{U}_y(t_n, h) &= O(h^p), \\ z^{[n]} - \mathcal{U}_z(t_n, h) &= O(h^r), \end{aligned}$$

whenever $nh \leq C$*, where*

1. $z^{[n]} - \mathcal{U}_z(t_n, h) = O(h^p)$*, if the method is stiffly accurate,*
2. $r = min(p, q + 1)$*, if the method is stable at infinity and* $1 \notin \sigma(M(\infty))$*,*
3. $r = min(p - 1, q)$*, if the method is stable at infinity and* $1 \in \sigma(M(\infty))$*,*
4. *the solution* $z^{[n]}$ *diverges, if the method is not stable at infinity.*

Proof. See [7], Theorem 1. □

Remark A.0.9. $\mathcal{U}_y(t, h)$ *denotes the correct value function of the* y*-component and* $\mathcal{U}_z(t, h)$ *denotes the correct value function of the* z*-component of the solution. Furthermore,* $\mathcal{S}_y(h, \hat{y}_0)$ *is an approximation to* $\mathcal{U}_y(t_0, h)$ *and* $\mathcal{S}_z(h, \hat{z}_0)$ *is an approximation to* $\mathcal{U}_z(t_0, h)$ *given by the starting procedure* $\mathcal{S}(h, (y(t_0), z(t_0)))$*.*

General Linear Methods for DAEs of d-Index 2

We now consider a semi-explicit system of d-index 2

$$\begin{aligned} \dot{y}(t) &= f(y(t), z(t)), \\ 0 &= g(y(t)), \end{aligned} \tag{A.8}$$

with consistent initial values $y(t_0) = \hat{y}_0$ and $z(t_0) = \hat{z}_0$ and properties as described in Section 2.2.2. The direct approach of method (A.4) to the d-index 2 problem (A.8) is given by

$$\begin{aligned} y^{[n+1]} &= (\mathcal{A} \otimes I_{n_y})y^{[n]} + h(\mathcal{B} \otimes I_{n_y})\dot{Y}_n, \\ z^{[n+1]} &= (\mathcal{A} \otimes I_{n_z})z^{[n]} + h(\mathcal{B} \otimes I_{n_z})\dot{Z}_n, \end{aligned} \tag{A.9a}$$

with internal stages

$$\begin{aligned} Y_n &= (\tilde{\mathcal{A}} \otimes I_{n_y})y^{[n]} + h(\tilde{\mathcal{B}} \otimes I_{n_y})\dot{Y}_n, \\ Z_n &= (\tilde{\mathcal{A}} \otimes I_{n_z})z^{[n]} + h(\tilde{\mathcal{B}} \otimes I_{n_z})\dot{Z}_n, \end{aligned} \tag{A.9b}$$

and

$$\begin{aligned} \dot{Y}_n &= f(Y_n, Z_n), \\ 0 &= g(Y_n). \end{aligned} \tag{A.9c}$$

We get the following convergence results for the d-index 2 case.

Theorem A.0.10. *Suppose that $g_y f_z(y, z)$ is nonsingular in a neighborhood of the solution $[y(t), z(t)]$ of* (A.8) *and that the initial values $[\hat{y}_0, \hat{z}_0]$ are consistent.*

1. *Consider a zero-stable stiffly accurate general linear method of order p and stage order q, with $p \geq 2$ and $q \geq 1$. Then the method* (A.9) *is convergent of order $r = min(p, q + 1)$, i.e.,*

$$y^{[n]} - \mathcal{U}_y(t_n, h) = O(h^r) \tag{A.10}$$

   for $t_n - t_0 = nh \leq C$.

2. *Consider a zero-stable general linear method that is strictly stable at infinity such that the global error of the y-component is $O(h^r)$, $g(y_{n,i}) = O(h^{r+1})$ for all $i = 1, \dots, k$ and the local error of the z-component is $O(h^r)$ with $r \geq 2$. Then we have for the global error of the z-component*

$$z^{[n]} - \mathcal{U}_z(t_n, h) = O(h^r) \tag{A.11}$$

   for $t_n - t_0 = nh \leq C$.

Proof. See [7], Theorem 5 and Theorem 6. □

General Linear Methods for DAEs of d-Index 3

Finally, we consider a semi-explicit differential-algebraic system of d-index 3 of the form

$$\begin{aligned} \dot{y}(t) &= f(y(t), z(t)), \\ \dot{z}(t) &= k(y(t), z(t), u(t)), \\ 0 &= g(y(t)), \end{aligned} \tag{A.12}$$

with consistent initial values $y(t_0) = \hat{y}_0$, $z(t_0) = \hat{z}_0$, $u(t_0) = \hat{u}_0$ and properties as described in Section 2.2.3. Method (A.4) applied to the d-index 3 problem (A.12) reads

$$\begin{aligned} y^{[n+1]} &= (\mathcal{A} \otimes I_{n_y}) y^{[n]} + h(\mathcal{B} \otimes I_{n_y}) \dot{Y}_n, \\ z^{[n+1]} &= (\mathcal{A} \otimes I_{n_z}) z^{[n]} + h(\mathcal{B} \otimes I_{n_z}) \dot{Z}_n, \\ u^{[n+1]} &= (\mathcal{A} \otimes I_{n_u}) u^{[n]} + h(\mathcal{B} \otimes I_{n_u}) \dot{U}_n, \end{aligned} \tag{A.13a}$$

with internal stages

$$\begin{aligned} Y_n &= (\tilde{\mathcal{A}} \otimes I_{n_y}) y^{[n]} + h(\tilde{\mathcal{B}} \otimes I_{n_y}) \dot{Y}_n, \\ Z_n &= (\tilde{\mathcal{A}} \otimes I_{n_z}) z^{[n]} + h(\tilde{\mathcal{B}} \otimes I_{n_z}) \dot{Z}_n, \\ U_n &= (\tilde{\mathcal{A}} \otimes I_{n_u}) u^{[n]} + h(\tilde{\mathcal{B}} \otimes I_{n_u}) \dot{U}_n, \end{aligned} \tag{A.13b}$$

and

$$\begin{aligned} \dot{Y}_n &= f(Y_n, Z_n), \\ \dot{Z}_n &= k(Y_n, Z_n, U_n), \\ 0 &= g(Y_n). \end{aligned} \tag{A.13c}$$

Theorem A.0.11. *Let a general linear method* (A.13) *have order p and stage order $q \geq 3$, $p \geq q$. We suppose that $\tilde{\mathcal{B}}$ is non-singular and that the initial values $[\hat{y}_0, \hat{z}_0, \hat{u}_0]$ are consistent. We assume that the method is stable at the origin $(0 \in S)$ and strictly stable at infinity. Then the global error of method* (A.13) *applied to problem* (A.12) *satisfies*

$$\begin{aligned} y^{[n]} - \mathcal{U}_y(t_n, h) &= O(h^q), \\ z^{[n]} - \mathcal{U}_z(t_n, h) &= O(h^q), \\ u^{[n]} - \mathcal{U}_u(t_n, h) &= O(h^{q-1}) \end{aligned} \tag{A.14}$$

for $0 < h \leq h_0$ and $t_n - t_0 = nh \leq C$.

Proof. See [28] Theorem 2.4. □

General Linear Methods for Second Order DAEs

The same approach as for Runge-Kutta methods for the numerical solution of second order differential-algebraic systems can be done for general linear methods.
Consider the second order ODE

$$\ddot{y}(t) = f(y(t), \dot{y}(t), t). \tag{A.15}$$

A k-step s-stage general linear method for the numerical solution of (A.15) is given by

$$
\begin{aligned}
y_{n+1,i} &= \sum_{j=1}^{k} a_{ij} y_{n,j} + h \sum_{j=1}^{k} b_{ij} \dot{y}_{n,j} + h^2 \sum_{j=1}^{s} d_{ij} f(Y_{n,j}, \dot{Y}_{n,j}, t_n + c_j h), \\
\dot{y}_{n+1,i} &= \sum_{j=1}^{k} e_{ij} \dot{y}_{n,j} + h \sum_{j=1}^{s} f_{ij} f(Y_{n,j}, \dot{Y}_{n,j}, t_n + c_j h), \quad i = 1, \ldots, k, \\
Y_{n,i} &= \sum_{j=1}^{k} \tilde{a}_{ij} y_{n,j} + h \sum_{j=1}^{k} \tilde{b}_{ij} \dot{y}_{n,j} + h^2 \sum_{j=1}^{s} \tilde{d}_{ij} f(Y_{n,j}, \dot{Y}_{n,j}, t_n + c_j h), \\
\dot{Y}_{n,i} &= \sum_{j=1}^{k} \tilde{e}_{ij} \dot{y}_{n,j} + h \sum_{j=1}^{s} \tilde{f}_{ij} f(Y_{n,j}, \dot{Y}_{n,j}, t_n + c_j h) \quad i = 1, \ldots, s.
\end{aligned}
\tag{A.16}
$$

With the coefficient matrices
$\mathcal{A} = [a_{ij}] \in \mathbb{R}^{k \times k}$, $\mathcal{B} = [b_{ij}] \in \mathbb{R}^{k \times k}$, $\mathcal{D} = [d_{ij}] \in \mathbb{R}^{k \times s}$, $\mathcal{E} = [e_{ij}] \in \mathbb{R}^{k \times k}$, $\mathcal{F} = [f_{ij}] \in \mathbb{R}^{k \times s}$, $\tilde{\mathcal{A}} = [\tilde{a}_{ij}] \in \mathbb{R}^{s \times k}$, $\tilde{\mathcal{B}} = [\tilde{b}_{ij}] \in \mathbb{R}^{s \times k}$, $\tilde{\mathcal{D}} = [\tilde{d}_{ij}] \in \mathbb{R}^{s \times s}$, $\tilde{\mathcal{E}} = [\tilde{e}_{ij}] \in \mathbb{R}^{s \times k}$, $\tilde{\mathcal{F}} = [\tilde{f}_{ij}] \in \mathbb{R}^{s \times s}$, and vector $c = [c_1, \ldots, c_s]^T$ this can be written in the compact form

$$
\begin{aligned}
y^{[n+1]} &= (\mathcal{A} \otimes I_m) y^{[n]} + h(\mathcal{B} \otimes I_m) \dot{y}^{[n]} + h^2 (\mathcal{D} \otimes I_m) F(Y_n, \dot{Y}_n), \\
\dot{y}^{[n+1]} &= (\mathcal{E} \otimes I_m) \dot{y}^{[n]} + h(\mathcal{F} \otimes I_m) F(Y_n, \dot{Y}_n), \\
Y_n &= (\tilde{\mathcal{A}} \otimes I_m) y^{[n]} + h(\tilde{\mathcal{B}} \otimes I_m) \dot{y}^{[n]} + h^2 (\tilde{\mathcal{D}} \otimes I_m) F(Y_n, \dot{Y}_n), \\
\dot{Y}_n &= (\tilde{\mathcal{E}} \otimes I_m) \dot{y}^{[n]} + h(\tilde{\mathcal{F}} \otimes I_m) F(Y_n, \dot{Y}_n),
\end{aligned}
\tag{A.17}
$$

where m is the dimension of the system and

$$
F(Y_n, \dot{Y}_n) = \begin{bmatrix} f(Y_{n,1}, \dot{Y}_{n,1}, t_n + c_1 h) \\ \vdots \\ f(Y_{n,s}, \dot{Y}_{n,s}, t_n + c_s h) \end{bmatrix}, \; y^{[n]} = \begin{bmatrix} y_{n,1} \\ \vdots \\ y_{n,k} \end{bmatrix}, \; \dot{y}^{[n]} = \begin{bmatrix} \dot{y}_{n,1} \\ \vdots \\ \dot{y}_{n,k} \end{bmatrix},
$$

$$
Y_n = \begin{bmatrix} Y_{n,1} \\ \vdots \\ Y_{n,s} \end{bmatrix} \text{ and } \dot{Y}_n = \begin{bmatrix} \dot{Y}_{n,1} \\ \vdots \\ \dot{Y}_{n,s} \end{bmatrix}.
$$

Example A.0.12. *When we consider the Runge-Kutta method* (6.15) *in Chapter 6 as such a general linear method, then the coefficient matrices in* (A.17) *are given by*

$$
\begin{aligned}
&\mathcal{A} = [1], \mathcal{B} = [1], \mathcal{E} = [1], \mathcal{D} = [\tilde{b}_1, \ldots, \tilde{b}_s]^T, \mathcal{F} = [b_1, \ldots, b_s]^T, \\
&\tilde{\mathcal{A}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^s, \tilde{\mathcal{B}} = \begin{bmatrix} c_1 \\ \vdots \\ c_s \end{bmatrix} \in \mathbb{R}^s, \tilde{\mathcal{E}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^s, \\
&\tilde{\mathcal{D}} = [\tilde{a}_{ij}] \in \mathbb{R}^{s,s}, \tilde{\mathcal{F}} = [a_{ij}] \in \mathbb{R}^{s,s}, c = [c_1, \ldots, c_s]^T.
\end{aligned}
\tag{A.18}
$$

In (A.18) *the coefficients a_{ij}, b_i, $\tilde{a}_{ij}$, $\tilde{b}_i$ and c_i are the coefficients of the Runge-Kutta method* (6.15).

Now, we consider a second order DAE of the form

$$\begin{aligned} \ddot{y}(t) &= f(y(t), \dot{y}(t), \lambda(t)), \\ 0 &= g(y(t)). \end{aligned} \tag{A.19}$$

When we use method (A.17) for the numerical solution of this differential-algebraic system we get the following method

$$\begin{aligned} y_{n+1,i} &= \sum_{j=1}^{k} a_{ij} y_{n,j} + h \sum_{j=1}^{k} b_{ij} \dot{y}_{n,j} + h^2 \sum_{j=1}^{s} d_{ij} \ddot{Y}_{n,j}, \\ \dot{y}_{n+1,i} &= \sum_{j=1}^{k} e_{ij} \dot{y}_{n,j} + h \sum_{j=1}^{s} f_{ij} \ddot{Y}_{n,j}, \\ \lambda_{n+1,i} &= \sum_{j=1}^{k} a_{ij} \lambda_{n,j} + h \sum_{j=1}^{k} b_{ij} \dot{\lambda}_{n,j} + h^2 \sum_{j=1}^{s} d_{ij} \ddot{\Lambda}_{n,j}, \\ \dot{\lambda}_{n+1,i} &= \sum_{j=1}^{k} e_{ij} \dot{\lambda}_{n,j} + h \sum_{j=1}^{s} f_{ij} \ddot{\Lambda}_{n,j}, \end{aligned} \tag{A.20}$$

for $i = 1, \ldots, k$, where

$$\begin{aligned} \ddot{Y}_{n,j} &= f(Y_{n,j}, \dot{Y}_{n,j}, \Lambda_{n,j}), \\ 0 &= g(Y_{n,j}) \end{aligned} \tag{A.21}$$

for $j = 1, \ldots, s$ and the internal stages are given by

$$\begin{aligned} Y_{n,i} &= \sum_{j=1}^{k} \tilde{a}_{ij} y_{n,j} + h \sum_{j=1}^{k} \tilde{b}_{ij} \dot{y}_{n,j} + h^2 \sum_{j=1}^{s} \tilde{d}_{ij} \ddot{Y}_{n,j}, \\ \dot{Y}_{n,i} &= \sum_{j=1}^{k} \tilde{e}_{ij} \dot{y}_{n,j} + h \sum_{j=1}^{s} \tilde{f}_{ij} \ddot{Y}_{n,j}, \\ \Lambda_{n,i} &= \sum_{j=1}^{k} \tilde{a}_{ij} \lambda_{n,j} + h \sum_{j=1}^{k} \tilde{b}_{ij} \dot{\lambda}_{n,j} + h^2 \sum_{j=1}^{s} \tilde{d}_{ij} \ddot{\Lambda}_{n,j}. \end{aligned} \tag{A.22}$$

Numerical results for this class of methods remain for future work.