Numerical Solution of Semi-explicit Systems of Second Order Differential-Algebraic Equations

Lena Wunderlich
Institut für Mathematik, Technische Universität Berlin,
Straße des 17. Juni 136, D-10623 Berlin
wunder@math.tu-berlin.de

15th September 2005

Abstract

Second order systems of differential-algebraic equations arise naturally in many industrial applications. The classical approach of transforming a second order system into a first order system and then discretizing the first order system may lead to certain numerical difficulties. Therefore, a direct discretization and numerical solution of the second order system is preferred. We study BDF methods, Runge-Kutta methods and general linear methods for the numerical solution of second order differential-algebraic systems in semi-explicit form and demonstrate their behavior with numerical examples.

1 Introduction

Second order systems of differential-algebraic equations (DAEs) of the form $F(t, y, \dot{y}, \ddot{y}) = 0$ arise naturally in many technical applications. One important application is the modeling and simulation of mechanical multibody systems, where we usually have second order differential equations to describe the dynamics of the system coupled with some algebraic equations describing constraints [7]. Other applications where second order DAEs frequently arise are models of electrical circuits [10, 11]. In this paper we will restrict ourselves to second order semi-explicit differential-algebraic systems of the form

$$\begin{aligned} \ddot{y}(t) &= f(t, y(t), \dot{y}(t), \lambda(t)), \\ 0 &= g(t, y(t)), \end{aligned} \tag{1}$$

with sufficiently smooth functions $f : [t_0, t_0 + T] \times \mathbb{R}^{m_y} \times \mathbb{R}^{m_y} \times \mathbb{R}^{m_\lambda} \to \mathbb{R}^{m_y}$ and $g : [t_0, t_0 + T] \times \mathbb{R}^{m_y} \to \mathbb{R}^{m_\lambda}$ and initial values $y(t_0) = \eta_0$, $\dot{y}(t_0) = \eta_1$, $\lambda(t_0) = \lambda_0$.
The classical approach for the numerical solution of second order differential-algebraic systems is the transformation into a first order system by introducing new variables for the first order derivatives and then to discretize and numerically solve the first order system. The aim of this work is to show that the direct discretization of the second order system yields better numerical results and is able to prevent certain numerical difficulties.

For DAEs the accuracy and stability of the numerical solution strongly depends on a characteristic quantity called the index, in a way that the higher the index of the DAE the more sensitive is the numerical solution to perturbations and errors in the data. A higher index leads to difficulties in the numerical solution as the numerical method may not converge for higher index problems and instabilities may occur. There are several index concepts, the *differentiation index (d-index)* [2, 14], the *strangeness index (s-index)* [18], the *perturbation index (p-index)* [14] or the *tractability index (t-index)* [9, 19], but in this paper we will only

consider the perturbation index, because it is best suited for higher order equations. The notion of the perturbation index is based on the investigation of the sensitivity of a solution of the DAE with respect to initial values and right-hand sides. For the exact definition see [4, 14]. An extension of the definition for second order nonlinear DAEs can be given as follows.

Definition 1.1. *The differential-algebraic equation $F(t, y, \dot{y}, \ddot{y}) = 0$ has* perturbation index (p-index) *m along a solution $y(t)$ on an interval $[0, \bar{t}]$, if m is the smallest integer such that for all functions $\hat{y}(t)$ having a defect $F(t, \hat{y}, \dot{\hat{y}}, \ddot{\hat{y}}) = \delta(t)$, with sufficiently small δ, there exists on $[0, \bar{t}]$ an estimate*

$$\|\hat{y}(t) - y(t)\| \leq C \left(\|\hat{y}(0) - y(0)\| + \max_{0\leq\xi\leq t} \|\delta(\xi)\| + \max_{0\leq\xi\leq t} \|\delta'(\xi)\| + \cdots + \max_{0\leq\xi\leq t} \|\delta^{(m-1)}(\xi)\| \right),$$

with a constant C.

The approach of transforming a second order DAE into a first order DAE leads to two major problems. On the one hand it may increase the index of the DAE [21, 24] and on the other hand the numerical method can fail [1, 2, 22]. To illustrate the first problem we consider the following trivial example, taken from [21]:

Example 1.2. *The linear second order system*

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \ddot{x}(t) + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \dot{x}(t) + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} x(t) = f(t), \quad t \in \mathbb{I}, \tag{2}$$

with $x(t) = [x_1(t), x_2(t)]^T$ and $f(t) = [f_1(t), f_2(t)]^T$ has the unique solution

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

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \dot{y}(t) + \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} y(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ 0 \\ 0 \end{bmatrix}, \tag{4}$$

which has the unique solution

$$\begin{cases} x_1(t) = f_2(t), \\ x_2(t) = f_1(t) - \dot{f}_2(t) - \ddot{f}_2(t), \\ v_1(t) = \dot{f}_2(t), \\ v_2(t) = \dot{f}_1(t) - \ddot{f}_2(t) - f_2^{(3)}(t). \end{cases} \tag{5}$$

The solution of the first order system requires the derivatives of f_2 up to order 3 and thus it is obvious that the p-index is 3, while the solution of the original system only requires $\ddot{f}_2$ and thus has p-index 2. If the degree of differentiability of the function f is limited, then the transformation to first order may be mathematically incorrect and 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 extra differentiation occurring in the solution of the first order system is responsible for the increase in the p-index. However, if we only introduce $v = \dot{x}_1$ as new variable, then no extra derivations of f are needed and, therefore, no increase in the p-index occurs.

As we see in Example 1.2, introducing only some derivatives in the transformation to first order systems may avoid an increase of the index, but in general we do not know which derivatives can be included without difficulties. To solve this problem a condensed form for linear higher order differential-algebraic systems is introduced in [21, 24], which allows an identification of those higher order derivatives of variables that can be replaced to obtain a first order system without changing the smoothness requirements or increasing the index.

The second problem, i.e., the failure of numerical methods is illustrated by the following example taken from [1, 22].

Example 1.3. *Consider the second order system*

$$\begin{aligned} \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} &= 2 \begin{bmatrix} y_2 \\ -y_1 \end{bmatrix} + \lambda \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \\ 0 &= y_1^2 + y_2^2 - 1. \end{aligned} \tag{6}$$

A corresponding first order system reads

$$\begin{aligned} \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} &= \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \\ \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} &= 2 \begin{bmatrix} y_2 \\ -y_1 \end{bmatrix} + \lambda \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \\ 0 &= y_1^2 + y_2^2 - 1. \end{aligned} \tag{7}$$

The equations model a particle on a circular track subject to a tangential force and form a semi-explicit differential-algebraic system of p-index 3.
For the numerical solution of the first order system (7) *we use BDF methods of order* $k = 1, 2$ *once with constant stepsize and once with variable stepsize. In Table 1 the absolute errors of the algebraic variable* λ *are presented, which are obtained by solving the first order system* (7) *with the BDF method of order* $k = 1$ *and the variable stepsizes given in the second column of Table 1. We can observe large errors (displayed in bold face), whenever there is a change in the stepsize. The reason for this effect is, that for problems of p-index* $\nu \geq 3$ *the BDF method with* $k = 1$ *(implicit Euler method) in general produces large errors in the first* $\nu - 2$ *steps after a change of stepsize. In Table 2 the absolute errors of the algebraic variable* λ *are given, which are obtained by solving the first order system* (7) *with constant stepsize BDF methods of order* $k = 1, 2$*. More precisely, we use the BDF method of order* $k = 1$ *in the first step and then BDF methods of order* $k = 2$ *in the following steps with the stepsize* $h = 0.005$ *in the first case and* $h = 0.01$ *in the second case. Here, we observe again high errors at the first step after the stepsize changes from* 0 *to* h*. In addition, large errors occur when the order is changed after the first step. In general, BDF methods for solving higher index DAEs may suffer from a reduction of the convergence order, whenever there is a change in the stepsize or in the order of the method, see [2].*

A third difficulty that arises in practice, is that the second order system 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 [14]. Furthermore, the transformation to first order leads to systems of double dimension and it may destroy certain structures, e.g. symmetries of the second order system as can be seen in Example 1.2.

In the following we will present numerical methods for the numerical solution of semi-explicit second order DAEs. First, we will describe numerical methods for the numerical solution of second order ordinary differential equations and extend these methods to the solution of second order semi-explicit DAEs. Altogether, we will consider BDF methods, Runge-Kutta methods and general linear methods. Finally, we give some numerical examples to demonstrate the behavior of the discussed methods.

Step no.	Stepsize $\times 10^{-3}$	$\lvert\lambda(t_j) - \lambda_j\rvert$ (BDF1)
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 1: Absolute errors for the algebraic variable λ in Example 1.3 obtained by BDF methods of order $k = 1$ with variable stepsizes.

Step no.	time t_n	$\lvert\lambda(t_j) - \lambda_j\rvert$ $h = 0.005$	$\lvert\lambda(t_j) - \lambda_j\rvert$ $h = 0.01$
1	1.0050	**2.0400**	
2	1.0100	**4.0190**	**2.0810**
3	1.0150	**1.0120**	
4	1.0200	0.0012	**4.0350**
5	1.0300	0.0013	**1.0280**
6	1.0400	0.0013	0.0052
7	1.0500	0.0014	0.0054

Table 2: Absolute errors for the algebraic variable λ in Example 1.3 obtained by constant stepsize BDF methods of order $k = 1, 2$. We use the BDF method of order $k = 1$ in the first step and then BDF methods of order $k = 2$ in the following steps with the stepsize $h = 0.005$ in the first case and $h = 0.01$ in the second case.

2 Numerical Methods for Second Order Systems

2.1 BDF Methods

For the direct numerical solution of second order differential-algebraic systems with BDF methods we follow an approach introduced in [22], where a variable-step version of the implicit Euler method for second order DAEs of p-index $\nu \in \{2, 3\}$ is introduced and a generalization to variable-step variable-order BDF methods is given. To explain this approach we consider the semi-explicit second order differential-algebraic initial value problem (1) of p-index 3.
Following the classical approach of transforming the second order system (1) into a first order system leads to

$$\begin{aligned} \dot{y}_0(t) &= y_1(t), \\ \dot{y}_1(t) &= f(t, y_0(t), y_1(t), \lambda(t)), \\ 0 &= g(t, y_0(t)), \end{aligned} \tag{8}$$

where $[y_0(t), y_1(t)] := [y(t), \dot{y}(t)]$.
We want to discretize systems (1) and (8) on an interval $\mathbb{I} = [t_0, t_0 + T]$, where $t_n = t_{n-1} + h_n$ for all $n = 1, \ldots, N$ and h_n denotes the stepsize in step n. First, we consider the first order system (8). Approximating the first order derivatives using the implicit Euler method (BDF

method of order $k = 1$) leads to the implicit Euler approximation

$$\begin{aligned} \frac{\hat{y}_{0,n} - \hat{y}_{0,n-1}}{t_n - t_{n-1}} &= \hat{y}_{1,n}, \\ \frac{\hat{y}_{1,n} - \hat{y}_{1,n-1}}{t_n - t_{n-1}} &= f(t_n, \hat{y}_{0,n}, \hat{y}_{1,n}, \lambda_n), \\ 0 &= g(t_n, \hat{y}_{0,n}), \end{aligned} \tag{9}$$

where $\hat{y}_{0,0} = \eta_0$ and $\hat{y}_{1,0} = \eta_1$. Here, $\hat{y}_{0,n}$, $\hat{y}_{1,n}$ and λ_n approximate $y_0(t_n)$, $y_1(t_n)$ and $\lambda(t_n)$, respectively. We have stated the difficulties which arise using this approach above, see also [2, 22]. For the direct numerical solution of the second order system (1) we now exchange order reduction and discretization. This means that the second order initial value problem (1) is first discretized by divided differences and after that the discrete equations are written as a system of equations. We denote by $y[t_n, \ldots, t_{n-k}]$ the k-th divided difference of y, which is recursively defined by

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

For divided differences and small stepsize $h_n = t_n - t_{n-1}$ it is well known that $k! y[t_n, \ldots, t_{n-k}] \approx y^{(k)}(t_n)$, see [6], and thus we get the approximation

$$2! y[t_n, \ldots, t_{n-2}] = f(t_n, y_{0,n}, y_{1,n}, \lambda_n) + R, \tag{10a}$$

$$0 = g(t_n, y_{0,n}), \tag{10b}$$

where R denotes a remainder term. Here, $y_{j,n}$ is an approximation to $y^{(j)}(t_n)$ for $j = 0, 1$ and λ_n is an approximation to $\lambda(t_n)$. With the relation

$$y_{j,n} \approx j! y[t_n, \ldots, t_{n-j}] \approx \frac{y_{j-1,n} - y_{j-1,n-1}}{(t_n - t_{n-j})/j},$$

we can write (10a) as a system of equations and obtain the discretization

$$\begin{aligned} \frac{y_{0,n} - y_{0,n-1}}{t_n - t_{n-1}} &= y_{1,n}, \\ \frac{y_{1,n} - y_{1,n-1}}{(t_n - t_{n-2})/2} &= f(t_n, y_{0,n}, y_{1,n}, \lambda_n), \\ 0 &= g(t_n, y_{0,n}), \end{aligned} \tag{11}$$

where $y_{j,0} = \eta_j$ for $j = 0, 1$ and $t_m = t_0$ if $m < 0$.
In contrast to approximation (9), which only depends on the current stepsize, the modified Euler approximation (11) considers a mean value of the current stepsize h_n and the previous stepsize $h_{n-1} = t_{n-1} - t_{n-2}$ for the approximation of $y_{2,n}$. Thus, for constant stepsizes the two approximations are the same. It is interesting to note that method (11) differs from the implicit Euler approximation (9) only in the first step after a change of stepsize. If we have a system of p-index 3, this is just the step where the implicit Euler method (9) produces large errors as we have seen in Example 1.3.
Method (11) can be generalized to variable-step variable-order methods based on BDF methods. To derive the variable-step variable-order methods given in [22] 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

$$\sum_{i=1}^{k} \prod_{m=1}^{i-1} (t_n - t_{n-m}) y_0[t_n, t_{n-1}, \ldots, t_{n-i}] = y_{1,n}. \tag{12}$$

For $j = 1$ this BDF method of order k has to be modified in order to produce exact values of $\ddot{y}(t_n)$ for polynomials $y(t)$ of degree $k+1$. The resulting method is given by

$$\sum_{i=1}^{k} \alpha_{i,n}^{[2]} y_1[t_n, t_{n-1}, \ldots, t_{n-i}] = y_{2,n}, \tag{13}$$

where the coefficients $\alpha_{i,n}^{[2]}$ are the BDF-coefficients for the approximation of $y_{2,n}$. In order to obtain exact solutions for polynomials of degree up to $k+1$, these coefficients $\alpha_{i,n}^{[2]}$ have to satisfy the conditions

$$\sum_{i=1}^{k} \alpha_{i,n}^{[2]} \tilde{y}_1[t_n, t_{n-1}, \ldots, t_{n-i}] = \frac{(s+1)!}{(s-1)!} t_n^{s-1} \tag{14}$$

for $\tilde{y}_0(t) = t^{s+1}$ and $s = 1, 2, \ldots, k$. In the same way, variable-step variable-order BDF methods for systems of arbitrary high order can be constructed.
Using (12) for the approximation of the first order derivatives and (13) for the approximation of the second order derivatives, a *variable-step, variable-order BDF discretization of the second order semi-explicit system* (1) is given by

$$\begin{aligned} \sum_{i=1}^{k} \alpha_{i,n}^{[2]} y_1[t_n, t_{n-1}, \ldots, t_{n-i}] &= f(t_n, y_{0,n}, y_{1,n}, \lambda_n), \\ 0 &= g(t_n, y_{0,n}), \\ y_{1,n} &= \sum_{i=1}^{k} \prod_{m=1}^{i-1} (t_n - t_{n-m}) y_0[t_n, t_{n-1}, \ldots, t_{n-i}], \end{aligned} \tag{15}$$

for given initial values $y_{0,0}$, $y_{1,0}$ and λ_0 and with the unknowns $y_{1,n}$, $y_{0,n}$ and λ_n at time t_n and the divided differences

$$\begin{aligned} y_0[t_n, t_{n-1}, \ldots, t_{n-i}] &= \sum_{j=0}^{i} \alpha_j (y_{0,n-j+1} - y_{0,n-j}), \\ y_1[t_n, t_{n-1} \ldots, t_{n-i}] &= \sum_{j=0}^{i} \hat{\alpha}_j (y_{1,n-j+1} - y_{1,n-j}), \end{aligned} \tag{16}$$

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

For the BDF method (13) of fixed order k we get different coefficients $\alpha_{i,n}^{[2]}$, as the calculation of the $\alpha_{i,n}^{[2]}$ depends on the current and previous stepsizes and on the accuracies of the last taken steps. This means that we have to consider the order of the ordinary BDF methods used for the approximation of $y_{1,n}, \ldots, y_{1,n-i}$ occurring in (14). As the coefficients $\alpha_{i,n}^{[2]}$ depend on the current stepsizes, we have to solve a linear system of equations to determine $\alpha_{i,n}^{[2]}$ in each step, provided the stepsize has changed in one of the last taken steps. This linear system is of the form

$$A\alpha = b, \tag{17}$$

where

$$b = \begin{bmatrix} 2 \\ 6t_n \\ \vdots \\ \frac{(k+1)!}{(k-1)!} t_n^{k-1} \end{bmatrix}, \quad A = \left[\tilde{y}_1^{(i)}[t_n, \ldots, t_{n-j}] \right]_{i,j=1,..,k}, \quad \alpha = \left[\alpha_{1,n}^{[2]}, \ldots, \alpha_{k,n}^{[2]} \right]^T \tag{18}$$

arising from (14). Here, we have used

$$\tilde{y}_{1,s}^{(i)} = \sum_{l=1}^{k_s} \prod_{m=1}^{l-1} (t_s - t_{s-m}) \tilde{y}_0^{(i)}[t_s, .., t_{s-l}] \quad \text{for } s = n, .., n-j, \tag{19}$$

with $\tilde{y}_0^{(i)}(t_n) = t_n^{i+1}$ in the divided differences occurring in A. In this formula, k_s denotes the order of the BDF method used for approximating $\tilde{y}_{1,s}^{(i)}$ for $s = n, \ldots, n-j$. If we only write k we will always mean the order k_n of the current step n.
The algorithm for the numerical solution of system (15) will be briefly described in the following. In every step of the integration procedure we have to solve a nonlinear system of equations with unknown values $x_n = [y_{0,n}^T, \lambda_{0,n}^T, y_{1,n}^T]^T$ at time t_n. This nonlinear system can be written as

$$F(x_n) = \begin{bmatrix} \sum_{i=1}^{k} \alpha_{i,n}^{[2]} y_1[t_n, \ldots, t_{n-i}] - f(t_n, y_{0,n}, y_{1,n}, \lambda_n) \\ g(t_n, y_{0,n}) \\ y_{1,n} - \sum_{i=1}^{k} \prod_{m=1}^{i-1} (t_n - t_{n-m}) y_0[t_n, \ldots, t_{n-i}] \end{bmatrix} = 0. \tag{20}$$

In each time step this nonlinear system may be iteratively solved for x_n using Newton's method, i.e.,

$$x_n^{(m+1)} = x_n^{(m)} - DF(x_n^{(m)})^{-1} F(x_n^{(m)}), \tag{21}$$

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

$$DF(x_n^{(m)}) \underbrace{(x_n^{(m+1)} - x_n^{(m)})}_{:= \Delta x^{(m+1)}} = -F(x_n^{(m)}). \tag{22}$$

The new iterate $x_n^{(m+1)}$ is then given by $x_n^{(m+1)} = x_n^{(m)} + \Delta x^{(m+1)}$. The solution of the linear system (22) requires the evaluation of the Jacobian $DF(x_n)$ in each step. For system (20) the Jacobian $DF(x_n)$ is given by

$$DF(x_n) = \begin{bmatrix} -f_{y_0} & -f_{\lambda_0} & \sum_{i=1}^{k} \alpha_{i,n}^{[2]} \frac{1}{\prod_{m=1}^{i} (t_n - t_{n-m})} - f_{y_1} \\ g_{y_0} & 0 & 0 \\ -\sum_{i=1}^{k} \frac{1}{t_n - t_{n-i}} & 0 & 1 \end{bmatrix}. \tag{23}$$

Here, f_x denotes the partial derivative of f with respect to x, i.e., $f_x(.) := \frac{\partial}{\partial x} f(.)$ and for simplicity we have omitted the arguments of the functions and the index n.
The initial guess $x_n^{(0)}$ for the Newton method is formed by evaluating a predictor polynomial $w_p(t)$ at t_n, which interpolates x_{n-1-i} at the last $k+1$ time steps t_{n-1-i}, i.e., $w_p(t_{n-1-i}) = x_{n-1-i}$, for $i = 0, \ldots, k$. For the termination of the Newton iteration we use the same strategy that is used in the code DASSL and is described in [2]. The following algorithm in MATLAB [20] notation summarizes the procedure.

Algorithm *Modified BDF method for second order systems*

INPUT: $X = [x_{n-k_n}, \ldots, x_{n-1}]$, $T = [t_{n-k_n}, \ldots, t_n]$
OUTPUT: $x_n = [y_{0,n}^T, \lambda_{0,n}^T, y_{1,n}^T]^T$

1: Determine b, A as in (18);
2: $\alpha = A \backslash b$;
3: $x^{(0)} = w_p(X, T)$; {starting value for Newton Iteration}
4: $m = 0$;
5: **repeat** {Newton Iteration}

6: Determine $F(x^{(m)})$, $DF(x^{(m)})$ as in (20), (23);
7: $[L, U, P] = lu(DF)$;
8: $f = -P \cdot F$;
9: $z = L \backslash f$;
10: $\Delta x^{(m+1)} = U \backslash z$;
11: $x^{(m+1)} = x^{(m)} + \Delta x^{(m+1)}$;
12: $m = m + 1$;
13: **until** converged
14: $x_n = x^{(m)}$;

Here, $lu(DF)$ denotes the LU-decomposition of the matrix DF into an upper and lower triangular matrix, multiplied with a permutation matrix, and $A \backslash b$ denotes Gaussian elimination with partial pivoting as used in MATLAB.

In [22] existence and uniqueness of the solution as well as convergence of the modified implicit Euler methods, i.e., the modified BDF method of order k=1, applied to a d-th order semi-explicit differential-algebraic system of index $\nu = d + 1$ has been shown. Further, multistep methods for second order ordinary differential equations are treated in [12]. For the special case of second order ordinary differential equations, where the first derivative does not occur explicitly on the right-hand side there are stability, consistency and convergence results as well as order conditions given in [12, 17].

Remark 2.1. *A proof of convergence for the modified BDF methods of higher order and the examination of stability properties of these methods when applied to second order differential-algebraic systems are currently under investigation.*

2.2 Runge-Kutta Methods

To derive Runge-Kutta methods for second order DAEs we will use the concept of collocation for the second order initial value problem. The concept of Runge-Kutta methods as collocation methods is described in detail in [14]. The basic idea of collocation is to find a polynomial of degree s whose derivative coincides at s given points with the vector field of the differential equation. We will first derive the Runge-Kutta methods for second order ordinary differential equations and later extend the approach to DAEs.
Consider the second order initial value problem

$$\begin{aligned} \ddot{y}(t) &= f(t, y(t), \dot{y}(t)), \\ y(t_0) &= \eta_0, \quad \dot{y}(t_0) = \eta_1, \end{aligned} \tag{24}$$

with a function $f : [t_0, t_0 + T] \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ on the interval $[t_0, t_0 + h]$. The polynomial $w_s(t)$ of degree s is a collocation polynomial for the initial value problem (24) if it satisfies the conditions

$$\begin{aligned} w_s(t_0) &= \eta_0, \\ \dot{w}_s(t_0) &= \eta_1, \\ \ddot{w}_s(t_0 + c_i h) &= f(t_0 + c_i h, w_s(t_0 + c_i h), \dot{w}_s(t_0 + c_i h)). \end{aligned} \tag{25}$$

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

$$\begin{aligned} \ddot{w}_s(t_0 + xh) &= \sum_{j=1}^{s} \ddot{w}_s(t_0 + c_j h) L_j(x), \\ \dot{w}_s(t_0 + xh) &= \sum_{j=1}^{s} \dot{w}_s(t_0 + c_j h) L_j(x), \quad \text{for } x \in [0, 1], \end{aligned} \tag{26}$$

with the Lagrange polynomials

$$L_j(x) = \prod_{\substack{l=1 \\ l \neq j}}^{s} \frac{x - c_l}{c_j - c_l}, \quad j = 1, \ldots, s. \tag{27}$$

Integrating the equations (26), it follows that

$$\begin{aligned} \dot{w}_s(t_0 + c_i h) &= \dot{w}_s(t_0) + h \int_0^{c_i} \sum_{j=1}^{s} \ddot{w}_s(t_0 + c_j h) L_j(x) dx \\ &= \dot{w}_s(t_0) + h \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, w_s(t_0 + c_j h), \dot{w}_s(t_0 + c_j h)), \end{aligned} \tag{28a}$$

$$\dot{w}_s(t_0 + h) = \dot{w}_s(t_0) + h \sum_{i=1}^{s} b_i f(t_0 + c_i h, w_s(t_0 + c_i h), \dot{w}_s(t_0 + c_i h)), \tag{28b}$$

where the coefficients a_{ij} and b_j are defined as

$$a_{ij} = \int_0^{c_i} L_j(x) dx, \quad b_j = \int_0^1 L_j(x) dx, \quad i, j = 1, \ldots, s. \tag{28c}$$

Further integration yields

$$\begin{aligned} w_s(t_0 + c_i h) &= w_s(t_0) + h \int_0^{c_i} \sum_{j=1}^{s} \dot{w}_s(t_0 + c_j h) L_j(x) dx \\ &= w_s(t_0) + h c_i \dot{w}_s(t_0) + h^2 \sum_{l=1}^{s} \tilde{a}_{il} f(t_0 + c_l h, w_s(t_0 + c_l h), \dot{w}_s(t_0 + c_l h)), \end{aligned} \tag{28d}$$

$$w_s(t_0 + h) = w_s(t_0) + h \dot{w}_s(t_0) + h^2 \sum_{j=1}^{s} \tilde{b}_j f(t_0 + c_j h, w_s(t_0 + c_j h), \dot{w}_s(t_0 + c_j h)), \tag{28e}$$

where we have used that $\sum_{i=1}^{s} b_i = 1$ and $\sum_{j=1}^{s} a_{ij} = c_i$ and have defined the coefficients $\tilde{a}_{il}$ and $\tilde{b}_j$ via

$$\tilde{a}_{il} = \sum_{j=1}^{s} a_{ij} a_{jl} \text{ and } \tilde{b}_j = \sum_{i=1}^{s} b_i a_{ij}, \quad i, j, l = 1, \ldots, s. \tag{28f}$$

Using the formulas (28) in each integration step we get an s-stage, implicit Runge-Kutta method for the second order system (24) as

$$\begin{aligned} y_n &= y_{n-1} + h \dot{y}_{n-1} + h^2 \sum_{i=1}^{s} \tilde{b}_i f(t_{n-1} + c_i h, Y_{ni}, \dot{Y}_{ni}), \\ \dot{y}_n &= \dot{y}_{n-1} + h \sum_{i=1}^{s} b_i f(t_{n-1} + c_i h, Y_{ni}, \dot{Y}_{ni}), \\ Y_{ni} &= y_{n-1} + h c_i \dot{y}_{n-1} + h^2 \sum_{j=1}^{s} \tilde{a}_{ij} f(t_{n-1} + c_j h, Y_{nj}, \dot{Y}_{nj}), \\ \dot{Y}_{ni} &= \dot{y}_{n-1} + h \sum_{j=1}^{s} a_{ij} f(t_{n-1} + c_j h, Y_{nj}, \dot{Y}_{nj}), \end{aligned} \tag{29}$$

where we use the notation $y_n = w_s(t_{n-1} + h)$, $\dot{y}_n = \dot{w}_s(t_{n-1} + h)$ and $Y_{ni} = w_s(t_{n-1} + c_i h)$, $\dot{Y}_{ni} = \dot{w}_s(t_{n-1} + c_i h)$ for $i = 1, \ldots, s$.

Definition 2.2. *The Runge-Kutta method* (29) *is said to have* order p*, if for sufficiently smooth problems* (24) *it holds that*

$$y(t_0+h)-y_1=O(h^{p+1}),$$
$$\dot{y}(t_0+h)-\dot{y}_1=O(h^{p+1}).$$

Remark 2.3. *Just demanding order $p-1$ for the approximation of the derivative is not sufficient, since the approximation to the derivative enters the approximation to the solution itself. If the first is of order $p-1$ and the second of order p, then the overall approximation to the solution will be of order $p-1$.*

Convergence of the Runge-Kutta methods (29) applied to the second order ordinary differential equation (24) is treated in [12], where order conditions are specified. These order conditions can be given as simplifying assumptions similar to the first order case, see [12]:

$$B(p): \quad \sum_{i=1}^{s} b_i c_i^{k-1} = \frac{1}{k}, \text{ for } k=1,\dots,p,$$
$$C(q): \quad \sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{c_i^k}{k}, \text{ for } k=1,\dots,q, \quad i=1,\dots,s,$$
$$D(\xi): \quad \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = \frac{b_j}{k}(1-c_j^k), \text{ for } k=1,\dots,\xi, \quad j=1,\dots,s,$$
$$\tilde{B}(p): \quad \sum_{i=1}^{s} \tilde{b}_i c_i^{k-1} = \frac{1}{(k+1)k}, \text{ for } k=1,\dots,p-1,$$
$$\tilde{C}(q): \quad \sum_{j=1}^{s} \tilde{a}_{ij} c_j^{k-1} = \frac{c_i^{k+1}}{(k+1)k}, \text{ for } k=1,\dots,q-1, \quad i=1,\dots,s,$$
$$\tilde{D}(\xi): \quad \sum_{i=1}^{s} b_i c_i^{k-1} \tilde{a}_{ij} = b_j \frac{c_i^{k+1}}{(k+1)k} - \frac{c_i}{k} + \frac{1}{k+1}, \text{ for } k=1,\dots,\xi-1, \quad j=1,\dots,s.$$

These conditions are purely algebraic conditions for the coefficients of the Runge-Kutta method (29). If $C(q)$ and $\tilde{C}(q)$ are satisfied, then q is called the *stage order* of the method. For the notation of the coefficients we use an adaption to the general notation of Runge-Kutta coefficients using the so-called *Butcher diagram*, which is given by

c_1	a_{11}	$\dots$	a_{1s}	$\tilde{a}_{11}$	$\dots$	$\tilde{a}_{1s}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	$\dots$	a_{ss}	$\tilde{a}_{s1}$	$\dots$	$\tilde{a}_{ss}$
	b_1	$\dots$	b_s	$\tilde{b}_1$	$\dots$	$\tilde{b}_s$

or by

c	A	$\tilde{A}$
	b^T	$\tilde{b}^T$

where $A=[a_{ij}]$, $\tilde{A}=[\tilde{a}_{ij}]$, $b=[b_1,b_2,\dots,b_s]^T$, $\tilde{b}=[\tilde{b}_1,\tilde{b}_2,\dots,\tilde{b}_s]^T$ and $c=[c_1,c_2,\dots,c_s]^T$.

Example 2.4. *We give the coefficients for two Runge-Kutta methods of the form* (29) *with $s=2$. To do this, we choose known Runge-Kutta collocation methods such as Gauss and Radau-IIA methods [14] for the approximation of $\dot{y}_n$ with coefficients a_{ij}, b_i and c_i and determine $\tilde{a}_{ij}$ and $\tilde{b}_i$ using the formulas* (28f)*. This gives the following methods.*

Gauss

$\frac{3-\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{3-2\sqrt{3}}{12}$	$\frac{1}{24}$	$\frac{3-2\sqrt{3}}{24}$
$\frac{3+\sqrt{3}}{6}$	$\frac{3+2\sqrt{3}}{12}$	$\frac{1}{4}$	$\frac{3+2\sqrt{3}}{24}$	$\frac{1}{24}$
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3+\sqrt{3}}{12}$	$\frac{3-\sqrt{3}}{12}$

Radau-IIA

$\frac{1}{3}$	$\frac{5}{12}$	$\frac{-1}{12}$	$\frac{1}{9}$	$-\frac{1}{18}$
1	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0
	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	0

The Gauss method is of order $p = 4$ and the Radau method is of order $p = 3$.

After the development of Runge-Kutta methods for second order ordinary differential equations, we can modify these methods for the numerical solution of the second order semi-explicit differential-algebraic system (1). A Runge-Kutta method (29) applied to (1) can be written in the form

$$\begin{aligned} y_n &= y_{n-1} + h\dot{y}_{n-1} + h^2 \sum_{i=1}^{s} \tilde{b}_i \ddot{Y}_{ni}, \\ \dot{y}_n &= \dot{y}_{n-1} + h \sum_{i=1}^{s} b_i \ddot{Y}_{ni}, \\ \lambda_n &= \lambda_{n-1} + h\dot{\lambda}_{n-1} + h^2 \sum_{i=1}^{s} \tilde{b}_i \ddot{\Lambda}_{ni}, \\ \dot{\lambda}_n &= \dot{\lambda}_{n-1} + h \sum_{i=1}^{s} b_i \ddot{\Lambda}_{ni}, \end{aligned} \tag{30a}$$

where

$$\begin{aligned} \ddot{Y}_{ni} &= f(t_{n-1} + c_i h, Y_{ni}, \dot{Y}_{ni}, \Lambda_{ni}), \\ 0 &= g(t_{n-1} + c_i h, Y_{ni}), \end{aligned} \tag{30b}$$

and the internal stages are given by

$$\begin{aligned} Y_{ni} &= y_{n-1} + hc_i\dot{y}_{n-1} + h^2 \sum_{j=1}^{s} \tilde{a}_{ij} \ddot{Y}_{nj}, \\ \dot{Y}_{ni} &= \dot{y}_{n-1} + h \sum_{j=1}^{s} a_{ij} \ddot{Y}_{nj}, \\ \Lambda_{ni} &= \lambda_{n-1} + hc_i\dot{\lambda}_{n-1} + h^2 \sum_{j=1}^{s} \tilde{a}_{ij} \ddot{\Lambda}_{nj} \end{aligned} \tag{30c}$$

for $i = 1, \ldots, s$ and given initial values.

For the implementation it is useful to write method (30) in a slightly different form. We summarize the variables into $x = [y^T, \lambda^T]^T$ and define $X_n = [X_{n1}^T, \ldots, X_{ns}^T]^T$ with $X_{ni} = [Y_{ni}^T, \Lambda_{ni}^T]^T$ (the same holds for $\dot{X}_n$ and $\ddot{X}_n$). Further, we use the Kronecker product of two matrices defined as follows.

Definition 2.5. *Let A be a $n \times m$ matrix with entries a_{ij} and let B be a $p \times q$ matrix. Then the* Kronecker product *of A and B is the $np \times mq$ block matrix*

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{bmatrix}. \tag{31}$$

This means that for a term $(A \otimes I)V = W$ with vectors $V = [v_1, v_2, \dots, v_k]^T$ and $W = [w_1, w_2, \dots, w_k]^T$ the coefficients of W are given by $w_i = \sum_{j=1}^{k} a_{ij} v_j$ for $i = 1, \dots, k$.

Using these conventions, method (30) can be written as

$$\begin{aligned} x_n &= x_{n-1} + h\dot{x}_{n-1} + h^2(\tilde{b}^T \otimes I_m)\ddot{X}_n, \\ \dot{x}_n &= \dot{x}_{n-1} + h(b^T \otimes I_m)\ddot{X}_n, \end{aligned} \tag{32a}$$

where $\ddot{X}_n$ is given by the nonlinear system

$$F(X_n, \dot{X}_n, \ddot{X}_n) = \begin{bmatrix} F(X_{n1}, \dot{X}_{n1}, \ddot{X}_{n1}) \\ \\ \vdots \\ \\ F(X_{ns}, \dot{X}_{ns}, \ddot{X}_{ns}) \end{bmatrix} = \begin{bmatrix} \ddot{Y}_{n1} - f(t_{n-1} + c_1 h, Y_{n1}, \dot{Y}_{n1}, \Lambda_{n1}) \\ g(t_{n-1} + c_1 h, Y_{n1}) \\ \vdots \\ \ddot{Y}_{ns} - f(t_{n-1} + c_s h, Y_{ns}, \dot{Y}_{ns}, \Lambda_{ns}) \\ g(t_{n-1} + c_s h, Y_{ns}) \end{bmatrix} = 0, \tag{32b}$$

with internal stages given by

$$\begin{aligned} X_n &= (e_s \otimes I_m)x_{n-1} + h(c \otimes I_m)\dot{x}_{n-1} + h^2(\tilde{A} \otimes I_m)\ddot{X}_n, \\ \dot{X}_n &= (e_s \otimes I_m)\dot{x}_{n-1} + h(A \otimes I_m)\ddot{X}_n. \end{aligned} \tag{32c}$$

Here, $m = m_y + m_\lambda$ is the dimension of the system, I_m is the m-dimensional identity matrix and $e_s = [1, \dots, 1]^T$ of dimension s.

In each step of the integration procedure the nonlinear system (32b) has to be solved in order to obtain values $\ddot{X}_n$, which can then be inserted into (32a) for the determination of the new values x_n and $\dot{x}_n$. Again, this nonlinear system may be solved by means of a Newton method, where we use the Jacobian of F given by

$$DF(X_n, \dot{X}_n, \ddot{X}_n) = \{F_{\ddot{X}}\} + h^2(\tilde{A} \otimes I_m)\{F_X\} + h(A \otimes I_m)\{F_{\dot{X}}\}. \tag{33}$$

Here, we have used the notation

$$\{F_y\} = \mathit{blockdiag}\left(\frac{\partial F}{\partial y}(X_{n1}, \dot{X}_{n1}, \ddot{X}_{n1}), \dots, \frac{\partial F}{\partial y}(X_{ns}, \dot{X}_{ns}, \ddot{X}_{ns})\right).$$

The remaining procedure is analogous to that described for the BDF methods. One time step in the Runge-Kutta integration is summarized in the following algorithm.

Algorithm *Runge-Kutta method for second order systems*

INPUT: $x_{n-1} = [y_{n-1}^T, \lambda_{n-1}^T]^T$, $\dot{x}_{n-1} = [\dot{y}_{n-1}^T, \dot{\lambda}_{n-1}^T]^T$, t_{n-1}, h,
OUTPUT: $x_n = [y_n^T, \lambda_n^T]^T$, $\dot{x}_n = [\dot{y}_n^T, \dot{\lambda}_n^T]^T$,

1: $t_n = t_{n-1} + h$;
2: Get Runge-Kutta coefficients A, $\tilde{A}$, b, $\tilde{b}$, c;

3: $x^{(0)} = [0, \ldots, 0]^T$; {starting value for Newton Iteration}
4: $k = 0$;
5: **repeat** {Newton Iteration}
6: $\quad \ddot{X} = x^{(k)}$;
7: $\quad X = (e_s \otimes I_m)x_{n-1} + h(c \otimes I_m)\dot{x}_{n-1} + h^2(\tilde{A} \otimes I_m)\ddot{X}$;
8: $\quad \dot{X} = (e_s \otimes I_m)\dot{x}_{n-1} + h(A \otimes I_m)\ddot{X}$;
9: $\quad$ Compute $F(X, \dot{X}, \ddot{X})$ as in (32b);
10: $\quad$ Compute $DF(X, \dot{X}, \ddot{X})$ as in (33);
11: $\quad$ Scale $DF = 1/h^2 DF; \quad F = 1/h^2 F$;
12: $\quad [L, U, P] = lu(DF)$;
13: $\quad f = -P \cdot F$;
14: $\quad z = L \backslash f$;
15: $\quad \Delta x^{(k+1)} = U \backslash z$;
16: $\quad x^{(k+1)} = x^{(k)} + \Delta x^{(k+1)}$;
17: $\quad k = k + 1$;
18: **until** converged
19: $x_n = x_{n-1} + h\dot{x}_{n-1} + h^2(\tilde{b}^T \otimes I_m)x^{(k)}$;
20: $\dot{x}_n = \dot{x}_{n-1} + h(b^T \otimes I_m)x^{(k)}$;

The scaling in line 11 improves the condition number of the iteration matrix as for small stepsizes the iteration matrix is close to singular due to the factor h^2.

One-step methods for ordinary differential equations of higher order as e.g., Taylor's series methods, Runge-Kutta methods and Nyström methods with convergence and consistency results are treated in [8, 17, 28]. The algebraic theory concerning the relevant order conditions has been studied in some particular cases; see for example [16, 15, 28] where Runge-Kutta methods for differential systems of arbitrary order are discussed. Nyström methods [12, 13] are specially adapted to second order equations and are similar to the Runge-Kutta methods (29), but the coefficients do not necessarily satisfy condition (28f). Convergence and order conditions for Nyström methods applied to second order ordinary differential equations are treated in [12]. For special second order systems, where the right-hand side does not depend explicitly on the first derivative, the direct methods are shown to be more efficient than the classical Runge-Kutta methods.

Remark 2.6. *Convergence results and stability properties for Runge-Kutta methods applied to second order differential-algebraic equations are currently under investigation.*

2.3 General Linear Methods

The same approach as for Runge-Kutta methods for the numerical solution of second order differential-algebraic systems can be used to construct general linear methods. General linear methods can be seen as a generalization of multistep methods (e.g. BDF methods) and multistage methods (e.g. Runge-Kutta methods). For an introduction to general linear methods see [3, 12]. Again, we describe the method for second order ordinary differential equations and extend this approach to DAEs.

General linear methods are both multistage and multivalued methods. In order to obtain methods for second order systems we model the method in the style of a Runge-Kutta method, but with a number of k input quantities imported into and going out of step n, instead of just one as in the case of Runge-Kutta methods. For ease of representation we restrict ourselves to autonomous systems of second order ordinary differential equations and consider initial value problems of the form

$$\ddot{y}(t) = f(y(t), \dot{y}(t)), \quad y(t_0) = y_0, \quad \dot{y}(t_0) = y_1. \tag{34}$$

This is no limitation as each non-autonomous system can always be tranformed into an autonomous system by introducing the time t as new variable, without changing the solution behavior, see [18], Lemma 4.8.
A k-step s-stage general linear method for the numerical solution of the second order initial value problem (34) is then given by

$$y_{n,i} = \sum_{j=1}^{k} \tilde{v}_{ij} y_{n-1,j} + h^2 \sum_{j=1}^{s} \tilde{b}_{ij} f(Y_{n,j}, \dot{Y}_{n,j}), \tag{35a}$$

for $i = 1, \ldots, k$, with internal stages given by

$$\begin{aligned} Y_{n,i} &= \sum_{j=1}^{k} \tilde{u}_{ij} y_{n-1,j} + h^2 \sum_{j=1}^{s} \tilde{a}_{ij} f(Y_{n,j}, \dot{Y}_{n,j}), \\ \dot{Y}_{n,i} &= \sum_{j=1}^{k} u_{ij} y_{n-1,j} + h \sum_{j=1}^{s} a_{ij} f(Y_{n,j}, \dot{Y}_{n,j}), \end{aligned} \tag{35b}$$

for $i = 1, \ldots, s$, where $h > 0$ denotes the stepsize and $t_n = nh$. If we define the coefficient matrices

$$\begin{aligned} &\tilde{\mathcal{V}} = [\tilde{v}_{ij}] \in \mathbb{R}^{k \times k}, \quad \tilde{\mathcal{B}} = [\tilde{b}_{ij}] \in \mathbb{R}^{k \times s}, \quad \tilde{\mathcal{U}} = [\tilde{u}_{ij}] \in \mathbb{R}^{s \times k}, \\ &\tilde{\mathcal{A}} = [\tilde{a}_{ij}] \in \mathbb{R}^{s \times s}, \quad \mathcal{U} = [u_{ij}] \in \mathbb{R}^{s \times k}, \quad \mathcal{A} = [a_{ij}] \in \mathbb{R}^{s \times s}, \end{aligned}$$

and the vector $c = [c_1, \ldots, c_s]^T$, then we can write method (35) in a more compact form as

$$\begin{aligned} y^{[n]} &= (\tilde{\mathcal{V}} \otimes I_m) y^{[n-1]} + h^2 (\tilde{\mathcal{B}} \otimes I_m) F(Y_n, \dot{Y}_n), \\ Y_n &= (\tilde{\mathcal{U}} \otimes I_m) y^{[n-1]} + h^2 (\tilde{\mathcal{A}} \otimes I_m) F(Y_n, \dot{Y}_n), \\ \dot{Y}_n &= (\mathcal{U} \otimes I_m) y^{[n-1]} + h (\mathcal{A} \otimes I_m) F(Y_n, \dot{Y}_n), \end{aligned} \tag{36}$$

where m is again the dimension of the system and

$$F(Y_n, \dot{Y}_n) = \begin{bmatrix} f(Y_{n,1}, \dot{Y}_{n,1}) \\ \vdots \\ f(Y_{n,s}, \dot{Y}_{n,s}) \end{bmatrix}, \; y^{[n]} = \begin{bmatrix} y_{n,1} \\ \vdots \\ y_{n,k} \end{bmatrix}, \; Y_n = \begin{bmatrix} Y_{n,1} \\ \vdots \\ Y_{n,s} \end{bmatrix} \text{ and } \dot{Y}_n = \begin{bmatrix} \dot{Y}_{n,1} \\ \vdots \\ \dot{Y}_{n,s} \end{bmatrix}.$$

In the following we will remove the Kronecker product with I_m for ease of representation and get the following representation of the general linear method

$$\begin{aligned} y^{[n]} &= \tilde{\mathcal{V}} y^{[n-1]} + h^2 \tilde{\mathcal{B}} F(Y_n, \dot{Y}_n), \\ Y_n &= \tilde{\mathcal{U}} y^{[n-1]} + h^2 \tilde{\mathcal{A}} F(Y_n, \dot{Y}_n), \\ \dot{Y}_n &= \mathcal{U} y^{[n-1]} + h \mathcal{A} F(Y_n, \dot{Y}_n). \end{aligned} \tag{37}$$

It is also conventional to write the coefficients of the method as a matrix

$$\mathcal{M} = \begin{bmatrix} \tilde{\mathcal{A}} & \tilde{\mathcal{U}} \\ \mathcal{A} & \mathcal{U} \\ \tilde{\mathcal{B}} & \tilde{\mathcal{V}} \end{bmatrix}. \tag{38}$$

Example 2.7. *If we consider the Runge-Kutta method* (29) *as a general linear method, then the coefficient matrices in* (37) *are given by*

$$\begin{aligned} &\tilde{\mathcal{V}} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \tilde{\mathcal{U}} = \begin{bmatrix} 1 & c_1 \\ 1 & c_2 \end{bmatrix}, \quad \mathcal{U} = \begin{bmatrix} 0 & \frac{1}{h} \\ 0 & \frac{1}{h} \end{bmatrix}, \quad \tilde{\mathcal{B}} = \begin{bmatrix} \tilde{b}_1 & \ldots & \tilde{b}_s \\ b_1 & \ldots & b_s \end{bmatrix}, \\ &\tilde{\mathcal{A}} = [\tilde{a}_{ij}] \in \mathbb{R}^{s,s}, \quad \mathcal{A} = [a_{ij}] \in \mathbb{R}^{s,s}, \quad c = [c_1, \ldots, c_s]^T. \end{aligned} \tag{39}$$

In (39) *the coefficients a_{ij}, b_i, $\tilde{a}_{ij}$, $\tilde{b}_i$ and c_i are the coefficients of the Runge-Kutta method.*

The general linear method (37) only makes sense for second order systems, if we require that the quantities passed from step to step are in Nordsieck form, i.e.,

$$y^{[n]} \approx \begin{bmatrix} y(t_n) \\ h\dot{y}(t_n) \\ h^2\ddot{y}(t_n) \\ \vdots \\ h^{k-1}y^{(k-1)}(t_n) \end{bmatrix}. \tag{40}$$

In addition to the approximation scheme (37) we need a so-called *correct value function* and a *starting procedure* for the numerical solution of the second order system (34). We define the *correct value function* $z(t,h)$, which takes values in $\mathbb{R}^{mk}$. Each of the k components of $z(t,h)$ represents an m dimensional function, which relates in some way to the exact solution $y(t)$. This means that the correct value function gives an interpretation of the values $y^{[n]}$, i.e., $z_n = z(t_n,h)$ is approximated by $y^{[n]}$, so that the global error is given by $y^{[n]} - z_n$. The exact value function for a method in Nordsieck form is therefore given by

$$z(t,h) = \begin{bmatrix} y(t) \\ h\dot{y}(t) \\ \vdots \\ h^{k-1}y^{(k-1)}(t) \end{bmatrix}. \tag{41}$$

Further, the internal stages $Y_{n,i}$ and $\dot{Y}_{n,i}$ approximate $y(t_n + c_ih)$ and $\dot{y}(t_n + c_ih)$ for $i = 1,\dots,s$, where $y(t)$ is the exact solution of (34).
In general, only the initial values $y(t_0) = y_0$ and $\dot{y}(t_0) = y_1$ are given, so that we need a *starting procedure* $S(h)$, which specifies the starting value $y^{[0]} = S(h)$. This means that $S(h)$ approximates $z_0 = z(t_0,h)$. The order of accuracy of a general linear method is defined relative to this starting method.

Definition 2.8. *A general linear method* (37) *has* order p *(relative to S), if*

$$y^{[1]} = z(x_1,h) + O(h^{p+1}). \tag{42}$$

A general linear method (37) *has* stage order q *(relative to S), if*

$$\begin{aligned} Y_{1i} &= y(t_0 + c_ih) + O(h^{q+1}), \\ \dot{Y}_{1i} &= \dot{y}(t_0 + c_ih) + O(h^{q+1}). \end{aligned} \tag{43}$$

When deriving general linear methods there are certain design choices to be made. The four main parameter choices for general linear methods are

- p, the overall order of the method,
- q, the stage order of the method,
- k, the number of approximations passed from step to step,
- s, the number of stages.

We make some simplifying assumptions which make the construction of general linear methods more simple. As suggested in [27], we assume that the stage order equals the overall order, i.e. $q = p$, and that $k = p + 1$ as well as $s = p + 1$ for several reasons as explained in [27]. Therefore, we have

$$y^{[n]} = \begin{bmatrix} y(t_n) \\ h\dot{y}(t_n) \\ h^2\ddot{y}(t_n) \\ \vdots \\ h^{p}y^{(p)}(t_n) \end{bmatrix} + O(h^{p+1}), \tag{44}$$

and the stage values satisfy

$$\begin{aligned} Y_{ni} &= y(t_{n-1} + c_i h) + O(h^{p+1}), \\ \dot{Y}_{ni} &= \dot{y}(t_{n-1} + c_i h) + O(h^{p+1}), \quad \text{for } i = 1, 2, \ldots, s. \end{aligned} \tag{45}$$

For deriving order conditions for general linear methods of the form (37) we define two shifting matrices of size $(p+1) \times (p+1)$

$$J = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 1 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & \ldots & 1 & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 & 0 \end{bmatrix}, \quad K = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{46}$$

and a basis vector Z as $Z = \begin{bmatrix} 1 & z & z^2 & \cdots & z^p \end{bmatrix}^T$ for a complex parameter z. Some useful properties of these shifting matrices are that

$$\begin{aligned} J^T Z &= zZ + O(z^{p+2}), \\ K^T Z &= z^2 Z + O(z^{p+2}). \end{aligned} \tag{47}$$

The order conditions are then given by the following theorem.

Theorem 2.9. *A general linear method* (37) *in Nordsieck form has order and stage order p, if and only if*

$$\begin{aligned} \exp(cz) &= z^2 \tilde{\mathcal{A}} \exp(cz) + \tilde{\mathcal{U}} Z + O(z^{p+1}), \\ \frac{z}{h} \exp(cz) &= \frac{z^2}{h} \mathcal{A} \exp(cz) + \mathcal{U} Z + O(z^{p+1}), \\ \exp(z) Z &= z^2 \tilde{\mathcal{B}} \exp(cz) + \tilde{\mathcal{V}} Z + O(z^{p+1}). \end{aligned} \tag{48}$$

Here, the exp *function is applied component-wise to a vector. The matrix C is given by $C = \begin{bmatrix} e & c & \frac{c^2}{2!} & \cdots & \frac{c^p}{p!} \end{bmatrix}$ and the matrix E is given by*

$$E = \begin{bmatrix} 1 & \frac{1}{1!} & \frac{1}{2!} & \cdots & \frac{1}{(p-1)!} & \frac{1}{p!} \\ 0 & 1 & \frac{1}{1!} & \cdots & \frac{1}{(p-2)!} & \frac{1}{(p-1)!} \\ 0 & 0 & 1 & \cdots & \frac{1}{(p-3)!} & \frac{1}{(p-2)!} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & \frac{1}{1!} \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}.$$

Proof. The stage values Y_n, $\dot{Y}_n$, the output approximations $y^{[n]}$ and the scaled second derivatives $h^2 F(Y_n, \dot{Y}_n)$ can be represented in terms of $y^{[n-1]}$ using Taylor series expansions about t_{n-1}. Therefore,

$$\begin{aligned} Y_n &= C y^{[n-1]} + O(h^{p+1}), \\ \dot{Y}_n &= \frac{1}{h} C J^T y^{[n-1]} + O(h^{p+1}), \\ y^{[n]} &= E y^{[n-1]} + O(h^{p+1}), \\ h^2 F(Y_n, \dot{Y}_n) &= C K^T y^{[n-1]} + O(h^{p+1}). \end{aligned} \tag{49}$$

Substituting the above expressions into the method (37) it follows that

$$\begin{aligned} Cy^{[n-1]} &= \tilde{\mathcal{A}}CK^T y^{[n-1]} + \tilde{\mathcal{U}} y^{[n-1]} + O(h^{p+1}), \\ \frac{1}{h} CJ^T y^{[n-1]} &= \frac{1}{h} \mathcal{A}CK^T y^{[n-1]} + \mathcal{U} y^{[n-1]} + O(h^{p+1}), \\ Ey^{[n-1]} &= \tilde{\mathcal{B}}CK^T y^{[n-1]} + \tilde{\mathcal{V}} y^{[n-1]} + O(h^{p+1}). \end{aligned} \tag{50}$$

By making the substitution $h^k y^{(k)}(t_{n-1})$ for z^k, the stage order and order conditions are now

$$\begin{aligned} CZ &= \tilde{\mathcal{A}}CK^T Z + \tilde{\mathcal{U}} Z + O(z^{p+1}), \\ \frac{1}{h} CJ^T Z &= \frac{1}{h} \mathcal{A}CK^T Z + \mathcal{U} Z + O(z^{p+1}), \\ EZ &= \tilde{\mathcal{B}}CK^T Z + \tilde{\mathcal{V}} Z + O(z^{p+1}). \end{aligned} \tag{51}$$

Using the relations (47), the stage order and order conditions now become

$$\begin{aligned} CZ &= z^2 \tilde{\mathcal{A}}CZ + \tilde{\mathcal{U}} Z + O(z^{p+1}), \\ \frac{1}{h} zCZ &= \frac{1}{h} z^2 \mathcal{A}CZ + \mathcal{U} Z + O(z^{p+1}), \\ EZ &= z^2 \tilde{\mathcal{B}}CZ + \tilde{\mathcal{V}} Z + O(z^{p+1}). \end{aligned} \tag{52}$$

The result now follows by substituting

$$\begin{aligned} EZ &= \exp(z) Z + O(z^{p+1}), \\ CZ &= \exp(cz) + O(z^{p+1}) \end{aligned} \tag{53}$$

into the above stage order and order conditions. □

Remark 2.10. *A direct consequence of the proof of Theorem 2.9 is that the matrices $\tilde{\mathcal{U}}$, $\tilde{\mathcal{V}}$ and $\mathcal{U}$ are completely defined by $\tilde{\mathcal{A}}$, $\tilde{\mathcal{B}}$, $\mathcal{A}$ and the vector c, respectively. To see this, equate coefficients of $z^0, z^1, \ldots, z^p$ in* (51) *which leads to*

$$\begin{aligned} \tilde{\mathcal{U}} &= C - \tilde{\mathcal{A}}CK^T, \\ h\mathcal{U} &= CJ^T - \mathcal{A}CK^T, \\ \tilde{\mathcal{V}} &= E - \tilde{\mathcal{B}}CK^T. \end{aligned} \tag{54}$$

Example 2.11. *A possible general linear method of order $p = 2$ satisfying the conditions* (54) *with vector $c = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & 1 \end{bmatrix}$ is given below.*

$$\mathcal{M} = \left[\begin{array}{ccc|ccc} \frac{37}{432} & -\frac{41}{576} & \frac{29}{1728} & 1 & \frac{1}{4} & 0 \\ \frac{23}{108} & -\frac{1}{9} & \frac{5}{216} & 1 & \frac{1}{2} & 0 \\ \frac{13}{27} & -\frac{1}{18} & \frac{2}{27} & 1 & 1 & 0 \\ \hline \frac{4}{9} & -\frac{11}{48} & \frac{5}{144} & 0 & \frac{1}{h} & 0 \\ \frac{5}{9} & -\frac{1}{12} & \frac{1}{36} & 0 & \frac{1}{h} & 0 \\ \frac{4}{9} & \frac{1}{3} & \frac{2}{9} & 0 & \frac{1}{h} & 0 \\ \hline \frac{13}{27} & -\frac{1}{18} & \frac{2}{27} & 1 & 1 & 0 \\ \frac{4}{9} & \frac{1}{3} & \frac{2}{9} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

To start the integration it is necessary to approximate the initial Nordsieck vector at the initial point

$$y^{[0]} = \begin{bmatrix} y(t_0) \\ h\dot{y}(t_0) \\ \vdots \\ h^{p+1}y^{(p+1)}(t_0) \end{bmatrix} + O(h^{p+1}), \tag{55}$$

using a starting method $\mathcal{S}(h)$. Up till now no starting methods have been constructed. The question of finding good starting methods is currently under investigation. Therefore, in numerical examples we will always start the integration with an exact initial Nordsieck vector in this early stage of the work.
Further, for the effective implementation of a numerical method we require stepsize changes. For a general linear method stepsize variation is more difficult than for Runge-Kutta methods, but the Nordsieck representation makes it relatively easy to vary the stepsize. Denote by $h_n = t_n - t_{n-1}$ the stepsize in step n. The general linear method (37) on a non-uniform grid is then given by

$$\begin{aligned} y^{[n]} &= \tilde{\mathcal{V}}\tilde{y}^{[n-1]} + h_n^2\tilde{\mathcal{B}}F(Y_n, \dot{Y}_n), \\ Y_n &= \tilde{\mathcal{U}}\tilde{y}^{[n-1]} + h_n^2\tilde{\mathcal{A}}F(Y_n, \dot{Y}_n), \\ \dot{Y}_n &= \mathcal{U}\tilde{y}^{[n-1]} + h_n\mathcal{A}F(Y_n, \dot{Y}_n), \end{aligned} \tag{56}$$

where the incoming and outgoing Nordsieck vectors are

$$y^{[n]} = \begin{bmatrix} y_n \\ h_n\dot{y}_n \\ h_n^2\ddot{y}_n \\ \vdots \\ h_n^{p+1}y_n^{(p+1)} \end{bmatrix}, \quad \tilde{y}^{[n-1]} = \begin{bmatrix} y_{n-1} \\ h_n\dot{y}_{n-1} \\ h_n^2\ddot{y}_{n-1} \\ \vdots \\ h_n^{p+1}y_{n-1}^{(p+1)} \end{bmatrix}. \tag{57}$$

If we define a rescaling matrix $D(d_n)$ as

$$D(d_n) = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & d_n & 0 & \dots & 0 & 0 \\ 0 & 0 & d_n^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & d_n^{p-1} & 0 \\ 0 & 0 & 0 & \dots & 0 & d_n^{p+1} \end{bmatrix}, \tag{58}$$

where $d_n = \frac{h_n}{h_{n-1}}$, then we have $\tilde{y}^{[n-1]} = D(d_n)y^{[n-1]}$ and we can write the variable stepsize method as

$$\begin{aligned} y^{[n]} &= \tilde{\mathcal{V}}D(d_n)y^{[n-1]} + h_n^2\tilde{\mathcal{B}}F(Y_n, \dot{Y}_n), \\ Y_n &= \tilde{\mathcal{U}}D(d_n)y^{[n-1]} + h_n^2\tilde{\mathcal{A}}F(Y_n, \dot{Y}_n), \\ \dot{Y}_n &= \mathcal{U}D(d_n)y^{[n-1]} + h_n\mathcal{A}F(Y_n, \dot{Y}_n). \end{aligned} \tag{59}$$

Now, we return to the numerical solution of the second order semi-explicit differential-algebraic system (1). If we use method (59) for the numerical solution of (1) and define $x = [y^T, \lambda^T]^T$ and $X_n = [Y_n^T, \Lambda_n^T]^T$ as in (32), we get the following method:

$$x^{[n]} = \tilde{\mathcal{V}}D(d_n)x^{[n-1]} + h_n^2\tilde{\mathcal{B}}\ddot{X}_n, \tag{60a}$$

where $\ddot{X}_n$ is determined from the nonlinear system

$$F(X_n, \dot{X}_n, \ddot{X}_n) = \begin{bmatrix} \ddot{Y}_{n,1} - f(t_{n-1} + c_1 h_n, Y_{n,1}, \dot{Y}_{n,1}, \Lambda_{n,1}) \\ g(t_{n-1} + c_1 h_n, Y_{n,1}) \\ \vdots \\ \ddot{Y}_{n,s} - f(t_{n-1} + c_s h_n, Y_{n,s}, \dot{Y}_{n,s}, \Lambda_{n,s}) \\ g(t_{n-1} + c_s h_n, Y_{n,s}) \end{bmatrix}, \tag{60b}$$

and the internal stages are given by

$$\begin{aligned} X_n &= \tilde{\mathcal{U}} D(d_n) x^{[n-1]} + h_n^2 \tilde{\mathcal{A}} \ddot{X}_n, \\ \dot{X}_n &= \mathcal{U} D(d_n) x^{[n-1]} + h_n \mathcal{A} \ddot{X}_n. \end{aligned} \tag{60c}$$

Solving the nonlinear system (60b) using Newton's method requires the Jacobian of F, which is given by

$$DF(X_n, \dot{X}_n, \ddot{X}_n) = \{F_{\ddot{X}}\} + h_n^2 (\tilde{\mathcal{A}} \otimes I_m)\{F_X\} + h_n (\mathcal{A} \otimes I_m)\{F_{\dot{X}}\}. \tag{61}$$

In the following we describe one step of the integration procedure used for solving the semi-explicit second order DAE (1) with a general linear method.

Algorithm *General linear method for second order systems*

INPUT: $x^{[n-1]}$, t_{n-1}, h_n, h_{n-1},
OUTPUT: $x^{[n]}$,

$t_n = t_{n-1} + h_n$;
Get GLM coefficients $\mathcal{A}$, $\tilde{\mathcal{A}}$, $\mathcal{U}$, $\tilde{\mathcal{U}}$, $\tilde{\mathcal{B}}$, $\tilde{\mathcal{V}}$, c;
Compute $D(d_n)$ as in (58);
$x^{(0)} = [0, \ldots, 0]^T$; {starting value for Newton Iteration}
$k = 0$;
repeat {Newton Iteration}
  $\ddot{X} = x^{(k)}$;
  $X = (\tilde{U} D(d_n) \otimes I_m) x^{[n-1]} + h_n^2 (\tilde{\mathcal{A}} \otimes I_m) \ddot{X}$;
  $\dot{X} = (U D(d_n) \otimes I_m) x^{[n-1]} + h_n (\mathcal{A} \otimes I_m) \ddot{X}$;
  Compute $F(X, \dot{X}, \ddot{X})$ as in (60b);
  Compute $DF(X, \dot{X}, \ddot{X})$ as in (61);
  $[L, U, P] = lu(DF)$;
  $f = -P \cdot F$;
  $z = L \backslash f$;
  $\Delta x^{(k+1)} = U \backslash z$;
  $x^{(k+1)} = x^{(k)} + \Delta x^{(k+1)}$;
  $k = k + 1$;
until converged
$x^{[n]} = (\tilde{\mathcal{V}} D(d_n) \otimes I_m) x^{[n-1]} + h_n^2 (\tilde{\mathcal{B}} \otimes I_m) x^{(k)}$;

In [3] order conditions for general linear methods applied to first order ordinary differential equations are investigated. Furthermore, general linear methods for linear DAEs are investigated in [23] and for nonlinear DAEs that arise in circuit simulation in [26].

Remark 2.12. *Convergence results for general linear methods applied to second order differential-algebraic equations are currently under investigation.*

3 Numerical Example

We consider the following example taken from [25]. The equations of motion describe a mechanical system consisting of just one body of mass 1 in a two-dimensional space, which is moving on the unit circle.

$$\begin{aligned} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} &= \begin{bmatrix} -y_1 - 2y_1\dot{y}_1\dot{y}_2 \\ -\dot{y}_1 + 2y_1 y_2^2 \end{bmatrix} - \begin{bmatrix} 2y_1 \\ 2y_2 \end{bmatrix} \lambda, \\ 0 &= y_1^2 + y_2^2 - 1, \end{aligned} \tag{62}$$

with the initial values

$$t_0 = 0, \quad y(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \dot{y}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \lambda(0) = 0. \tag{63}$$

This is a semi-explicit differential-algebraic system of p-index 3. Reducing the order by introducing new variables for the derivatives yields the corresponding first order form

$$\begin{aligned} \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} &= \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} &= \begin{bmatrix} -y_1 - 2y_1 v_1 v_2 \\ -v_1 + 2y_1 y_2^2 \end{bmatrix} - \begin{bmatrix} 2y_1 \\ 2y_2 \end{bmatrix} \lambda, \\ 0 &= y_1^2 + y_2^2 - 1. \end{aligned} \tag{64}$$

In the following the second order system (62) is solved by the described modified BDF method (15), by the Runge-Kutta method (30) with $s = 2$ and Radau-IIA coefficients and by the general linear method (60) with coefficients given in Example 2.11. For comparison the first order system (64) is solved by ordinary BDF methods, by the ordinary Radau-IIA method of order 3 [14] and by the general linear method

$$\mathcal{M} = \left[\begin{array}{ccc|ccc} \frac{1}{4} & 0 & 0 & 1 & 0 & -\frac{1}{32} \\ \frac{1}{6} & \frac{1}{4} & 0 & 1 & \frac{1}{12} & -\frac{1}{24} \\ \frac{1}{6} & \frac{1}{2} & \frac{1}{4} & 1 & \frac{1}{12} & -\frac{1}{24} \\ \hline \frac{1}{6} & \frac{1}{2} & \frac{1}{4} & 1 & \frac{1}{12} & -\frac{1}{24} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 & 0 & 0 \end{array}\right] \tag{65}$$

with $c = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & 1 \end{bmatrix}$ given in [27]. For error estimation we use the relative and absolute error tolerances $RTOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$ and $ATOL = [10^{-4}, 10^{-4}, 10^{-2}, 10^{-4}, 10^{-4}]$ for the variables $[y_1, y_2, \lambda, \dot{y}_1, \dot{y}_2]$ and $[y_1, y_2, \lambda, v_1, v_2]$, respectively.

Figure 2 displays the results obtained by solving the second and first order systems (62) and (64) with the BDF methods for second and first order systems, respectively. We have used constant stepsize $h = 0.001$ and varying order, i.e., the order is increased step by step starting with order $k = 1$ up to order $k = 4$. The integration is performed on the interval $I = [t_0, t_0{+}0.1]$ using the consistent initial values (63). The results displayed are the numerical solutions for the variables $[y_1, y_2, \lambda]$ together with the absolute errors. We can observe that the modified BDF methods for the second order system yield better results. For the position coordinates y_1 and y_2 both methods yield similar results, but the first order formulas fail for the algebraic variable λ, which is responsible for the higher index of the system. We observe a highly oscillating error when we change the order of the method. When the order is kept

constant (as in the first three steps for starting the integration and in the later steps, when the maximal order is reached) the absolute error for λ is much lower. The high error in the first step in the solution of the first order system is due to the changing of the stepsize from 0 to $h = 0.001$. Better results are obtained regarding the BDF methods for the second order system. We can see that there are no problems in changing the order of the modified BDF method.
Figure 3 displays the results obtained by solving the second order system (62) and the first order system (64) with variable stepsizes and constant order $k = 2$ (in the first step the first order formulas are used, from the second step on the order is kept constant). The integration is again performed on the interval $I = [t_0, t_0 + 0.1]$ using the consistent initial values (63). Again, we can observe that the modified BDF methods for the second order system yield better results. Especially for the algebraic variable λ the formulas for the first order system fail due to the changing of the stepsize as the high errors indicate.
Altogether, the modified BDF methods for second order systems yield better results with respect to changes in the stepsize and in the order, which are important for efficient numerical methods. The high errors in the algebraic variable, which occur due to the higher index in the first order case, do not occur for the second order methods.
Next, we have solved the second order system (62) with a 2-stage Runge-Kutta method (29) of order 3 with Radau-IIA coefficients and the first order system (64) is solved with the ordinary Radau-IIA method of order 3. The integration is performed on the interval $I = [t_0, t_0 + 0.1]$ using the consistent initial values (63). Figure 4 displays the corresponding results for the first and second order system using constant stepsize $h = 0.001$. We can observe that the method for the second order system yields better results. The achieved accuracy is much higher than for the first order case, although both methods are of the same order p. In Figure 5 the corresponding results are given for the case of variable stepsizes. Here, we can observe that both methods yield almost the same results and no significant differences can be observed. The Runge-Kutta method for the second order system works as well as the method for the first order system. A real improvement using the second order system is possible in the case when the right-hand side of the differential equation depends only on t and y, i.e., if we have a differential equations of the form $\ddot{y} = f(t, y)$. Then, the Runge-Kutta methods for the second order system are superior to the methods for first order systems with regard to precision and function evaluations, see [12].
In general, we can conclude that if we consider higher index DAEs, then implicit Runge-Kutta methods are advantageous for solving these problems in comparison to BDF methods, because they can start at a higher order. BDF codes like DASSL are designed to start the integration with a first order method (i.e. implicit Euler method). If the index is three or higher, this method does not converge on the first step, especially for the algebraic variables. Carefully chosen higher order implicit Runge-Kutta methods have no difficulty in determining an accurate numerical solution in all the variables, even on the first step, see [2].
Finally, we have solved the second order system (62) with a general linear method (60) of order $p = 2$ with coefficients given in Example 2.11. The first order system (64) is solved with the general linear method (65). In both cases the initial Nordsieck vectors are given exactly. Again, the integration is performed on the interval $I = [t_0, t_0 + 0.1]$ with consistent initial values (63) once with constant stepsize $h = 0.001$ and once with variable stepsize. The results are given in Figure 6 and 7. In both cases, that is for constant and variable stepsizes, the method for the second order system yields better results. Especially, when we use constant stepsize the second order formula works better regarding the accuracy of the algebraic variable.

Our second example is the simple RLC electrical circuit [5] in Figure 1. We have a voltage source $v_s(t)$ and R, L, C are the resistance, inductance and capacitance, respectively. The corresponding voltage drops are denoted by $v_R(t)$, $v_L(t)$ and $v_C(t)$, respectively, and $I(t)$ denotes the current. Applying Kirchhoff's laws we obtain the following circuit equations in

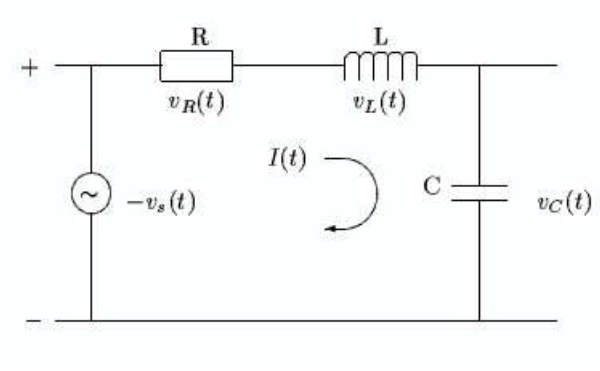

Figure 1: A simple RLC circuit

the standard first order form

$$\begin{bmatrix} L & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{I}(t) \\ \dot{v}_L(t) \\ \dot{v}_C(t) \\ \dot{v}_R(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1/C & 0 & 0 & 0 \\ -R & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} I(t) \\ v_L(t) \\ v_C(t) \\ v_R(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} v_S(t). \tag{66}$$

Using the derivative of the second equation $\dot{I}(t) = C\ddot{v}_c(t)$ this system can be written in an equivalent second order form

$$\begin{bmatrix} 0 & LC & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{v}_L(t) \\ \ddot{v}_C(t) \\ \ddot{v}_R(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -RC & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{v}_L(t) \\ \dot{v}_C(t) \\ \dot{v}_R(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_L(t) \\ v_C(t) \\ v_R(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} v_S(t). \tag{67}$$

The circuit equations (67) have been solved using the BDF method (15) of order 2, the Runge-Kutta method (30) with $s = 2$ and Radau-IIA coefficients and the general linear method (60) with coefficients given in Example 2.11. The integration is performed on the interval $I = [0, 7]$ using the consistent initial values $[v_L(0), v_C(0), v_R(0)] = [0.0627, 1, -1.0627]$ and $[\dot{v}_L(0), \dot{v}_C(0), \dot{v}_R(0)] = [-0.0222, -0.3542, 0.3765]$, $v_S(t) \equiv 0$ and constant stepsize $h = 0.01$. Further, we use the error tolerances $RTOL = ATOL = [10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$ for the variables $[v_L, v_C, v_R, \dot{v}_L, \dot{v}_C, \dot{v}_R]$. The results are given in Figure 8. We can observe that the general linear method yields the best results and that the BDF method yields the highest errors. The accuracy of the Runge-Kutta method is close to that of the general linear method. Although we have used constant order and constant stepsize the BDF method cannot reach the same accuracy as the Runge-Kutta method or the general linear method, but all methods yield satisfactory results for the second order differential-algebraic system.

To sum up, we can say that it is possible and advantageous to solve second order differential-algebraic systems directly. In the case of the BDF methods the results are better as for the corresponding first order system, when changes in the stepsize and in the order of the methods are taken. Thus, by numerically solving the second order system directly we can avoid certain numerical difficulties as an increasing index or the failure of the numerical method as described before and therefore we can increase the efficiency of the numerical methods and obtain better numerical results.

4 Conclusion

In the numerical solution of second order semi-explicit differential-algebraic equations the direct solution of the second order system is preferred to the transformation into a corresponding first order system and numerically solving this first order system. When transforming a second order DAE into a first order system certain difficulties can occur as an increase of the index, the destruction of structures and symmetries, an increase of the dimension and ill-conditioning. These problems can be circumvented using the direct solution of the second order system. We have presented BDF methods, Runge-Kutta methods and general linear methods for the numerical solution of second order differential-algebraic systems in semi-explicit form. Certain numerical examples illustrate that we can obtain better numerical results solving second order systems directly compared to the solution of corresponding first order systems, particularly with regard to changes in the stepsize or in the order of BDF methods. In general, Runge-Kutta methods and general linear methods seem to be advantageous to BDF methods as numerical examples illustrate.

References

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

[2] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential Algebraic Equations*, volume 14 of *Classics in Applied Mathematics*. SIAM, Philadelphia, PA, 1996.

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

[4] S.L. Campbell and C.W. Gear. The index of general nonlinear DAEs. *Numerische Mathematik*, 72(2):173–196, 1995.

[5] L. Dai. *Singular Control Systems*. volume 118 of Lecture Notes in Control and Information Sciences. Springer Verlag, Berlin, 1989.

[6] C. De Boor. *Splinefunktionen*. Birkhäuser, Basel, 1990.

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

[8] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1971.

[9] E. Griepentrog and R. März. *Differential-Algebraic Equations and Their Numerical Treatment*, volume 88 of *Teubner-Texte zur Mathematik*. BSB B.G.Teubner Verlagsgesellschaft, Leipzig, 1986.

[10] M. Günther and U. Feldmann. CAD-based electric-circuit modeling in industry, I. Mathematical structure and index of network equations. *Surveys on Mathematics for Industry*, 8:97–129, 1999.

[11] M. Günther and U. Feldmann. CAD-based electric-circuit modeling in industry, II. Impact of circuit configuration and parameters. *Surveys on Mathematics for Industry*, 8:131–157, 1999.

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

[13] E. Hairer and G. Wanner. A theory for Nyström methods. *Numerische Mathematik*, 25:383–400, 1976.

[14] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*. Springer, Berlin Heidelberg, 2nd edition, 1996.

[15] M. Hebsaker. Conditions for the coefficients of Runge-Kutta methods for systems of nth order differential equations. *J. Comput. Appl. Math.*, 11:287–303, 1973.

[16] M. Hebsaker. *Neue Runge-Kutta-Fehlberg-Verfahren für Differentialgleichungs-Systeme n-ter Ordnung*. PhD thesis, Siegen, 1980.

[17] P. Henrici. *Discrete variable Methods in ordinary differential equations*. Wiley, 1962.

[18] P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations. Analysis and Numerical Solution*. EMS Publishing House, Zürich, Switzerland, 2006. To appear.

[19] R. März. The index of linear differential algebraic equations with properly stated leading terms. *Results in Mathematics*, 42(3-4):308–338, 2002.

[20] The MathWorks. MATLAB, version 7.0.4, 2005.

[21] V. Mehrmann and C. Shi. Analysis of higher order linear differential-algebraic system. Technical Report Preprint 17-2004, Institut für Mathematik, Technische Universität Berlin, 2004.

[22] J. Sand. On implicit Euler and related methods for high-order high-index DAEs. *Applied Numerical Mathematics*, 42:411–424, 2002.

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

[24] C. Shi. *Linear differential-algebraic equations of higher-order and the regularity or singularity of matrix polynomials*. PhD thesis, Institut für Mathematik, Technische Universität Berlin, 2004.

[25] A. Steinbrecher. Regularization of nonlinear equations of motion of multibody systems by index reduction with preserving the solution manifold. Technical Report 742-02, Institut für Mathematik, Technische Universität Berlin, 2002.

[26] S. Voigtmann. General linear methods for nonlinear DAEs in circuit simulation. Technical Report Preprint 20, Institut für Mathematik, Humboldt Universität Berlin, 2004.

[27] W. Wright. *General linear methods with inherent Runge-Kutta stability*. PhD thesis, University of Auckland, 2002.

[28] R. Zurmühl. Runge-Kutta-Verfahren zur numerischen Integration von Differentialgleichungen n-ter Ordnung. *ZAMM*, 28:173–182, 1948.

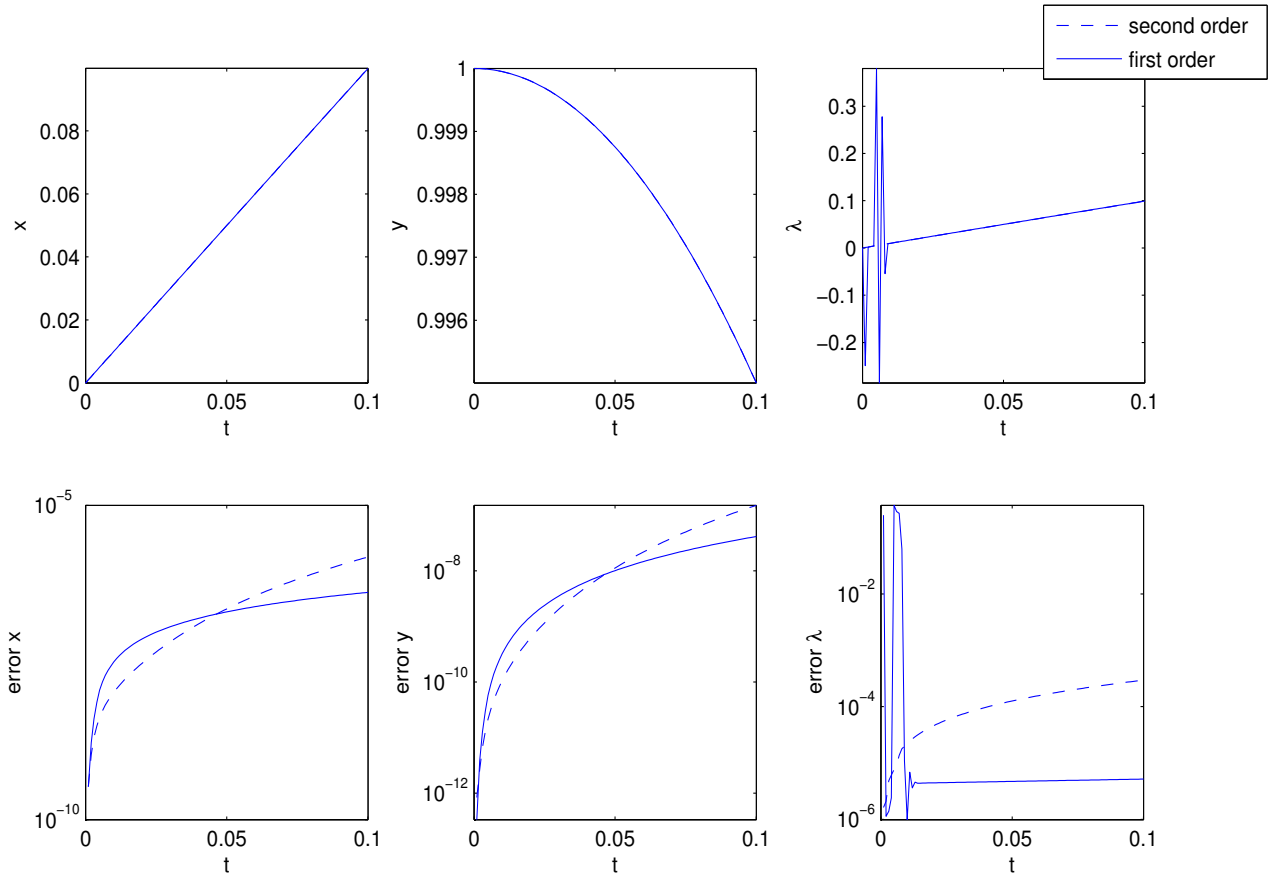

Figure 2: Numerical solutions obtained by solving system (62) and (64) with ordinary and modified BDF methods using constant stepsize $h = 0.001$ and varying order, i.e., starting with order $k = 1$ the order is increased step by step up to order $k = 4$ and is then kept constant.

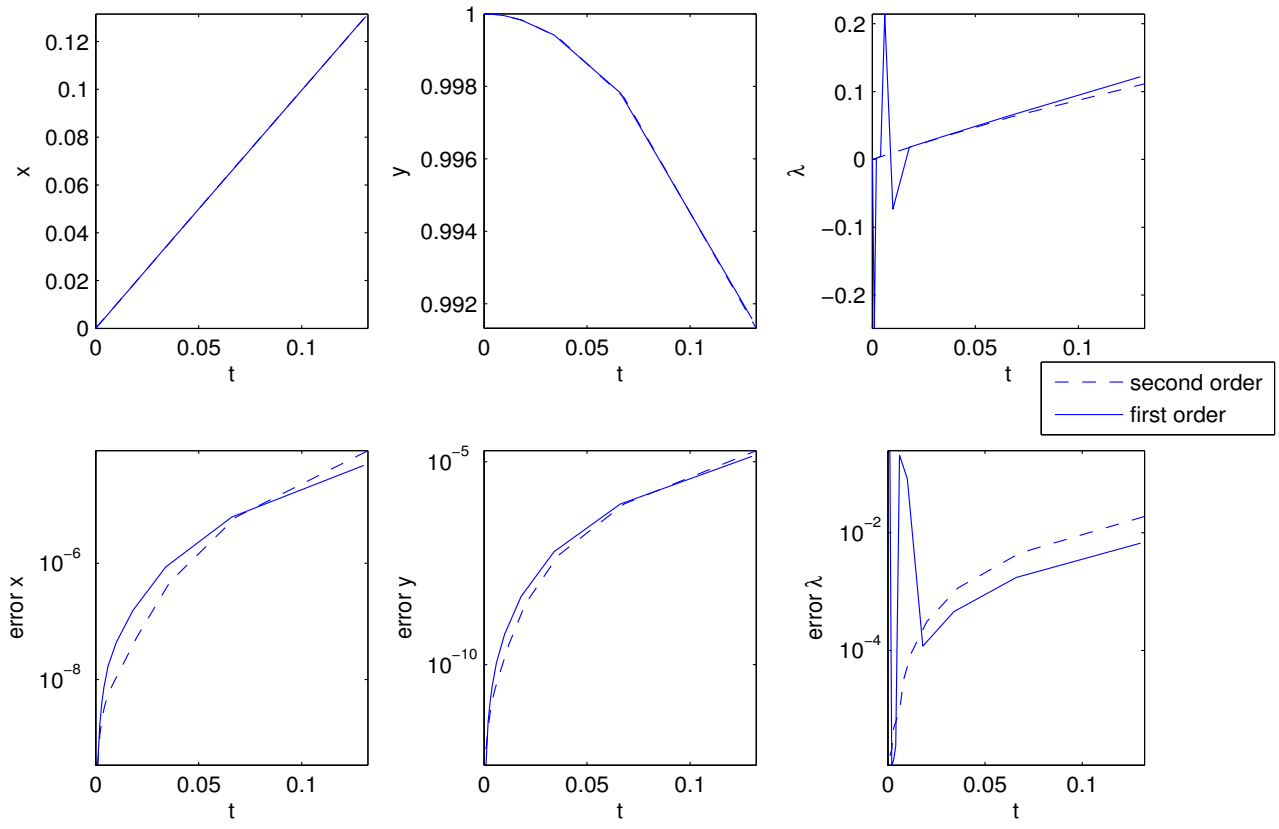

Figure 3: Numerical solutions obtained by solving system (62) and (64) with ordinary and modified BDF methods using variable stepsizes and constant order $k = 2$.

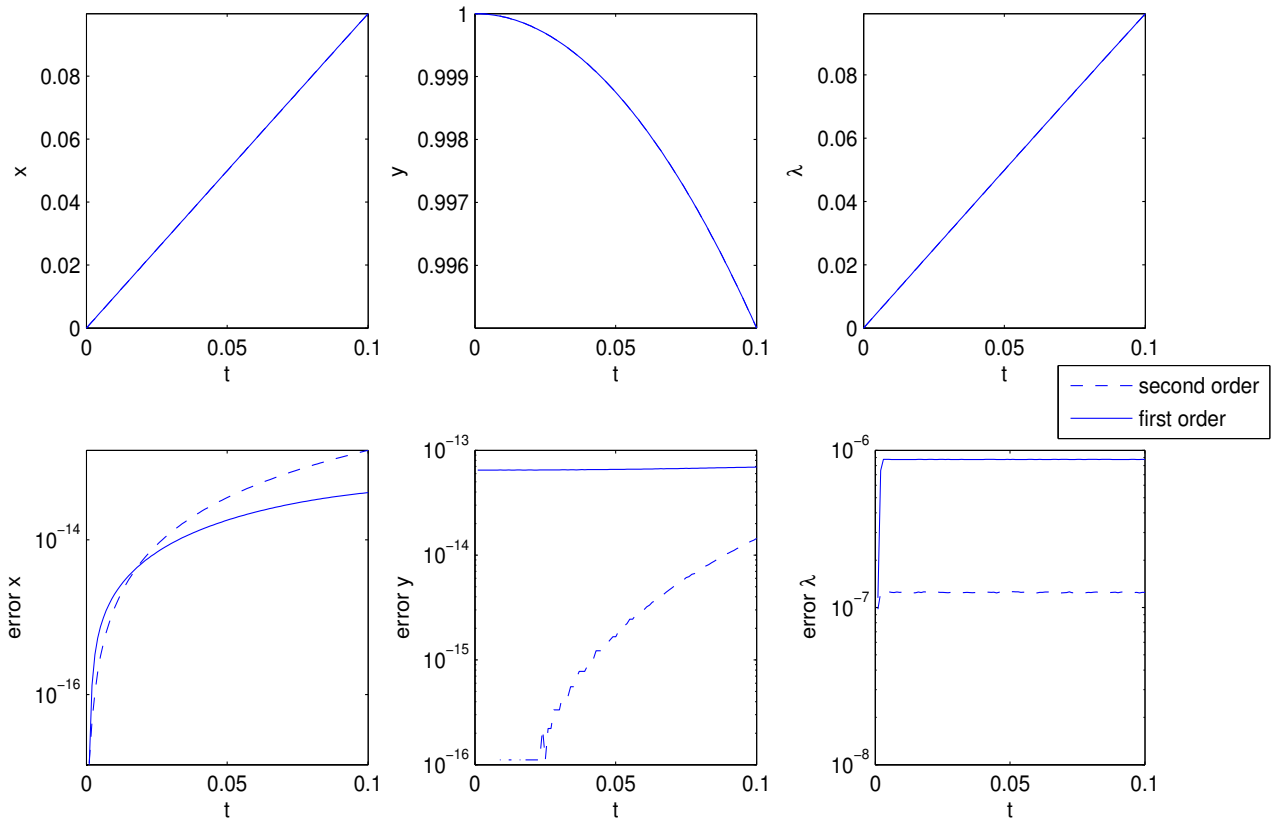

Figure 4: Numerical solutions obtained by solving system (62) and (64) with 2-stage Runge-Kutta methods using constant stepsize $h = 0.001$.

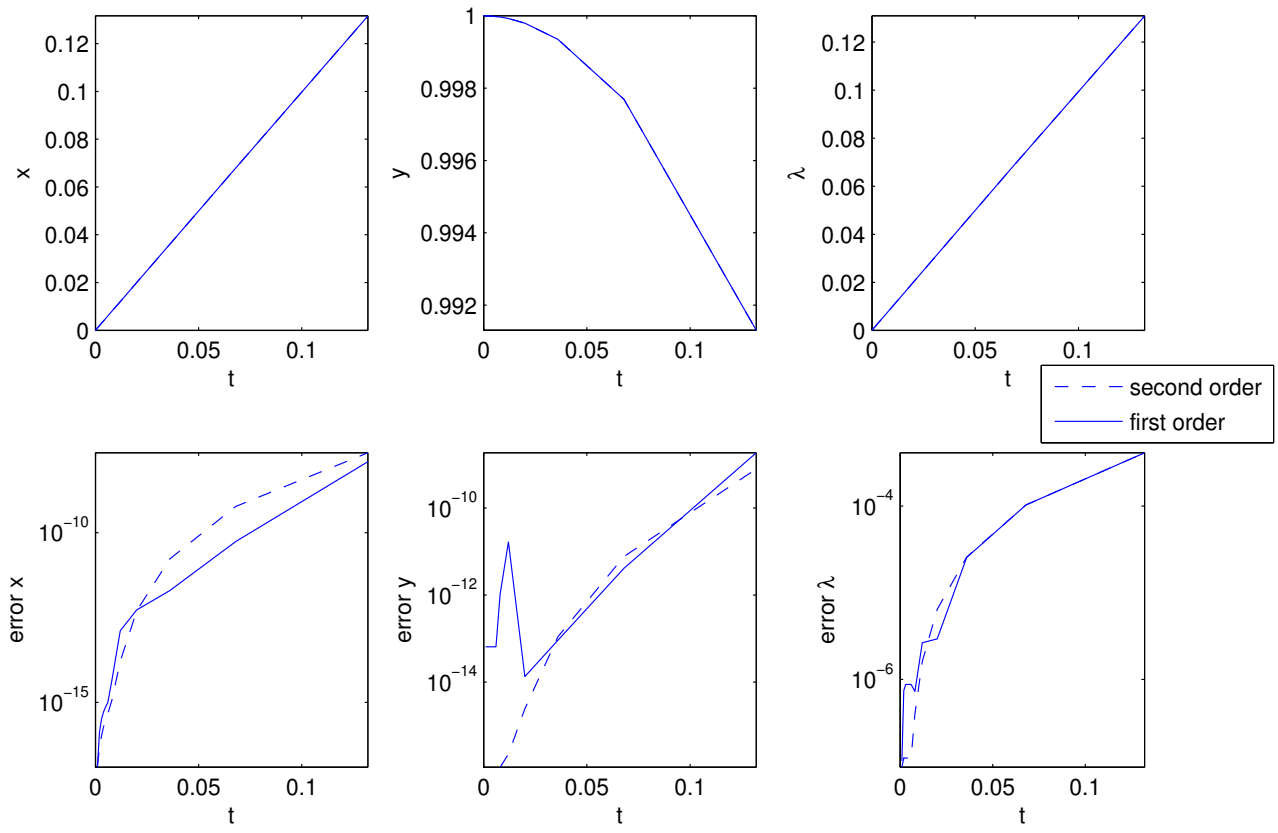

Figure 5: Numerical solutions obtained by solving system (62) and (64) with 2-stage Runge-Kutta methods using variable stepsizes.

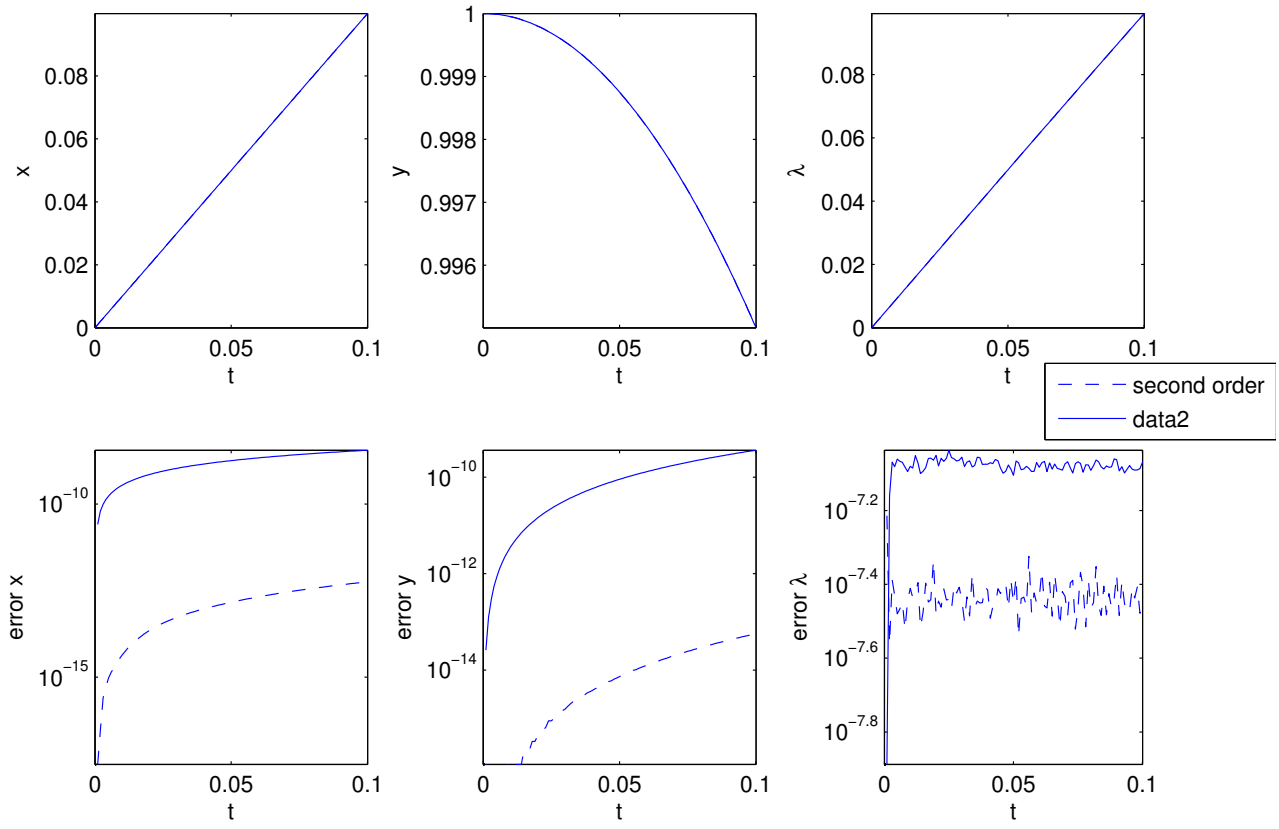

Figure 6: Numerical solutions obtained by solving system (62) and (64) with a general linear method using constant stepsize $h = 0.001$.

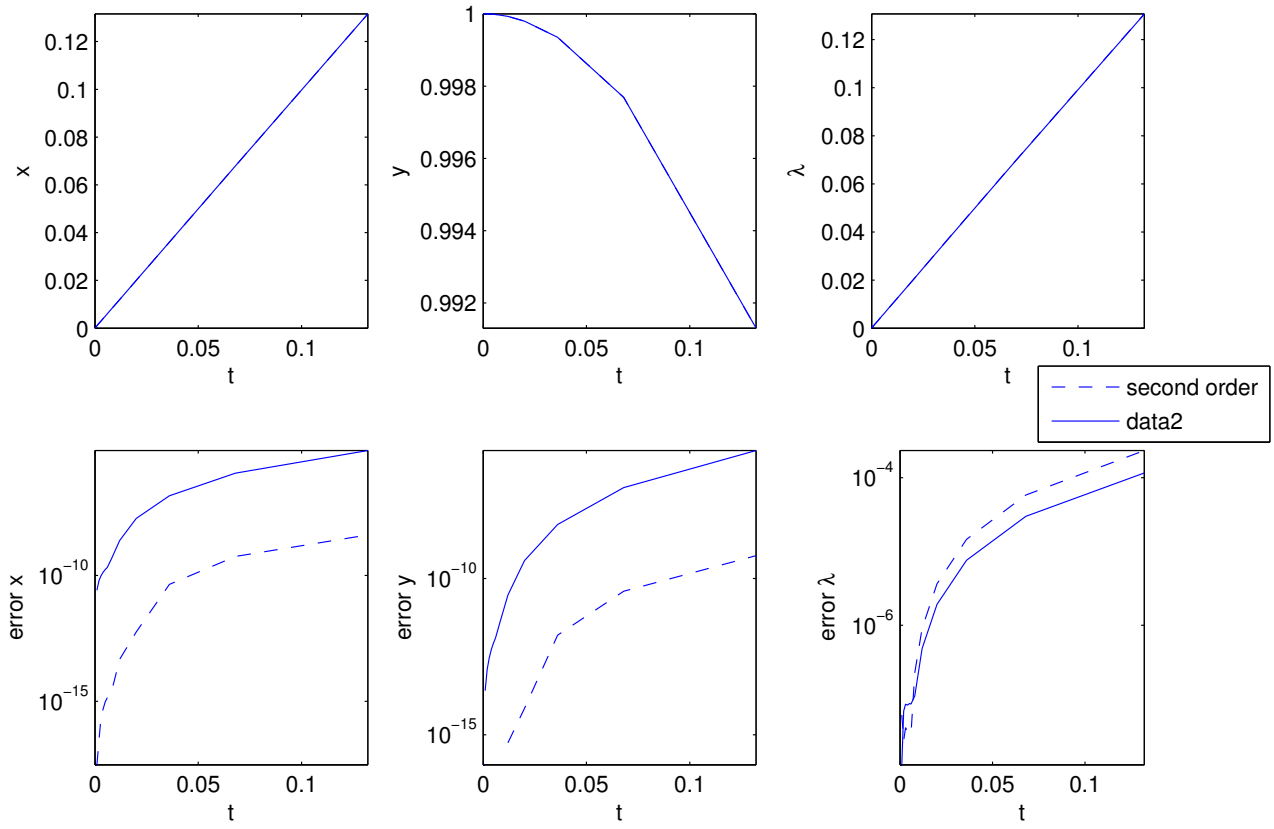

Figure 7: Numerical solutions obtained by solving system (62) and (64) with a general linear method using variable stepsizes.

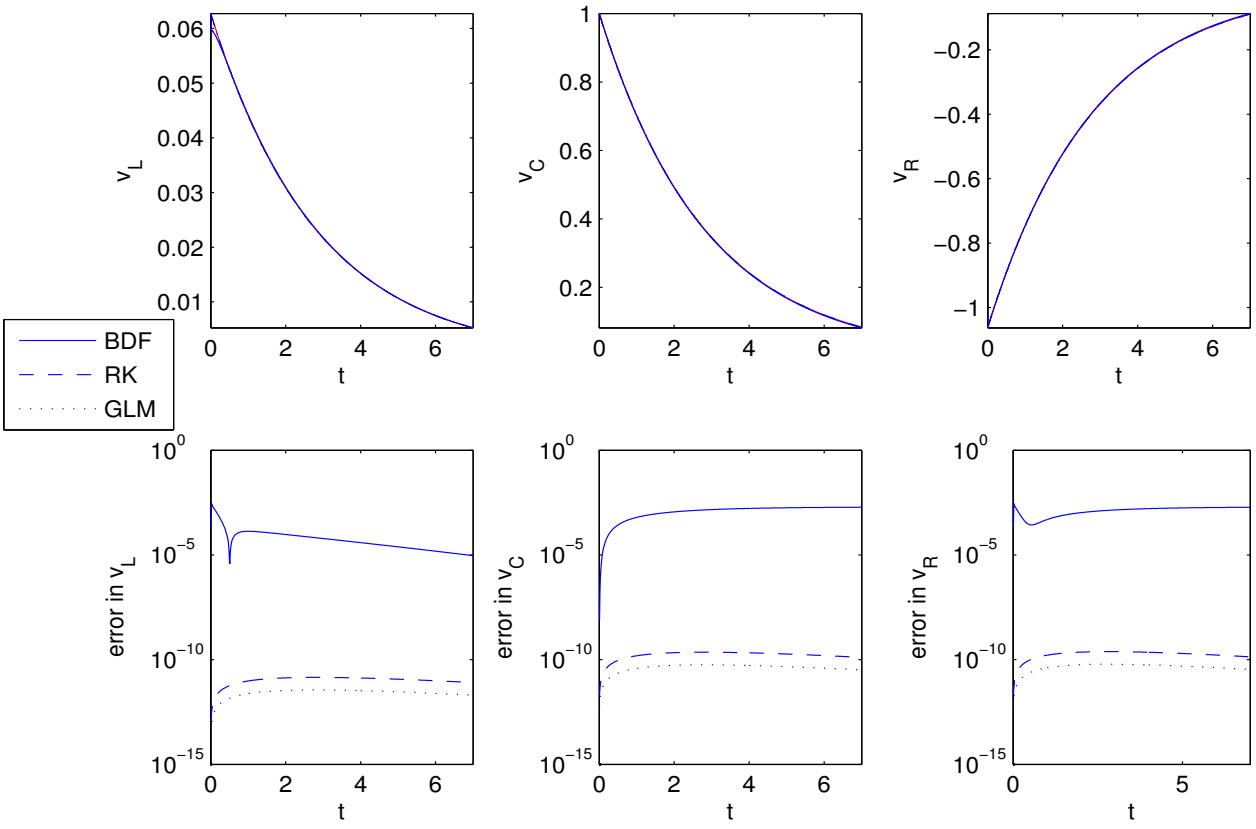

Figure 8: Numerical solution of the circuit equations (67) using BDF methods, a Runge-Kutta method and a general linear method with constant stepsize $h = 0.01$.