Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

A Novel Class of Automatic Differentiation Based Explicit Integrators for Stiff Differential Equations

Boris Houska and Moritz Diehl

Received: date / Accepted: date

Abstract In this paper we propose a novel numerical method which is suitable for the integration of nonlinear and stiff ordinary differential equations (ODE). Existing integrators for stiff ODEs are typically implicit, i.e., they proceed by solving a nonlinear equation system in each step of the method. For the case that this nonlinear system is solved by an exact Newton method, we need to compute and decompose the full Jacobian of the right-hand side function in each step. The contribution of this paper is the introduction of a nonlinear integration scheme which is explicit and for which various desirable stability properties can be proven. Each integrator step of this explicit method requires only a fixed number of directional derivatives of the right-hand side function which can efficiently be computed by automatic differentiation. We illustrate the performance and implementation advantages of the method by testing it with numerical examples.

Keywords Differential Equations · Explicit Integrators · Stability Analysis · Automatic Differentiation

Mathematics Subject Classification (2000) 65L04 · 65L06 · 65L20

Boris Houska
Department of Chemical Engineering & Centre for Process Systems Engineering,
Imperial College London, South Kensington Campus, SW7 2AZ, London, UK.
E-mail: b.houska@imperial.ac.uk

Moritz Diehl
Electrical Engineering Department & Optimization in Engineering Center,
KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium.
E-mail: moritz.diehl@esat.kuleuven.be

1 Introduction

In this paper, we are interested in numerical methods for solving nonlinear ordinary differential equations of the form

$$\forall t \in \mathbb{R}_+ : \quad \dot{y}(t) = f(y(t)) \quad \text{with} \quad y(0) = y_0 \; .$$

Here, $f : \mathbb{R}^n \to \mathbb{R}^n$ is assumed to be a sufficiently often differentiable right-hand side function and $y_0 \in \mathbb{R}^n$ a given initial value. For simplicity of notation, we regard only autonomous differential equations.

A basic class of integration algorithms are the one-step methods. Starting with $x = y_0$ and $t = 0$ a numerical one-step integration method proceeds by choosing at time t a step size $h \in \mathbb{R}_+$ and propagating the state approximation

$$x^+ = \Phi(h, f, x) \; . \tag{1}$$

Here, $\Phi(h, f, \cdot) : \mathbb{R}^n \to \mathbb{R}^n$ maps the state approximation $x \approx y(t)$ at time t to the state approximation $x^+ \approx y(t^+)$ at time $t^+ := t + h$. After the step has been performed we set $x \to x^+$ and $t \to t^+$ in order to proceed with the next step. Note that the method Φ can be regarded as a functional in f.

1.1 A Review of Existing Numerical Integration Methods

The numerical integration of nonlinear differential equations is important for many research areas. Whenever we deal with dynamic processes the question of how to simulate them in a computer arises. There exists a huge amount of literature on numerical integration methods and a similarly huge amount of software implementations. In this paper, we cannot give a complete overview about all of them reviewing only some selected articles.

Most of the numerical methods for differential equations are in one or the other way inspired by Euler's original work [20]. For example, based on Euler's ideas, Adams and Bashforth have developed the first version of a linear multistep method [5]. Extensions of these multistep methods can for example be found in [36] and [37]. In this context, we also highlight the work of Curtiss and Hirschfelder who developed the backward differentiation formulae [15] - also known under the name BDF methods. Another type of explicit and implicit numerical integration methods has been developed by Runge and Kutta [34, 40]. Note that Runge-Kutta methods have been analyzed in all their details and by many authors. We refer to [2, 18, 25, 27] for a modern overview including a discussion of step-size control heuristics.

It is a well-known fact that for some differential equations numerical integrators, e.g., explicit Runge-Kutta methods, have to take excessively small steps, although the exact solution is perfectly smooth [27]. In this case, the differential equation is called stiff. The prototype for such an equation is the scalar ODE

$$\dot{y}(t) \; = \; \alpha \, y(t)$$

which converges to 0 for $t \to \infty$ as long as α is a complex number with negative real part. If the discrete iterates of the numerical method inherit this behavior, it is called A-stable. Unfortunately, most of the explicit methods are not A-stable. For example, all explicit linear methods are not A-stable [27]. This implies in particular that neither explicit Runge-Kutta nor explicit linear multistep methods are A-stable which is one of the main motivations for the implementation of implicit methods. Here, a step of an implicit method is typically more expensive than the step of an explicit integrator, as a nonlinear equation has to be solved. However, in terms of overall computation time, it is for most of the existing integration schemes worthwhile to invest into solving a nonlinear equation in each step whenever this allows the integrator to choose significantly larger step sizes.

Note that most of the theory on stability and convergence for numerical integrators has been initiated by the work of Dahlquist [16,17] who was pioneering A- and B-stability criteria. For example, one of his famous results is that implicit multistep methods can only be A-stable if their consistency order is at most two. Later Butcher analyzed the B-stability properties of implicit Runge-Kutta methods [11]. This work has been generalized in [10] where also the concepts of AN-, BN-, and algebraic stability have been introduced. For an overview of these developments we refer to [27].

Besides the standard Runge-Kutta and linear multistep integrators there are many other variants. One variant, which is of interest in the context of this paper, are the Taylor series integration methods [3,12,35,39]. For example, the explicit second order Taylor series method

$$x^+ = x + hf(x) + \frac{1}{2}h^2 f'(x)f(x)$$

can be implemented by evaluating the directional derivative $f'(x)f(x)$ with automatic differentiation. For an extensive overview of such automatic differentiation based techniques we refer to [24]. Note that there exist also implicit variants of the Taylor series integration methods [4,13] for which A-stability can be proven. In this context, we highlight an idea of Griewank [23] who suggested an implementation of an implicit variant of the Hermite-Obreshkov-Padé method featuring modern automatic differentiation tools. In this context, Griewank suggests to exploit the fact that the Taylor coefficients of the state vector are closely related to the Jacobian of the right-hand side function which is needed to solve the implicit equations. This is an important idea which will in a similar version be employed in this paper, too.

As it was mentioned above, there are many simulation codes around – and again, we will only review a small selection. One famous historical and efficient code has been published under the name `RADAU5`. This Runge-Kutta based code can be found in [26]. Concerning early implementations of multistep methods, we refer to [21,38]. Finally, it is interesting to note that many famous modern codes are based on BDF methods, e.g., the code `DAESOL` which has been developed in [1,6]. Moreover, we highlight the integrators which are implemented in the package `SUNDIALS` for which we refer to [29,30].

1.2 A Review of Existing Explicit Methods for Stiff ODEs

As it was explained above, existing integrators with desirable stability properties are typically implicit, i.e., we have to solve a nonlinear equation in each step. If we use a Newton type method to solve this nonlinear equation, we encounter at least three implementation problems which are not present in explicit methods: first, it is in general difficult to guarantee that the Newton type iterations converge. Second, we need to compute the Jacobian of the right-hand side function - or at least a suitable approximation. And third, for the Newton type iterations we typically have to invert an $(n \times n)$-matrix, i.e., if no sparsity is exploited, the computational cost is $\mathbf{O}(n^3)$.

Are there numerical methods which are both explicit and applicable for stiff differential equations? As it was discussed above, explicit linear methods, such as explicit Runge-Kutta methods, cannot be A-stable. Consequently, if we want to construct explicit methods for stiff equations we have to look for a nonlinear method. In the literature, there exist many suggestion for explicit methods which all work well for a limited class of stiff differential equations. However, most of them do not perform well in general. In order to start our review of such explicit methods, we start with the work of Lambert [32] whose easiest method can for scalar differential equations be written as

$$x^+ \;=\; x + \frac{hxf(x)}{x - hf(x)}\,. \tag{2}$$

Lambert suggests several higher order variants of this method and also transfers it to vector valued differential equations in a componentwise way. In fact, it is easy to check that the method (2) is A-stable. Unfortunately, it turns out that Lambert's method is not so useful for more general classes of stiff ODEs. In order to illustrate this, we apply the method (2) to a scalar test equation of the form $\dot{y}(t) = \alpha y(t) + \beta$ with $\alpha < 0$ which yields

$$x^+ \;=\; x + \frac{hx(\alpha x + \beta)}{x - h(\alpha x + \beta)} \;=\; \frac{x^2}{(1 - h\alpha)x - h\beta}\,.$$

For $\beta = 0$ this simplifies to $x^+ = \frac{x}{1-h\alpha}$ which proves A-stability. However, for $\beta \neq 0$ the steady state is at $\bar{x} = -\frac{\beta}{\alpha}$ and we have the limit behavior

$$\lim_{h\to\infty} x^+ \;=\; \begin{cases} \bar{x} & \text{if } \; x = \bar{x}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Thus, the method is for large h not stable at $\bar{x} = -\frac{\beta}{\alpha} \neq 0$. This renders the method useless for most practical applications. Note that Lambert suggested several higher order variants of his nonlinear A-stable methods [32] which suffer unfortunately from similar difficulties.

Another class of explicit methods that has been developed for stiff systems are the exponential time differencing methods for which we refer to [14] and the references therein. For scalar differential equations these methods assume

that the right-hand side can be written as $f(y) = \alpha y + g(y)$ where α has a negative real part. The exact solution for the corresponding differential equation satisfies

$$y(t+h) = e^{\alpha h} y(t) + e^{\alpha h} \int_0^h e^{-\alpha \tau} g(y(t+\tau)) \, \mathrm{d}\tau \; .$$

Now, the idea is to approximate the integral, e.g., by Taylor expansion. In the easiest case, we obtain an explicit method of the form

$$x^+ = e^{\alpha h} \left(x + h g(x) \right)$$

which is also known under the name Integrating Factor Euler method [8]. For vector valued differential equations α is a matrix. For some PDE applications [14], it is possible to compute (or approximate) the matrix $e^{\alpha h}$ once and use the exponential time differencing method. However, as for Lambert's method, the exponential differencing methods work well for a limited class of stiff problems, but not in general.

Note that there exist also other explicit methods which are not A-stable but suitable for some stiff differential equations. For example, Gear and Kevrekidis [22] developed the projective methods where the steps of an inner and an outer integrator are combined. These methods work well for problems in which the eigenvalues of the Jacobian are separated into two clusters, one containing the stiff and one containing the slow components. A similar approach has been developed in [19] where small and large explicit time-steps are combined in order to obtain a more stable iteration. In this context, we also refer to the work of Verwer [43] who analyzes the possibility to use the coefficients in the Butcher tableau of an explicit Runge-Kutta method to not only obtain consistency but also a stability region which is as large as possible. Such stabilized Runge-Kutta methods cannot be A-stable, but they work acceptably for some moderately stiff PDE systems as demonstrated in [43].

1.3 Contributions and Overview

The contribution of this paper is the development of a novel class of explicit one-step integration methods for which non-trivial stability properties can be proven. Here, we propose an algorithm that requires us to compute a few directional derivatives of the right hand side function f in each step, but not the full Jacobian. Moreover, there are no nonlinear equations solved in the integrator steps, i.e., for a given step-size sequence the run-time and memory requirements are known in advance. Finally, the computational cost of each step is of order $\mathbf{O}(n)$, i.e., there is in particular no need to invert any $(n \times n)$-matrix. As we shall discuss, the proposed algorithm overcomes some of the above mentioned disadvantages of both the existing linear implicit methods as well as the existing nonlinear explicit methods.

In Section 2 we start with a motivation and definition of a novel class of Taylor expansion based explicit integration methods. Here, we first outline

a simple prototype integrator of order 1. This prototype integrator is then generalized in Section 3 where we construct higher order methods. In Section 4 we analyze the stability properties of the proposed integration scheme for linear test equations. This stability analysis is the core contribution of this paper (cf. Theorem 4). In Section 5 we discuss a general construction principle for low-rank Jacobian approximations which are needed for the stabilization of our explicit method. Section 6 summarizes the implementation details. In Section 7 we demonstrate that the proposed explicit integrator can be used to simulate large stiff differential equations without solving any nonlinear equations or exploiting any sparsity properties. Here, we compare run-times with respect to existing state of the art implementations of explicit Runge Kutta and implicit BDF methods. Finally, we conclude the paper in Section 8.

1.4 Notation

Besides mathematical standard notation, we use in this paper some conventions which are briefly introduced in this section. We denote with $\mathbb{S}^n_+ \subseteq \mathbb{R}^{n\times n}$ the set of symmetric and positive semi-definite matrices and with $\mathbb{S}^n_{++}$ the set of strictly positive definite matrices. In this context, we also write $A \succeq B$ for symmetric matrices A and B if and only if we have $A - B \in \mathbb{S}^n_+$. The corresponding strict semi-definite inequality is denoted as $A \succ B$ which is equivalent to $A - B \in \mathbb{S}^n_{++}$. Similarly, we denote the set of negative semi-definite (but not necessarily symmetric) matrices with

$$\mathbb{D}^n := \left\{ Z \in \mathbb{R}^{n\times n} \;\middle|\; Z + Z^\mathsf{T} \preceq 0 \right\} .$$

Finally, we introduce the set of $(n \times n)$-matrices whose eigenvalues have negative real parts:

$$\mathbb{M}^n := \left\{ Z \in \mathbb{R}^{n\times n} \;\middle|\; \exists P \succ 0 : \;\; ZP + PZ^\mathsf{T} \preceq 0 \right\} .$$

Note that we have $\mathbb{D}^n \subseteq \mathbb{M}^n$, as any matrix $Z \in \mathbb{D}^n$ satisfies the inequality $ZP + PZ^\mathsf{T} \preceq 0$ for $P = I$.

From Section 2 on, our focus will be on explicit methods. This allows us at many places to use a simplified notation regarding integration methods. For example, for Euler's explicit method we will use the notation

$$x^+ \;=\; x + hf \qquad \text{rather than} \qquad x_{k+1} = x_k + hf(x_k) \; .$$

Here, we neither write out the iteration index "k" nor the dependence of a right-hand side function f on the current state approximation x_k. We will only use this notation at places where it is clear from the context at which point f is evaluated. Similarly, we will usually write f' instead of $\frac{\partial f(x_k)}{\partial x}$ assuming again that is clear from the corresponding context where the derivative is evaluated. For higher order derivatives we use an analogous notation, e.g., f'' or $f^{(k)}$ for

the second or k-th derivative of f, respectively. In particular, we introduce for all $i \in \{1, 2, 3, \ldots\}$ a matrix $J_i(h, x) \in \mathbb{R}^{n \times n}$ which we define as

$$J_i(h, x) \;:=\; h^i \frac{\partial}{\partial x} \left[\frac{\partial^{i-1}}{\partial \epsilon^{i-1}} f(x + \epsilon f(x)) \right]_{\epsilon=0} .$$

At some places we will use the equivalent short notation $J_i = h^i f^{(i)} f^{i-1}$ which does not expand all dependencies on h and x.

2 Motivation and A Novel Explicit First Order Method

As our aim is to construct integration methods which satisfy suitable stability criteria, the implicit Euler integrator can be regarded as a good starting point. Thus, we are looking for an explicit method which chooses the next iterate x^+ such that

$$\frac{x^+ - x}{h} - f(x^+) \;\approx\; 0 .$$

Most of the practical implementations of the implicit Euler method choose a suitable Jacobian approximation function $Z : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}^{n \times n}$, such that

$$Z(h, x) \;\approx\; h f'(x) \;=\; J_1(h, x) ,$$

and solve the nonlinear equation in x^+ approximately using a Newton type method. If we choose h sufficiently small, we might implement the implicit Euler method by just taking one Newton type step in each iteration. This leads to an explicit method of the form

$$x^+ = x - \left[\frac{1}{h} I - \frac{1}{h} Z(h, x) \right]^{-1} \left(\frac{x - x}{h} - f(x) \right) \;=\; x + h M_1(h, x) f(x) . \quad (3)$$

Here, we have introduced the short hand

$$M_1(h, x) \;:=\; [1 - Z(h, x)]^{-1} .$$

For the special case that f is affine in x and $Z(h, x) = h f'(x)$ is the exact Jacobian, this method is equivalent to the implicit Euler method. In the above notation, we have still kept the dependence on x and h in order to highlight that the method is explicit.

One of the main contributions of this paper is to show that there exist low-rank Jacobian approximation functions Z for which the method (3) has non-trivial stability properties. In order to outline how this is possible, we analyze the integrator (3) for the rank-one Jacobian approximation function

$$Z(h, x) \;:=\; h \frac{f(x) f(x)^\mathsf{T}}{f(x)^\mathsf{T} f(x)} f'(x) .$$

Here, we assume for simplicity that we have $f(x) \neq 0$ such that the matrix $Z(h, x)$ is well defined. Using this low rank Jacobian approximation the inverse

$[I - Z(h,x)]^{-1}$ can be computed efficiently by using the Sherman-Morrison-Woodbury formula. The step for the proposed nonlinear integrator takes the simple form

$$x^+ \;=\; x + h\frac{f(x)^\mathsf{T} f(x)}{f(x)^\mathsf{T} f(x) - h f(x)^\mathsf{T} f'(x) f(x)} f(x)\;. \tag{4}$$

If f is dissipative, i.e., if $f'(x)$ is a negative semi-definite (but not necessarily symmetric) matrix, this step is well-defined for all step-sizes $h \geq 0$. For an implementation of the integrator step (4) we only need to evaluate the right-hand side $f(x)$ and either the forward directional derivative $f'(x)f(x)$ or the adjoint derivative $f(x)^\mathsf{T} f'(x)$. The computational cost for an implementation of such an integrator step is of order $\mathbf{O}(n)$.

Let us illustrate the stability of the integrator (4) by applying it to a decoupled linear test equation of the form

$$\dot{y}(t) \;=\; Ay(t) \quad \text{with} \quad A \;=\; -\mathrm{diag}(\lambda)\;. \tag{5}$$

Here, $\lambda \in \mathbb{R}^n_+$ is assumed to be a vector with non-negative components.

Theorem 1 *Let $h > 0$ be an arbitrary step size, let $x \in \mathbb{R}^n$ be a vector with $\mathrm{diag}(\lambda)x \neq 0$, and let x^+ be the iterate which is obtained by applying the method (4) to the linear test equation (5) such that*

$$x^+ \;=\; x - h\frac{\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)}{\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right) + h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)}\,\mathrm{diag}(\lambda)x\;.$$

Now, we have $\|x^+\|_2 < \|x\|_2$, i.e., the sequence of iterates which is generated by the integrator (4) is strictly contractive.

Before we prove the above theorem, we briefly introduce the following fundamental inequality:

Lemma 1 *The general inequality*

$$\left(\sum_{i=1}^n a_i b_i\right)\left(\sum_{i=1}^n a_i^3 b_i\right) \geq \left(\sum_{i=1}^n a_i^2 b_i\right)^2$$

holds for all vectors $a, b \in \mathbb{R}^n_+$ with non-negative components.

Proof : The above inequality can simply be proven by starting with the fact that we have $(a_i - a_j)^2 \geq 0$ which implies $a_i^2 + a_j^2 \geq 2a_i a_j$ for all indices $i, j \in \{1, \ldots, n\}$. Thus, we have

$$\begin{aligned}
\left(\sum_{i=1}^n a_i b_i\right)\left(\sum_{i=1}^n a_i^3 b_i\right) &= \sum_{i=1}^n a_i^4 b_i^2 + \sum_{(i,j),i<j} b_i b_j \left(a_i^3 a_j + a_j^3 a_i\right) \\
&= \sum_{i=1}^n a_i^4 b_i^2 + \sum_{(i,j),i<j} a_i a_j \left(a_i^2 + a_j^2\right) b_i b_j \\
&\geq \sum_{i=1}^n a_i^4 b_i^2 + 2\sum_{(i,j),i<j} a_i^2 a_j^2 b_i b_j \\
&= \left(\sum_{i=1}^n a_i^2 b_i\right)^2
\end{aligned}$$

which is the statement of the Lemma. □

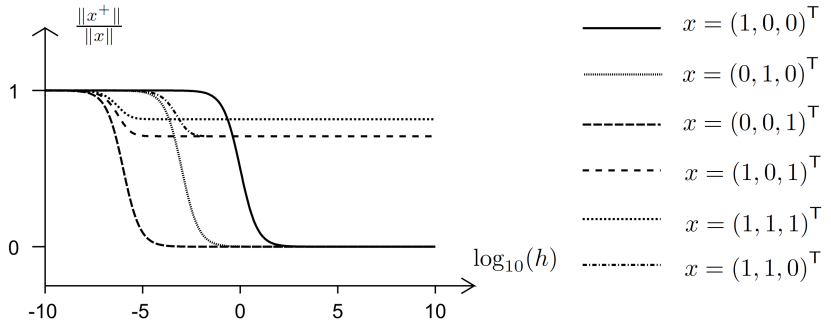

Fig. 1 The ratio $\frac{\|x^+\|}{\|x\|}$ for the integrator (4) applied to the stiff test differential equation $\dot{y} = \mathrm{diag}\left(-1\,,\,-10^3\,,\,-10^6\right)\,y$ for $h \in \left[10^{-10}, 10^{10}\right]$ and for various inputs $0 \neq x \in \mathbb{R}^3$.

Proof of Theorem 1: If we employ the inequality from Lemma 1 for the positive vectors $a = \lambda$ and $b = (x_1^2, \ldots, x_n^2)^\mathsf{T}$, we find

$$\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)\left(\sum_{i=1}^n \lambda_i x_i^2\right) \geq \left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)^2 \,.$$

Now, if we multiply the above relation with $h\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right) > 0$ and add a positive term to right hand-side, the inequality remains correct, i.e., we have

$$\left[\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right) + h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)\right]\left(\sum_{i=1}^n \lambda_i x_i^2\right)\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right) \geq h\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)^3 \,.$$

In the next step, we divide the above inequality on both sides by the strictly positive term $\frac{1}{h}\left[\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right) + h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)\right]^2 > 0$. This yields

$$\frac{h\left(\sum_{i=1}^n \lambda_i x_i^2\right)\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)}{\left[\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)+h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)\right]} \geq \frac{h^2\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)^3}{\left[\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)+h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)\right]^2} \,. \tag{6}$$

On the other hand, we can expand the expression for the difference

$$\begin{aligned}\|x^+\|_2^2 - \|x\|_2^2 \;&=\; h^2 \frac{\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)^3}{\left[\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)+h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)\right]^2} - 2h \frac{\left(\sum_{i=1}^n \lambda_i x_i^2\right)\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)}{\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)+h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)} \\ &\overset{(6)}{\leq} -h \frac{\left(\sum_{i=1}^n \lambda_i x_i^2\right)\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)}{\left(\sum_{i=1}^n \lambda_i^2 x_i^2\right)+h\left(\sum_{i=1}^n \lambda_i^3 x_i^2\right)} \;<\; 0 \,.\end{aligned}$$

Consequently, we find the contractivity relation $\|x^+\|_2 < \|x\|_2$. □

Example 1 In order to demonstrate the stability properties of the integrator (4) numerically, we apply it to the linear test differential equation

$$\dot{y}(t) \;=\; \mathrm{diag}\left(-1\,,\,-10^3\,,\,-10^6\right)\,y(t)$$

which is stiff. Figure 1 shows a plot of the ratio of the Euclidean norm of the output x^+ over the Euclidean norm of the input x for $h \in \left[10^{-10}, 10^{10}\right]$ and for various inputs $0 \neq x \in \mathbb{R}^3$. This can be repeated for many randomly chosen inputs observing that the ratio $\|x^+\|/\|x\|$ is always strictly smaller than 1 as guaranteed by Theorem 1.

Let us briefly come back to our discussion from the introduction where we have argued that Lambert's nonlinear method (2) looses its stability properties if we add a constant offset to the test equation. This is due to the fact that nonlinear methods are—in contrast to linear methods—in general not invariant with respect to affine transformations of the differential equation. In fact, this example shows that the concept of A-stability based on homogenous and scalar linear test equations is not sufficient for the assessment of the stability of nonlinear integration methods. Concerning stability analysis for nonlinear test equations, the concept of B-stability [17,27] is commonly used for the assessment of implicit linear methods. However requiring B-stability appears to be quite restrictive and there is not much hope that we will ever find generic explicit methods which are B-stable.

Fortunately, the class of nonlinear methods we are considering in this paper allows us to construct a class of affine test equations for which the stability of the method can be proven. In order to prepare this stability analysis, we regard the differential equations

$$\dot{y}_1(t) = f(y_1(t)) \quad \text{with} \quad y_1(0) \;=\; x_1 \quad \text{and} \tag{7}$$

$$\dot{y}_2(t) = \alpha Q f\left(\frac{1}{\alpha} Q^{\mathsf{T}}(y_2(t) - b)\right) \quad \text{with} \quad y_2(0) \;=\; x_2 \;:=\; \alpha Q x_1 + b\,, \tag{8}$$

where $Q \in \mathbb{R}^{n\times n}$ with $Q^{\mathsf{T}}Q = I$ is an orthogonal matrix, $b \in \mathbb{R}^n$ a vector, and $0 \neq \alpha \in \mathbb{R}$ a scaling factor. Note that the differential equations (7) and (8) are equivalent in the sense that we have

$$\forall t \in \mathbb{R}: \quad y_2(t) \;=\; \alpha Q y_1(t) + b$$

assuming that the solution trajectories y_1 and y_2 uniquely exist. The following theorem summarizes three important invariance properties of the method (4):

Theorem 2 *Let x_1^+ be the iterate which is obtained by applying the method* (4) *to the differential equation* (7) *and let x_2^+ be the iterate which is obtained by applying the method* (4) *to the transformed differential equation* (8)*. If we use in both cases the same stepsize $h \geq 0$, then we have*

$$x_2^+ \;=\; \alpha Q x_1^+ + b\,.$$

In other words, the method (4) *is invariant under orthogonal transformations, under translations, as well as under scalar multiplications.*

Proof : Using $\frac{1}{\alpha}Q^{\mathsf{T}}(x_2 - b) = x_1$ we can substituting the transformed differential equation in (4) which yields the following integrator step:

$$\begin{aligned} x_2^+ &= x_2 + \frac{h\alpha^3 f(x_1)^{\mathsf{T}} Q^{\mathsf{T}} Q f(x_1)}{\alpha^2 f(x_1)^{\mathsf{T}} Q^{\mathsf{T}} Q f(x_1) - h\alpha^2 f(x_1)^{\mathsf{T}} Q^{\mathsf{T}} \left[Q f'(x_1) Q^{\mathsf{T}}\right] Q f(x_1)} Q f(x_1) \\ &= \alpha Q \left(x_1 + h \frac{f(x_1)^{\mathsf{T}} f(x_1)}{f(x_1)^{\mathsf{T}} f(x_1) - h f(x_1)^{\mathsf{T}} f'(x_1) f(x_1)} f(x_1) \right) + b \\ &= \alpha Q x_1^+ + b\,. \end{aligned}$$

This is the statement of the theorem. □

Combining Theorem 1 and Theorem 2 we can state the following result:

Corollary 1 *Let $h > 0$ be an arbitrary step size, $A \in -\mathbb{S}^n_{++}$ a given negative definite matrix, $b \in \mathbb{R}^n$ a given vector, $x \in \mathbb{R}^n$ a vector with $x \neq -A^{-1}b$, and x^+ the iterate which is obtained by applying the method* (4) *to the linear test equation*

$$\dot{y}(t) \;=\; Ay(t) + b\,.$$

Then, we have $\left\|x^+ + A^{-1}b\right\|_2 < \left\|x + A^{-1}b\right\|_2$.

Proof : This statements follows immediately from the fact that any negative semi-definite matrix A can be written as $A = Q\,\mathrm{diag}(\lambda)\,Q^\mathsf{T}$, where Q is an orthogonal matrix. Thus, we can use the invariance of the iterates with respect to additive offsets and orthogonal transformations applying the statement of Theorem 1. □

Remark 1 Although Theorem 2 proves the invariance of the integrator (4) under orthogonal transformations of the differential equation, we have to be aware of the fact that the method (4) is not invariant under general affine transformations. The problem can directly be seen from equation (4), as the denominator $f(x)^\mathsf{T} f(x) - h f(x)^\mathsf{T} f'(x) f(x)$ can eventually become 0 for some $h > 0$ and $f(x) \neq 0$ if we consider stable but non-dissipative systems with $f'(x) \notin \mathbb{D}^n$ (but possibly $f'(x) \in \mathbb{M}^n$). This is a disadvantage of the method (4) in comparison to implicit linear methods. The corresponding problem can be avoided by choosing a different Jacobian approximation function Z as we shall discuss in Section 4.

3 Construction of Higher Order Methods

The aim of this section is to construct a class of nonlinear explicit integrators which have a higher consistency order than the method (4). In order to motivate the construction, we start with a singly diagonal implicit Runge-Kutta (SDIRK) method with Butcher Tableau

$$\begin{array}{c|cc} 0 & 1 & 0 \\ 2 & 1 & 1 \\ \hline & \frac{3}{2} & -\frac{1}{2} \end{array} \qquad \text{or explicitly} \qquad \begin{array}{l} k_1 := f(x + hk_1) \\ k_2 := f(x + hk_1 + hk_2) \\ \hline x^+_{\mathrm{RK}} := x + \frac{3}{2}hk_1 - \frac{1}{2}hk_2 \quad . \end{array} \tag{9}$$

It can easily be checked that this SDIRK method is A-stable[1] and that it has the consistency order 2, i.e., we have

$$x^+_{\mathrm{RK}} \;=\; x + hf(x) + \frac{1}{2}h^2 f'(x) f(x) + \mathbf{O}(h^3)\,.$$

[1] Recall that a non-degenerate Runge-Kutta method with coefficients $(\mathcal{A}, \mathcal{b})$ is A-stable if only if there exists a symmetric and positive definite matrix $R \in \mathbb{S}^s_{++}$ which satisfies $R\mathcal{A} + \mathcal{A}^\mathsf{T} R - \mathcal{b}\mathcal{b}^\mathsf{T} \succeq 0$ as well as $R(1, \ldots, 1)^\mathsf{T} = \mathcal{b}$ (for a proof of this statement we refer

In the next step, we construct an approximate implementation of the above method by applying one Newton type step to the nonlinear equations for k_1 and k_2 using the same Jacobian approximation $Z(h,x) \approx hf'(x)$. In other words, we define

$$\begin{aligned} k_1 \approx \bar{k}_1 &:= [I - Z(h,x)]^{-1} f(x) \\ \text{and} \quad k_2 \approx \bar{k}_2 &:= [I - Z(h,x)]^{-1} f(x + h\bar{k}_1) \, . \end{aligned}$$

Now, we ask the question whether we can find an approximation of $\bar{k}_2$ which can be computed by evaluating f or directional derivatives of f at the point x only. One way to do this is by Taylor expansion. For this aim, we use once more $Z(h,x) \approx hf'(x)$ such that

$$\begin{aligned} f(x + h\hat{k}_1) &= f(x) + hf'(x)\hat{k}_1 + \mathbf{O}(h^2) \\ &\approx f(x) + Z(h,x)[I - Z(h,x)]^{-1} f(x) \\ &= f(x) - [I - Z(h,x)][I - Z(h,x)]^{-1} f(x) + [I - Z(h,x)]^{-1} f(x) \\ &= [I - Z(h,x)]^{-1} f(x) \;=\; M_1(h,x) f(x) \end{aligned} \tag{10}$$

which motivates the definition

$$\begin{aligned} \hat{k}_1 := M_1(h,x) f(x) \quad &= \; \bar{k}_1 \\ \text{and} \quad \hat{k}_2 := M_1(h,x)^2 f(x) \; &\overset{(10)}{\approx} \; \bar{k}_2 \, . \end{aligned}$$

If we use this approximation in order to modify the original diagonal implicit Runge-Kutta integrator step, we obtain an explicit method of the form

$$x^+ := x + \frac{3}{2} h\hat{k}_1 - \frac{1}{2} h\hat{k}_2 \;=\; x + h\left[\frac{3}{2} M_1(h,x) - \frac{1}{2} M_1(h,x)^2\right] f(x) \, . \tag{11}$$

In order to justify the above approximation of the implicit Runge-Kutta step x^+_{RK} with the explicit step x^+, we state the following Lemma:

Lemma 2 *Let x^+_{RK} and x^+ be constructed as above and let $Z(h,x)$ be a linear function in h. Now, the following statements hold:*

1. *If $Z(h,x) = hf'(x)$ is the exact Jacobian approximation and if f is affine in x, then the steps x^+ and x^+_{RK} coincide, i.e., we have $x^+ = x^+_{\mathrm{RK}}$.*
2. *If the Jacobian approximation is exact in the direction $f(x)$ such that $Z(h,x)f(x) = hf'(x)f(x)$, then we have*

$$\| x^+ - x^+_{RK} \| \;\leq\; \mathbf{O}(h^3) \, .$$

In other words, the integration method (11) *has the consistency order* 2.

to [42]). In our case this condition can be satisfied with

$$\mathcal{A} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} , \quad \mathcal{b} = \begin{pmatrix} \frac{3}{2} \\ -\frac{1}{2} \end{pmatrix} , \quad \text{and} \quad R = \begin{pmatrix} \frac{5}{2} & -1 \\ -1 & \frac{1}{2} \end{pmatrix} \succ 0$$

which proves the A-stability of the method (9).

Proof : If we apply Newton's method to an affine equation the method converges after one step. Thus, if $Z(h,x)$ is the exact Jacobian approximation and if f is affine, we must have $k_1 = \bar{k}_1$ as well as $k_2 = \bar{k}_2$. Additionally, if f is affine, the approximation (10) is exact and we have $k_2 = \bar{k}_2 = \hat{k}_2$. Consequently, we have $x^+ = x^+_{\mathrm{RK}}$ which is the first statement of the Lemma. In order to also verify the second statement, we first note that we have

$$\begin{aligned}\bar{k}_1 &= [I - Z(h,x)]^{-1} f(x) \\ &= [I + Z(h,x)] f(x) + \mathbf{O}(h^2) \;=\; f(x) + hf'(x)f(x) + \mathbf{O}(h^2)\end{aligned}$$

as long as Z is affine in h while the condition $Z(h,x)f(x) = hf'(x)f(x)$ is satisfied. Consequently, we have $\|k_1 - \bar{k}_1\| \leq \mathbf{O}(h^2)$. Using a similar argumentation we also find

$$\|k_2 - \bar{k}_2\| \leq \mathbf{O}(h^2) \quad \text{as well as} \quad \|\hat{k}_2 - \bar{k}_2\| \leq \mathbf{O}(h^2)\,.$$

Combining everything yields

$$\|x^+ - x^+_{\mathrm{RK}}\| \;\leq\; \frac{3}{2}h\|\hat{k}_1 - k_1\| + \frac{1}{2}h\|\hat{k}_2 - k_2\| \;\leq\; \mathbf{O}(h^3)$$

which proves the second statement of the Lemma. □

One immediate consequence of the above Lemma is that the method (11) is A-stable if we use the exact Jacobian approximation $Z(h,x) = hf'(x)$, as the the method (11) coincides in this case for affine test equations with the A-stable SDIRK method (9). However, the main point of this paper is that there exist low rank Jacobian approximation functions for which the method (11) has still interesting stability properties, although we allow the computation of a few derivative directions only. The corresponding stability analysis can be found in Section 4.

As it was already mentioned in the introduction, we will from now on use a simplified symbolic notation, i.e., we do not write out all dependencies on x and h whenever this appears appropriate. For example, equation (11) is in our simplified notation written as

$$x^+ \;=\; x + h\left[\frac{3}{2}M_1 - \frac{1}{2}M_1^2\right] f \quad \text{with} \quad M_1 \;=\; [1 - Z]^{-1}\,.$$

Additionally, we introduce the matrix valued functions $M_i : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$ which we define as

$$M_i(h,x) \;:=\; h^i M_1(h,x) \frac{\partial}{\partial x}\left[\frac{\partial^{i-1}}{\partial \epsilon^{i-1}} f(x + \epsilon M_1(h,x) f(x))\right]_{\epsilon=0} M_1(h,x)\,.$$

for all $i \in \{2,3,4,\ldots\}$. Using our short hand convention this equation for the matrix $M_i(h,x)$ reads

$$M_i \;=\; h^i M_1 f^{(i)} (M_1 f)^{i-1} M_1 \quad \text{for } \; i \in \{2,3,4,\ldots\}\,.$$

In the following, we are interested in explicit one-step integration methods of the form

$$x^+ \;:=\; x \,+\, h\,\mathcal{P}[M]\,f\;, \tag{12}$$

where $\mathcal{P} : \mathbb{R}^{n\times n} \times \ldots \times \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is a polynomial in the matrices $M_1, M_2, \ldots M_N$. Here, N is a given integer and $M := (M_1, M_2, \ldots, M_N)$. For example, if we choose the polynomial

$$\mathcal{P}_1[M] := M_1$$

we obtain the method from equation (3) while the choice

$$\mathcal{P}_2[M] := \frac{3}{2}M_1 - \frac{1}{2}M_1^2$$

yields the method (11). Note that for the numerical computation of the step in equation (12), it is not necessary to compute the matrices $M_2, \ldots, M_N$ explicitly. This is due to the fact that we can evaluate the matrix-vector product $\mathcal{P}[M]\,f$ recursively using automatic differentiation. Here, it depends on the particular structure of the polynomial $\mathcal{P}[M]$ how many higher order directional derivatives are needed for the evaluation of a step. Finally, we note that an integration method of the type (12) is uniquely characterized if we specify the Jacobian approximation function Z as well as the coefficients of the polynomial $\mathcal{P}$ which is defined on the non-commutative ring of $(n \times n)$-matrices.

So far we have only discussed how to construct the polynomials $\mathcal{P}_1$ and $\mathcal{P}_2$ for first and second order methods. For the construction of higher order methods we prescind again from the construction via SDIRK methods, as it is much easier to derive consistency conditions for the polynomial $\mathcal{P}$ directly by computing a Taylor expansion of the method (12) and comparing coefficients. In order to explain this in detail, we first regard a Taylor expansion of the solution trajectory $y(t)$ at $t = 0$ assuming $x = y(0) = y_0$. This Taylor expansion has the form

$$\begin{aligned}
y(h) &= y(0) + h\frac{\mathrm{d}y(0)}{\mathrm{d}t} + \frac{1}{2}h^2\frac{\mathrm{d}^2y(0)}{\mathrm{d}t^2} + \frac{1}{6}h^3\frac{\mathrm{d}^3y(0)}{\mathrm{d}t^3} + \ldots \\
&= x + hf + \frac{1}{2}h^2 f'f + \frac{1}{6}h^3 f''ff + \frac{1}{6}h^3 f'f'f + \ldots \\
&= x + h\underbrace{\left(I + \frac{1}{2}J_1 + \frac{1}{6}J_2 + \frac{1}{6}J_1^2 + \ldots + \frac{1}{N!}J_{N-1} \right)}_{=:\,\mathcal{Q}[J_1,\ldots,J_{N-1}]} f + \mathbf{O}\left(h^{N+1}\right) \\
&= x + h\,\mathcal{Q}[J_1, \ldots, J_{N-1}]\,f \;+\; \mathbf{O}\left(h^{N+1}\right)\;.
\end{aligned}$$

In this context, we have introduced the matrix valued function $\mathcal{Q}[J]$ which is a matrix polynomial in the higher order directional derivative matrices $J_1, J_2 \ldots, J_{N-1}$. Planning a comparison of coefficients, it is helpful to note

that the Taylor expansions of the matrix powers M_1^m are for all $m \in \mathbb{N}$ given by:

$$M_1^m \;=\; [I - Z]^{-m} \;=\; I + \sum_{i=1}^{N-1} \binom{m+i-1}{i} Z^i + \mathbf{O}\left(h^N\right) \;. \tag{13}$$

In the next step, we can formulate conditions under which an integration method of the form (12) is consistent with order N:

Lemma 3 *Let the polynomial $\mathcal{Q}$ be defined as above. Now, the integration method* (12) *has the (global) consistency order N, if the following two conditions are satisfied:*

1. *The polynomial $\mathcal{P}$ satisfies the condition*

$$\mathcal{P}[M]f \;=\; \mathcal{Q}[Z, J_2, \ldots, J_{N-1}]f + \mathbf{O}(h^N) \;.$$

2. *The Jacobian approximation Z is a linear function in h and satisfies the directional condition*

$$\mathcal{Q}[Z, J_2, \ldots, J_{N-1}]f = \mathcal{Q}[J_1, \ldots, J_{N-1}]f \;.$$

Note that the first condition is an affine condition for coefficients of the polynomial $\mathcal{P}$ which can be obtained by a comparison of coefficients.

Proof : Using the definition of x^+ as well as the definition of $\mathcal{Q}$ and the conditions 1) and 2) from the statement, we obtain the following relation:

$$\begin{aligned} x^+ - y(h) &= h\,(\;\mathcal{P}[M]f \;-\; \mathcal{Q}[J_1, \ldots, J_{N-1}]f\;) + \mathbf{O}(h^{N+1}) \\ &= h\underbrace{(\;\mathcal{Q}[Z, M_2, \ldots, M_{N-1}]f(x) \;-\; \mathcal{Q}[J_1, \ldots, J_{N-1}]f\;)}_{=0} + \mathbf{O}(h^{N+1}) \;. \end{aligned}$$

Thus, we may conclude that the local consistency order is at least $N+1$ which implies that the global consistency order must be at least N. □

Example 2 In this tutorial example we explain in detail how to construct a method with consistency order $N = 3$ using the conditions from Lemma 3. For this aim, we first note that the associated matrix polynomial $\mathcal{Q}$ is given by

$$\mathcal{Q}[J_1, J_2] \;=\; I + \frac{1}{2}J_1 + \frac{1}{6}J_1^2 + \frac{1}{6}J_2 \;.$$

In the first step we try to determine coefficients $\beta_1, \beta_2, \beta_3$, and γ such that the polynomial

$$\mathcal{P}_3[M] \;:=\; \beta_1 M_1 + \beta_2 M_1^2 + \beta_3 M_1^3 + \gamma M_2$$

satisfies the first condition in Lemma 3. Using the Taylor expansion formula (13) we find that

$$\begin{aligned} M_1 &= I + Z + Z^2 + \mathbf{O}(h^3) \\ M_1^2 &= I + 2Z + 3Z^2 + \mathbf{O}(h^3) \\ M_1^3 &= I + 3Z + 6Z^2 + \mathbf{O}(h^3) \quad \text{and} \quad M_2 \;=\; J_2 + \mathbf{O}(h^3) \;. \end{aligned}$$

Consequently, we find the expansion

$$\mathcal{P}_3[M] \;=\; \begin{pmatrix} \beta_1 + \;\; \beta_2 + \;\; \beta_3 \\ \beta_1 + 2\beta_2 + 3\beta_3 \\ \beta_1 + 3\beta_2 + 6\beta_3 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} I \\ Z \\ Z^2 \end{pmatrix} + \gamma J_2 + \mathbf{O}(h^3)\;.$$

In the next step, we compare the coefficients of the polynomials $\mathcal{P}_3[M]$ and $\mathcal{Q}[Z, J_2]$ finding the affine consistency conditions

$$\begin{pmatrix} 1\;1\;1\;0 \\ 1\;2\;3\;0 \\ 1\;3\;6\;0 \\ 0\;0\;0\;1 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \gamma \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/6 \\ 1/6 \end{pmatrix} \quad \Longrightarrow \quad \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \gamma \end{pmatrix} \;=\; \frac{1}{6} \begin{pmatrix} 10 \\ -5 \\ 1 \\ 1 \end{pmatrix}\;.$$

Now, Lemma 3 guarantees that the explicit integration method (12) has the consistency order 3, if we employ the polynomial

$$\mathcal{P}_3[M] \;=\; \frac{10}{6} M_1 - \frac{5}{6} M_1^2 + \frac{1}{6} M_1^3 + \frac{1}{6} M_2$$

together with the exact Jacobian $Z = J_1$. More generally, the method has still the consistency order 3, if we have $Z \neq J_1$ as long as the two directional conditions of the form $Zf = J_1 f$ and $Z^2 f = J_1^2 f$ are satisfied.

Now, we can use an analogous procedure to also derive consistent polynomials $\mathcal{P}[M]$ for other orders. For example, for the consistency orders $1 - 4$ possible associated polynomials are shown in Table 1. Note that the computation of such polynomials can easily be automated for arbitrary consistency orders using modern computer algebra programs. This is due to the fact that the consistency condition from Lemma 3 yields an affine condition for the coefficients of the polynomial $\mathcal{P}[M]$.

Name	Polynomial	Order
$\mathcal{P}_1[M]$	M_1	1
$\mathcal{P}_2[M]$	$\frac{3}{2} M_1 - \frac{1}{2} M_1^2$	2
$\mathcal{P}_3[M]$	$\frac{10}{6} M_1 - \frac{5}{6} M_1^2 + \frac{1}{6} M_1^3 + \frac{1}{6} M_2$	3
$\mathcal{P}_4[M]$	$\frac{39}{24} M_1 - \frac{17}{24} M_1^2 + \frac{1}{24} M_1^3 + \frac{1}{24} M_1^4 + \frac{1}{24} M_2 M_1 + \frac{1}{24} M_3$	4

Table 1 Polynomials $\mathcal{P}[M]$ for integration methods with consistency order 1 to 4.

4 Stability Analysis for Linear Test Equations

The aim of this section is to derive stability conditions for the explicit integration method (12). Recall that this integration method is consistent with order N if the conditions from Lemma 3 are satisfied. Unfortunately, not every consistent method of the form (12) comes along with suitable stability properties and thus we have to introduce additional requirements. Let us analyze how the method (12) performs on linear differential equations of the form

$$\dot{y}(t) = f_A(y(t)) := Ay(t) \; .$$

Here, the higher order terms $M_2, \ldots, M_N$ are all zero such that the integrator step can be written as:

$$x^+ \; = \; x + h\mathcal{P}[(M_1, 0, \ldots, 0)]Ax \; = \; x + \sum_{i=0}^{m} \beta_i M_1^i J_1 x \; .$$

The constants $\beta_1, \ldots, \beta_m \in \mathbb{R}$ denote the coefficients in the matrix polynomial $\mathcal{P}[M]$ which are associated with the monomials of the form $M_1, M_1^2, \ldots, M_1^m$. In this context, m denotes the largest power of M_1 which occurs in $\mathcal{P}$. Note that the step can also be written as

$$x^+ \; = \; \hat{R}(Z, J_1)x \; , \quad \text{where we define} \quad \hat{R}(Z, J_1) \; := \; I + \sum_{i=1}^{m} \beta_i \left[I - Z\right]^{-i} J_1 \; .$$

Let us introduce the following directional condition on the Jacobian approximation Z:

Definition 1 (Adjoint Stability Condition) We say that the Jacobian approximation $Z(h, x)$ satisfies the adjoint stability condition if there exists a symmetric and positive semi-definite function $Z^* : \mathbb{R}^n \to \mathbb{S}_+^n$ such that the following conditions are satisfied:

1. We have $Z(h, x) \; = \; hZ^*(x)f'(x)$ for all $h \geq 0$ and for all $x \in \mathbb{R}^n$.
2. We have $Z^*(x)f(x) \; = \; f(x)$.

Note that the exact Jacobian $Z(h, x) = hf'(x)$ always satisfies these two conditions, as we can use the trivial choice $Z^*(x) = I$ in this case.

The name "adjoint stability condition" of the above definition is motivated by the following observation: if we have a low rank matrix $Z^*(x)$ which satisfies the directional condition $Z^*(x)f(x) \; = \; f(x)$, then we do not have to compute the full Jacobian $f'(x)$ in order to construct the Jacobian approximation $Z(h, x) \; = \; hZ^*(x)f'(x)$. This is due to fact that we can compute the low-rank matrix-matrix product $Z^*(x)f'(x)$ directly using the adjoint mode of automatic differentiation. In this case the number of required backward directions depends only on the rank of the matrix $Z^*(x)$.

Definition 2 (Reduced Stability Function) We define the reduced stability function $R : \mathbb{R}^{n \times n} \to \bar{\mathbb{R}}^{n \times n}$ as $R(Z) \; := \; \hat{R}(Z, Z)$ for all $Z \in \mathbb{R}^{n \times n}$.

The main reason why the introduction of the adjoint stability condition is useful is summarized in the form of the following technical result:

Lemma 4 *If the adjoint stability condition from Definition 1 is satisfied, then we have*

$$x^+ \;=\; \hat{R}(Z, J_1)x \;=\; R(Z)x \;.$$

Proof : As our test equation is linear in x, we have $f(x) = f'(x)x$. Thus, the adjoint stability condition yields

$$J_1(x)x \;=\; hf(x) \;=\; hZ^*(x)f(x) \;=\; hZ^*(x)f'(x)x \;=\; Z(h,x)x \;.$$

Consequently, the integrator step takes the form

$$\begin{aligned} x^+ = \hat{R}(Z, J_1)x \;&=\; x + \sum_{i=1}^{m} \beta_i \left[I - Z\right]^{-i} J_1 x \\ &= x + \sum_{i=1}^{m} \beta_i \left[I - Z\right]^{-i} Z x \;=\; R(Z)x \end{aligned}$$

which is the statement of the Lemma. □

Being at this point, it should be highlighted that the introduction of the adjoint stability condition in combination with Lemma 4 is one of the key ideas of this paper which makes the proposed integration scheme practical and applicable for stiff differential equations. However, besides the adjoint stability condition for the matrix Z we need an additional condition for the polynomial $\mathcal{P}$. In order to introduce this condition, we write the reduced stability function R in the form

$$R(Z) \;=\; I + \sum_{i=1}^{m} \beta_i \left[I - Z\right]^{-i} Z \;=\; \left[I - Z\right]^{-m} N(Z) \;.$$

Here, $N(Z)$ denotes a polynomial in Z whose coefficients depend on the constants $\beta_1, \ldots, \beta_m$. Additionally, we define the function $\Delta : \mathbb{R}^{n\times n} \to \mathbb{S}^n$ as

$$\forall Z \in \mathbb{R}^{n\times n} : \quad \Delta(Z) \;:=\; \left[I - Z\right]^m \left[I - Z^\mathsf{T}\right]^m - N(Z)N(Z)^\mathsf{T} \;. \qquad (14)$$

Here, Δ is a symmetric matrix polynomial in Z and Z^T. In the following, we define the short hands

$$\mathcal{Z}_1 \;:=\; \left(\, Z \,,\, Z^2 \,,\, \ldots \,,\, Z^m \right) \quad \text{and} \quad \mathcal{Z}_2 \;:=\; \left(\, I \,,\, Z^\mathsf{T}, \ldots, \left(Z^{m-1}\right)^\mathsf{T} \right)^\mathsf{T}$$

and introduce an algebraic condition on the polynomial $\mathcal{P}$:

Definition 3 (Algebraic Stability Condition for $\mathcal{P}$) We say that the polynomial $\mathcal{P}$ satisfies the algebraic stability condition, if there exists a symmetric and positive semi-definite matrix $Q \in \mathbb{S}_+^m$ as well as a matrix $S \in \mathbb{R}^{m\times s}$ such that we have

$$\Delta(Z) \;=\; \mathcal{Z}_1 \,(Q \otimes I)\, \mathcal{Z}_1^{\mathsf{T}} - \sum_{i=1}^{s} (S_i \otimes I)^{\mathsf{T}}\, \mathcal{Z}_2 \left(Z + Z^{\mathsf{T}}\right) \mathcal{Z}_2^{\mathsf{T}}\, (S_i \otimes I) \qquad (15)$$

for all $Z \in \mathbb{R}^{n\times n}$. Here, S_i denotes the i-th column of the matrix S and $\otimes$ the Kronecker product.

Before we explain why the algebraic stability condition from the above definition turns out to be useful, we first start by working out the following example which explains how we can check this condition in practice:

Example 3 Let us regard the polynomial $\mathcal{P}_2[M] = \frac{3}{2}M_1 - \frac{1}{2}M_1^2$ from Table 1. The reduced stability function which is associated with this polynomial can be written as

$$R(Z) \;=\; I + \frac{3}{2}[1-Z]^{-1}Z - \frac{1}{2}[I-Z]^{-2}Z \;=\; [I-Z]^{-2}\left(I - Z - \frac{1}{2}Z^2\right)\,,$$

i.e., the polynomial N is in this case given by $N(Z) \;=\; I - Z - \frac{1}{2}Z^2$. In the next step, we expand the expression for the polynomial Δ. This yields

$$\Delta(Z) = -Z^{\mathsf{T}} - Z + \frac{3}{2}Z^{\mathsf{T}}Z^{\mathsf{T}} + 3ZZ^{\mathsf{T}} - \frac{5}{2}ZZ^{\mathsf{T}}Z^{\mathsf{T}} + \frac{3}{2}ZZ - \frac{5}{2}ZZZ^{\mathsf{T}} + \frac{3}{4}ZZZ^{\mathsf{T}}Z^{\mathsf{T}}.$$

In order to prove that the algebraic stability condition (15) is satisfied we collect terms and complete squares finding

$$\Delta(Z) = -\left(I - \frac{3}{2}Z\right)\left(Z + Z^{\mathsf{T}}\right)\left(I - \frac{3}{2}Z\right)^{\mathsf{T}} - \frac{1}{4}Z(Z + Z^{\mathsf{T}})Z^{\mathsf{T}} + \frac{3}{4}ZZZ^{\mathsf{T}}Z^{\mathsf{T}}.$$

Consequently, the polynomial $\mathcal{P}_3$ satisfies the algebraic stability condition with

$$Q \;=\; \begin{pmatrix} 0 & 0 \\ 0 & \frac{3}{4} \end{pmatrix} \quad \text{and} \quad S \;=\; \begin{pmatrix} 1 & 0 \\ -\frac{3}{2} & \frac{1}{2} \end{pmatrix}\,.$$

Remark 2 For the polynomial $\mathcal{P}_1$ from Table 1 it can easily be checked that the algebraic stability condition (15) is satisfied for $Q = 1$ and $S = 1$. For the polynomial $\mathcal{P}_2$ a proof of algebraic stability has been elaborated in Example 3. For the polynomial $\mathcal{P}_3$ it can still be verified with "paper and pencil" that the algebraic stability condition is satisfied. For example, the matrices

$$Q \;=\; \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{12} & 0 \\ 0 & 0 & \frac{5}{9} \end{pmatrix} \quad \text{and} \quad S \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{5}{2} & \frac{1}{4} & \frac{1}{48} & 0 \\ \frac{5}{3} & -\frac{2}{3} & 0 & \frac{1}{9} \end{pmatrix}$$

satisfy the condition (15) for the polynomial $\mathcal{P}_3$. However, in general the computations become some sort of cumbersome if we do them by hand. Thus, we suggest to automate the algebraic stability check with a computer program. This can for example be done by employing a suitable polynomial semi-definite program (SDP) solver which searches for feasible matrices S and $Q \succeq 0$. In this paper we have used the software CVX [44] sequentially to solve the corresponding SDP finding that the polynomials $\mathcal{P}_4$ satisfies the algebraic stability condition, too. In this context, we also refer to [28,33] for a general overview on the theory of positive polynomials and suitable numerical solvers.

Being at this point, we recall that $\mathbb{D}^n$ denotes the set of negative semi-definite (but not necessarily symmetric) matrices in $\mathbb{R}^{n\times n}$ while $\mathbb{M}^n$ denotes the set of matrices whose eigenvalues have all non-positive real parts. The aim of the following consideration is to analyze the properties of the reduced stability function R on the domains $\mathbb{D}^n$ and $\mathbb{M}^n$, respectively.

Theorem 3 *Let the polynomial $\mathcal{P}$ satisfy the algebraic stability condition from Definition 3. With R denoting the reduced stability function and $\rho(R(Z))$ the spectral radius of $R(Z)$ the following statements hold:*

1. *We have $R(Z)^\mathsf{T} R(Z) \preceq I$ for all $Z \in \mathbb{D}^n$.*
2. *We have $\rho(R(Z)) \leq 1$ for all $Z \in \mathbb{M}^n$.*

Proof of Statement 1: Note that we have $Z + Z^\mathsf{T} \preceq 0$ for all $Z \in \mathbb{D}^n$ which implies that the term

$$-\sum_{i=1}^{s} (S_i \otimes I)^\mathsf{T} \mathcal{Z}_2 \left(Z + Z^\mathsf{T}\right) \mathcal{Z}_2^\mathsf{T} (S_i \otimes I)$$

in the algebraic stability condition (3) is positive-semi definite. Consequently, equation (3) implies that we have $\Delta(Z) \succeq 0$ for all $Z \in \mathbb{D}^n$. Thus, we have

$$\forall Z \in \mathbb{D}^n : \quad [I - Z]^m \left[I - Z^\mathsf{T}\right]^m \succeq N(Z) N(Z)^\mathsf{T} .$$

We assume for a moment that $N(Z)$ is invertible such that we can multiply this inequality with the matrix $N(Z)^{-1}$ from the left and, simultaneously, with its transpose from the right. This yields

$$I \preceq N(Z)^{-1} [I - Z]^m \left[I - Z^\mathsf{T}\right]^m \left(N(Z)^\mathsf{T}\right)^{-1} = R(Z)^{-1} \left(R(Z)^\mathsf{T}\right)^{-1} .$$

Inverting this inequality shows that we have $R(Z)^\mathsf{T} R(Z) \preceq I$ for all $Z \in \mathbb{D}^n$ for which $N(Z)$ is invertible. Now, let us assume that we have a $Z \in \mathbb{D}^n$ for which $N(Z)$ is not invertible. As N is a polynomial in Z with $N(0) = I$, we know that the exists for every $\epsilon > 0$ a $\bar{Z} \in \mathbb{D}^n$ with $\|Z - \bar{Z}\| \leq \epsilon$ such that $N(\bar{Z})$ is invertible. As the function $R(Z)$ is continuous on the domain $\mathbb{D}^n$ while we have $R\left(\bar{Z}\right)^\mathsf{T} R\left(\bar{Z}\right) \preceq I$, we may conclude that we have $R(Z)^\mathsf{T} R(Z) \preceq I$ for all $Z \in \mathbb{D}^n$. □

Proof of Statement 2: In order to show $\rho(R(Z)) \leq 1$ for a given $Z \in \mathbb{M}^n$ it is enough to prove that there exists a positive definite matrix $L \succ 0$ which satisfies the inequality $R(Z)\, L\, R(Z)^{\mathsf{T}} \preceq L$. For this aim, we define a matrix valued function $\Delta_L : \mathbb{M}^n \to \mathbb{S}^n$ as

$$\Delta_L(Z) := [I - Z]^m\, L^{-1} \left[I - Z^{\mathsf{T}}\right]^m - N(Z) L^{-1} N(Z)^{\mathsf{T}} .$$

Now, the condition $R(Z)^{\mathsf{T}}\, L\, R(Z) \preceq L$ is satisfied for all $Z \in \mathbb{M}^n$ if and only if we have $\Delta_L(Z) \succeq 0$ for all $Z \in \mathbb{M}^n$. This follows by applying an analogous argumentation as in the proof of statement 1. In the next step, we use the algebraic stability condition (15) which guarantees that there exist matrices $Q \in \mathbb{S}^m_+$ and $S \in \mathbb{R}^{m \times s}$ such that

$$\forall Z \in \mathbb{D}^n : \; \Delta_I(Z) = \mathcal{Z}_1 \left(Q \otimes I\right) \mathcal{Z}_1^{\mathsf{T}} - \sum_{i=1}^{s} \left(S_i \otimes I\right)^{\mathsf{T}} \mathcal{Z}_2 \left(Z + Z^{\mathsf{T}}\right) \mathcal{Z}_2^{\mathsf{T}} \left(S_i \otimes I\right).$$

Fortunately, as we can simply rescale the above condition with a positive definite matrix L, we find that we have

$$\Delta_L(Z) = \mathcal{Z}_1 \left(Q \otimes L^{-1}\right) \mathcal{Z}_1^{\mathsf{T}} - \sum_{i=1}^{s} \left(S_i \otimes I\right)^{\mathsf{T}} \mathcal{Z}_2 \left(Z L^{-1} + L^{-1} Z^{\mathsf{T}}\right) \mathcal{Z}_2^{\mathsf{T}} \left(S_i \otimes I\right)$$

for all $Z \in \mathbb{M}^n$ and for all $L \succ 0$. Recall that there exists for every $Z \in \mathbb{M}^n$ a matrix $L^{-1} \succ 0$ which satisfies $Z L^{-1} + L^{-1} Z^{\mathsf{T}} \preceq 0$. Thus, we may conclude that there exists for every $Z \in \mathbb{M}^n$ a matrix $L \succ 0$ with $\Delta_L(Z) \succeq 0$. This is equivalent to statement 2. □

For the special case that we are working with exact Jacobians $Z = hf'(x)$, the above Theorem is already sufficient to guarantee the stability of the sequence (x_k) which is generated by the method (12) when applied to a stable linear test equation. Although our main motivation is to work with Jacobian approximations, we summarize this observation in the following corollary:

Corollary 2 *Let the polynomial $\mathcal{P}$ satisfy the algebraic stability condition, let $Z = hf'(x)$ be the exact Jacobian approximation, and let $(x_k)_{k \in \mathbb{N}}$ be the sequence which is generated by the corresponding method* (12) *when applied to the linear test equation $\dot{y}(t) = Ay(t)$ with $y(0) = x_0$. Now, the following statements holds:*

1. *If the test equation is dissipative, i.e., if $A \in \mathbb{D}^n$, then the generated sequence is contractive, i.e., we have $\|x_{k+1}\|_2 \leq \|x_k\|_2$ for all $k \in \mathbb{N}$.*
2. *If we use a constant step size $h_k = h$ for all $k \in \mathbb{N}$ and if the test equation is stable, i.e., if we have $A \in \mathbb{M}^n$, then the generated sequence (x_k) is stable, too. I.e., there exists for all $\epsilon > 0$ a $\delta > 0$ such that $\|x_k\| \leq \epsilon$ for all x_0 with $\|x_0\| \leq \delta$ and for all $k \in \mathbb{N}$.*

Proof : As we consider exact Jacobians, the adjoint stability condition is trivially satisfied and we have $x_{k+1} = R(h_k A)x_k$. Thus, if we have $A \in \mathbb{D}^n$, the first statement of Theorem 3 implies

$$\|x_{k+1}\|_2 \;=\; x_k^\mathsf{T} R(h_k A)^\mathsf{T} R(h_k A) x \;\leq\; \|x\|_2 \;.$$

This is the first statement of the corollary. Similarly, if the step size is constant, we have

$$x_k \;=\; R(hA)^k x_0 \;.$$

Thus, the second statement of Theorem 3 implies that the sequence x_k remains stable, if we have $A \in \mathbb{M}^n$. This is due to the fact that the matrix power $R(hA)^k$ remains bounded for $k \to \infty$ as long as we have $\rho(R(hA)) \leq 1$. □

In the following, we have to come back to the case that Z is a low-rank Jacobian approximation. In order to analyze this case, we start with the following statement

Lemma 5 *Let Z satisfy the adjoint stability condition. If the right-hand side function f is dissipative such that $f' \in \mathbb{D}^n$, then we have $Z \in \mathbb{M}^n$.*

Proof : The adjoint stability condition guarantees the existence of a symmetric and positive semi-definite matrix $Z^* \in \mathbb{S}^n_+$ such that $Z = hZ^*f'$. Let $r \leq n$ be the rank of Z^*. As we have $Z^* \in \mathbb{S}^n_+$, there exists a matrix $U \in \mathbb{R}^{n\times r}$ with $Z^* = UU^\mathsf{T}$ and $U^\mathsf{T}U \in \mathbb{S}^r_{++}$. Now, we define the matrix $\hat{Z} := hU^\mathsf{T}f'U \in \mathbb{R}^{r\times r}$ which satisfies $\hat{Z} \in \mathbb{D}^n \subseteq \mathbb{M}^n$. The main idea of the proof is to show the equivalence

$$\hat{Z} \in \mathbb{M}^r \quad \Longleftrightarrow \quad Z \in \mathbb{M}^n \;. \tag{16}$$

Here, the argumentation is indirect: if we assume $Z = hUU^\mathsf{T}f' \notin \mathbb{M}^n$, then there exists a $h^* > 0$ for which the matrix $I - Z(h^*, x)$ is singular. On the other hand, the inverse

$$[I - Z(h,x)]^{-1} \;=\; I + hU\left(I - h\hat{Z}\right)^{-1} U^\mathsf{T} f'$$

cannot have a pole if we have $\hat{Z} \in \mathbb{M}^r$. As this argumentation works in both directions, we have the equivalence relation (16). Consequently, we have $Z \in \mathbb{M}^n$ which is the statement of the Lemma. □

Note that the above Lemma implies that we have $\rho(R(Z)) \leq 1$, if the adjoint stability assumption is satisfied and if we apply the integrator to a dissipative test equation. Unfortunately, this is not sufficient to guarantee the stability of the sequence (x_k) as the matrix Z might change from iteration to iteration. However, for the case the case that we restrict ourselves to linear test equations with symmetric negative definite matrices A, we can provide a proof of stability:

Theorem 4 *Let the polynomial $\mathcal{P}$ satisfy the algebraic stability condition and let the Jacobian Z satisfy the adjoint stability condition. Furthermore, let $(h_k)_{k\in\mathbb{N}}$ be an arbitrary sequence of steps sizes and let $(x_k)_{k\in\mathbb{N}}$ be the sequence which is obtained by applying the method (12) to the linear test equation $\dot{y}(t) = Ay(t)$. If $\bar{A} := -A \in \mathbb{S}^n_{++}$ is a symmetric and positive definite matrix, then the sequence (x_k) is stable, i.e., there exists for every $\epsilon > 0$ a $\delta > 0$ such that we have $\|x_k\| \leq \epsilon$ for all x_0 with $\|x_0\| \leq \delta$ and for all $k \in \mathbb{N}$.*

Proof : As the adjoint stability condition is satisfied, we have $Z = -hZ^*\bar{A}$ for some symmetric and positive semi-definite matrix $Z^* \in \mathbb{S}^n_+$. Thus, if we define $S := h\bar{A}^{\frac{1}{2}} Z^* \bar{A}^{\frac{1}{2}} \in \mathbb{S}^n_+$, we find the relations

$$\forall i \in \mathbb{N}: \quad \bar{A}^{\frac{1}{2}} Z^i = -S^i \bar{A}^{\frac{1}{2}} \quad \text{and} \quad \bar{A}^{\frac{1}{2}} \left[I - Z\right]^i = \left[I + S\right]^i \bar{A}^{\frac{1}{2}} \ .$$

As the reduced stability function R can be written as $R(Z) = [I-Z]^{-m}N(Z)$, we also have

$$\bar{A}^{\frac{1}{2}} R(Z) \ = \ R(-S)\bar{A}^{\frac{1}{2}} \quad \Longrightarrow \quad \bar{A}^{\frac{1}{2}} x^+ = R(-S)\bar{A}^{\frac{1}{2}} x$$

Now, the first statement of Theorem 3 can be applied, as we have $-S \in \mathbb{D}^n$. This, yields

$$\|\bar{A}^{\frac{1}{2}} x^+\|_2^2 \ = \ x^{\mathsf{T}} \bar{A}^{\frac{1}{2}} R(-S)^{\mathsf{T}} R(-S) \bar{A}^{\frac{1}{2}} x \ \leq \ \|\bar{A}^{\frac{1}{2}} x\|_2^2$$

which proves the stability of the sequence. □

5 Construction of Low-Rank Jacobian Approximations

The aim of this section is to discuss possible construction principles for the Jacobian approximation Z. Here, our focus is on low rank approximations of the form

$$Z(h,x) \ = \ hY(x)Y(x)^{\dagger} f'(x) \ ,$$

where $Y(x) \in \mathbb{R}^{n\times(d+1)}$ denotes a suitable low rank matrix and $Y(x)^{\dagger}$ its pseudo inverse. The idea is now to construct the matrix $Y(x)$ in such a way that we satisfy both the consistency conditions for Z as well as the adjoint stability condition. In the following, we will always assume that the first column of $Y(x)$ is equal to $f(x)$ such that we can employ the following technical result:

Lemma 6 *If the matrix $Y(x)$ has the form $Y(x) = (f(x)\,, Y_1(x), \ldots, Y_d(x))$ with $d < n$, then the matrix $Z := hY(x)Y(x)^{\dagger} f'(x)$ satisfies the adjoint stability condition.*

Proof : The matrix $Z^* := YY^{\dagger}$ is symmetric and positive semi-definite and satisfies by construction a relation of the form

$$Z^*Y \ = \ YY^{\dagger}Y \ = \ Y \ .$$

As the first column of Y is equal to f, we have $Z^* f = f$. This is equivalent to the adjoint stability condition. □

Note that the other rows $Y_1, \ldots, Y_d$ of the matrix Y can be used to impose the consistency conditions on the matrix Z. In the easiest case, we choose d equal to the number of directions which are required to obtain the consistency order of our choice. In order to understand this, it is helpful to consider the following example:

Example 4 Regard the polynomial $\mathcal{P}_3[M] = \frac{1}{6}\left(10M_1 - 5M_1^2 + M_1^3 + M_2\right)$ from Table 1 which yields a method with consistency order 3, if Z satisfies the two directional conditions $Zf = J_1 f$ and $Z^2 f = J_1^2 f$. Consequently, we can choose $d = 2$ defining

$$Y \;:=\; \left(f\,,\, f'f\,,\, f'f'f \right)$$

in order to satisfy both the consistency conditions of order 3 as well as the adjoint stability condition. Note that for an evaluation of $Y(x)$ we can use automatic differentiation. Here, we compute two forward directions of the Jacobian: first $f'f$ and then $f'(f'f)$. Additionally, it is easy to check that this particular choice of the function Y yields an integration algorithm which is invariant under orthogonal transformations of the differential equation. Here, the argumentation is analogous to Theorem 2.

In order to perform a step, we have to compute the matrix $M_1 = [I - Z]^{-1}$. Usually, we have to invert a $(n \times n)$-matrix to compute this term. However, if we employ for Z a low rank approximation of the above form, we can also compute M_1 by applying the Sherman-Morrison-Woodbury formula finding

$$M_1 \;=\; \left(I - hYY^\dagger f'\right)^{-1} \;=\; I + hY\left(1 - hY^\dagger f' Y\right)^{-1} Y^\dagger f' \,. \qquad (17)$$

There are two reasons why this re-formulation is useful: first, we only have to invert a $(d+1) \times (d+1)$ matrix for the computation of M_1. And second, we do not have to compute any adjoint derivatives of the Jacobian matrix f'. As we shall see below, this is due to the fact that we only have to evaluate the matrix-vector product $\mathcal{P}[M]f$, but never the matrix $\mathcal{P}[M]$ itself. In order to discuss how to obtain a numerically well-conditioned implementation of the whole evaluation procedure, we consider the following technical result:

Lemma 7 *Let us introduce the matrices $\hat{Z} := Y^\dagger f' Y$ and $\hat{M}_1 := (I - h\hat{Z})^{-1}$ assuming that Z and M_1 are constructed as explained above. Now, we have a general commutation rule of the form*

$$M_1^i Y \;=\; Y \hat{M}_1^i$$

which holds for all exponents $i \in \mathbb{N}$.

Proof : The main idea is to employ equation (17), which can in our notation be written as

$$M_1 \;=\; I + hY \left(I - h\hat{Z}\right)^{-1} Y^{\dagger} f' \; .$$

Multiplying this equation with the matrix Y from the right, we obtain the product

$$M_1 Y \;=\; Y \left(I + h(I - h\hat{Z})^{-1}\hat{Z}\right) \;=\; Y(I - h\hat{Z})^{-1} \;=\; Y\hat{M}_1 \; .$$

which proves the statement of the Lemma for $i = 1$. In the next step, we assume that the Lemma is correct for an integer valued exponent i and show that

$$M_1^{i+1} Y \;=\; M_1^i \, (M_1 Y) \;=\; M_1^i \left(Y\hat{M}_1\right) \;=\; Y\hat{M}_1^{i+1} \; .$$

Consequently, the statement of the Lemma follows by induction over i. □

As our aim is to evaluate the matrix-vector product $\mathcal{P}[M]f$ which is needed in the integrator step (12), we first write the polynomial $\mathcal{P}$ in the form

$$\mathcal{P}[M] \;=\; \mathcal{P}_{\rm lin}[M_1] + \mathcal{P}_{\rm nonlin}[M] \qquad \text{with} \qquad \mathcal{P}_{\rm lin}[M_1] \;:=\; \sum_{i=1}^{m} \beta_i M_1^i \; .$$

We start with the evaluation of the term $\mathcal{P}_{\rm lin}[M]f$. For this aim we can use the following Corollary which follows immediately from the commutation rule from Lemma 7:

Corollary 3 *Let the assumptions from Lemma 6 and 7 be satisfied. Moreover, we denote with* $e_1 = (1, 0, \ldots, 0)^{\mathsf{T}} \in \mathbb{R}^{d+1}$ *the first unit vector. Now, we have*

$$\mathcal{P}_{\rm lin}[M_1] f \;=\; Y \left(\sum_{i=1}^{m} \beta_i \hat{M}_1^i \right) e_1 \; .$$

Proof : The main idea is to use the commutation rule from Lemma 7 which can be applied to all summands in the polynomial $\mathcal{P}_{\rm lin}[M_1]$. This yields the relation

$$\mathcal{P}_{\rm lin}[M_1] Y \;=\; \sum_{i=1}^{m} \beta_i M_1^i Y \;=\; Y \left(\sum_{i=1}^{m} \beta_i \hat{M}_1^i \right) \; .$$

Now, we only have to recall that the first column of Y is by construction equal to f (cf. Lemma 6). Thus, we conclude the statement of the Corollary. □

Note that the computational cost for an evaluation of the polynomial term

$$Y \left(\sum_{i=1}^{m} \beta_i \hat{M}_1^i \right) e_1$$

is of order $\mathbf{O}(n)$, as the size of the matrix $\hat{M}_1$ does not depend on n. In order to also evaluate the product $\mathcal{P}_{\rm nonlin}[M]f$, we employ again the commutation rule

$M_1^i Y = Y \hat{M}_1^i$, but we have to additionally use automatic differentiation to evaluate the higher order directional derivatives. For example, if $\mathcal{P}_{\text{nonlin}}[M]f$ contains a monomial term of the form $M_2 M_1 f$, we would write this term in the form

$$\begin{aligned} M_2 M_1 f = h^2 M_1 f''(M_1 f) M_1^2 f &= M_1 f''(Y\hat{M}_1 e_1)(Y\hat{M}_1^2 e_1) \\ &= d_1 + hY\hat{M}_1 Y^\dagger d_2 \,, \end{aligned}$$

where $d_1 = f''(Y\hat{M}_1 e_1)(Y\hat{M}_1^2 e_1)$ is a second order directional derivative and $d_2 = f' d_1$ is a first order directional derivative which can both be computed by automatic differentiation.

6 Step Size Control and Implementation Details

Note that step size control algorithms for numerical integration schemes typically proceed as follows [18]: we first compute the step x_N^+ with an integration scheme of order N. After this, we compute another step x_{N+1}^+ using an integration scheme of order $N+1$. Now, we can use a norm suitable norm $\|\epsilon\|$ of the difference $\epsilon := x_{N+1}^+ - x_N$ between the two steps as an error estimate with local order $N+2$. In the standard framework [9], we choose a tolerance $\text{TOL} > 0$ and absolute tolerances $\text{ATOL}_i > 0$ defining the local error estimate as

$$\|\epsilon\|^2 := \frac{1}{n} \sum_i \left(\frac{\epsilon_i}{\text{TOL}\,|x_i| + \text{ATOL}_i} \right)^2 .$$

As the term $\text{TOL}\,|x_i| + \text{ATOL}_i$ is in the order of the given tolerances, we consider the error $\|\epsilon\|$ small, if we have $\|\epsilon\| \leq 1$. If this is not case, we do not accept the step but repeat the procedure with a smaller step-size

$$h := \max \left\{ \rho_1 h \sqrt[N+1]{1/\|\epsilon\|} \,,\, h_{\min} \right\} ,$$

where $0 < \rho_1 \leq \rho_2 \leq 1$ are some heuristic constants. Otherwise, if the step is accepted, a new step-size is computed as

$$h^+ := \min \left\{ \rho_2 h \sqrt[N+1]{1/\|\epsilon\|} \,,\, h_{\max} \right\} .$$

In the following we denote with $\mathcal{P}_N[M]$ and $\mathcal{P}_{N+1}[M]$ polynomials which satisfy the consistency conditions for order N and $N+1$, respectively. We give the corresponding numerical method the name "Integrator $\mathcal{P}_N/\mathcal{P}_{N+1}$". The local error estimate can then be evaluated by defining

$$\epsilon := h\mathcal{D}[M]f \qquad \text{with} \qquad \mathcal{D}[M] := \mathcal{P}_{N+1}[M] - \mathcal{P}_N[M] \,.$$

Note that that there is in this form nothing special about this step size control technique. However, if we tailor the outlined strategy to our method, we can re-use certain terms and derivative directions such that the joint evaluation of the terms $\mathcal{P}_N[M]f$ and $\mathcal{D}[M]f$ can be implemented efficiently. In order to

Algorithm 1: The Implementation Details for the Tutorial Integrator $\mathcal{P}_3/\mathcal{P}_4$.

Input:

a) Choose a time horizon $[0, T]$, set $x = x_0$ and $t = 0$.
b) Choose the absolute and relative tolerances ATOL and TOL.
c) Choose a feasible step size range $[h_{\min}, h_{\max}]$ and tuning constants $0 < \rho_1 \leq \rho_2 \leq 1$.

Integration Loop:

1) Evaluate $d_1 := f(x)$ and use automatic differentiation to compute

$$d_2 := f'd_1 \ , \ d_3 := f'd_2 \ , \ d_4 := f'd_3 \ , \ d_5 := f'd_4 \ ,$$
$$d_6 := f''d_1^2 \ , \ d_7 := f'd_6 \ , \ d_8 := f'd_7 \ , \text{ and } \ d_9 := f'''d_1^3 \ .$$

Define $Y := \left(d_1, \frac{d_2}{\|d_2\|}, \frac{d_3}{\|d_3\|}, \frac{d_4}{\|d_4\|}, d_7 \right)$ and $\hat{Y} := f'Y = (d_2, \frac{d_3}{\|d_2\|}, \frac{d_4}{\|d_3\|}, \frac{d_5}{\|d_4\|}, d_8)$.
2) Compute the (5×5)-matrix $\hat{Z} := Y^\dagger \hat{Y}$.
3) Compute the matrix $\hat{M}_1 = \left(I - h\hat{Z} \right)^{-1}$.
4) Compute the remaining derivative directions

$$d_{10} := f'' \left(hY\hat{M}_1 e_1 \right)^2 \ , \ \ d_{12} := f''' \left(hY\hat{M}_1 e_1 \right)^3 \ ,$$
$$d_{11} := f'' \left(hY\hat{M}_1 e_1 \right) \left(hY \left[\tfrac{1}{4}\hat{M}_1^2 - \hat{M}_1 \right] e_1 \right) \ ,$$
$$\text{and} \quad (d_{13}, d_{14}, d_{15}) := (d_{10}, d_{11}, d_{12}) + hY\hat{M}_1 Y^\dagger f'(d_{10}, d_{11}, d_{12}) \ .$$

5) Compute the local error estimate $\|\epsilon\|^2 := \frac{1}{n} \sum_i \left(\frac{\epsilon_i}{\mathrm{TOL}\,|x_i| + \mathrm{ATOL}_i} \right)^2$ with

$$\epsilon := \frac{h}{24} Y \left(-\hat{M}_1 + 3\hat{M}_1^2 - 3\hat{M}_1^3 + \hat{M}_1^4 \right) e_1 + \frac{h}{6} d_{14} + \frac{h}{24} d_{15} \ .$$

The evaluation of the polynomial in $\hat{M}_1$ can, e.g., be based on Horner's scheme.
6) If we have $\|\epsilon\| > 1$ there are two possibilities: if we have $h = h_{\min}$, we have to terminate with an error message. Otherwise, we set $h := \max \left\{ \rho_1 h \sqrt[4]{1/\|\epsilon\|} \, , \, h_{\min} \right\}$ and continue with step 3).
7) Compute the next iterate x^+ using the equation

$$x^+ = x + \frac{h}{6} Y \left(10\hat{M}_1 - 5\hat{M}_1^2 + \hat{M}_1^3 \right) e_1 + \frac{h}{6} d_{13} \ .$$

The evaluation of the polynomial in $\hat{M}_1$ can again be based on Horner's scheme.
8) Set $h^+ := \min \left\{ \rho_2 h \sqrt[4]{1/\|\epsilon\|} \, , \, T - t \, , \, h_{\max} \right\}$ and terminate successfully if $h^+ = 0$. Otherwise, set $x := x^+$, $t := t + h$, $h := h^+$, and continue with step 1).

discuss this it is helpful to consider an example. In this paper, we work out the details for Integrator $\mathcal{P}_3/\mathcal{P}_4$ which can be used as a tutorial. Recall that the polynomials $\mathcal{P}_3$ and $\mathcal{P}_4$ are listed in Table 1. A corresponding evaluation procedure can be found in Algorithm 1. In the following, we briefly go through the steps of Algorithm 1 elaborating some aspects about this code. Although the corresponding remarks are tailored for our tutorial integrator, they transfer analogously also for other orders or variants of the proposed integration scheme.

Remarks on Algorithm 1

- We compute the minimum number of derivative directions which are needed to guarantee first that the method $x^+ = x + h\mathcal{P}_4[M]f$ is consistent with order 4 and second that the adjoint stability condition is satisfied. For the actual integration using the polynomial $\mathcal{P}_3$ we would need less directions to achieve the consistency order 3. However, as we have to compute the directional derivatives for the order 4 polynomial anyhow, we use them all within the matrix Y to obtain a Jacobian approximation.
- Note that the matrix Y contains the directions in which the Jacobian approximation is required to be exact. Here, it is important to use a proper scaling. For example the derivative term $d_4 = f'f'f'f$ might contain some large components, if the matrix f' is ill-conditioned. Thus, it is advisable to scale this direction, e.g., with its own norm.
- Note that the computation of the matrix $\hat{Z}$ is of order $\mathbf{O}(n)$ and can be computed efficiently. However, in practice suitable regularizations should be used to compute the pseudo-inverse $Y^\dagger$. In this context, it is also helpful to note that $\hat{Z}$ can be written as the solution of a least squares optimization problem
$$\hat{Z} \;=\; \underset{Z}{\operatorname{argmin}} \left\| YZ - \hat{Y} \right\|_{\mathrm{F}}^2 \;,$$
since there exist efficient linear algebra codes (such as LAPACK) which solve this type of matrix least-squares problems directly accepting Y and $\hat{Y}$ as inputs.
- The expression for the local error can be derived by working out the difference polynomial $\mathcal{P}_4[M] - \mathcal{P}_3[M]$. Note that all computations in the steps of the algorithms, including the steps 5 and 7, are of order $\mathbf{O}(n)$. This is due to fact that the matrix $\hat{M}_1$ is – independent of n – always a simple 5×5 matrix.
- In step 6 it can happen that a step is rejected. Fortunately, we only have to go back to step 3 in this case. Thus, neither the vectors $d_1, d_2, \ldots, d_9$ nor the matrix $\hat{Z}$ have to be recomputed after a step rejection.

6.1 Extensions for Non-Autonomous Systems

In the previous section, we have always assumed that the right-hand side f does not explicitly depend on the time t. Note that it is straightforward to extend our framework for this case as we can always reformulate a differential equation into an autonomous system by regarding the time t as an additional differential state which satisfies the trivial differential equation $\dot{t} = 1$ with the initial value $t(0) = t_0$. This leads to a method of the form
$$\begin{pmatrix} x^+ \\ t^+ \end{pmatrix} \;=\; \begin{pmatrix} x \\ t \end{pmatrix} \;+\; h\mathcal{P}[M] \begin{pmatrix} g(t,z) \\ 1 \end{pmatrix} \;.$$
Being at this point, we would like to construct the Jacobian approximation in such a way that the update for t is of the form $t^+ = t + h$. In order to impose

this, we have to keep the particular block structure of the matrix M_1, i.e., we define

$$M_1 \;:=\; \begin{pmatrix} I - Z & -hg_t \\ 0 & 1 \end{pmatrix}^{-1} \;=\; \begin{pmatrix} (I-Z)^{-1} & h\,(I-Z)^{-1}\,g_t \\ 0 & 1 \end{pmatrix} .$$

Here, we only introduce the Jacobian approximation $Z \approx hg_x$ while the derivative g_t of g with respect to t is anyhow cheaply available, as it can be computed by automatic differentiation. As the last row of the higher order derivative matrices M_2, M_3, ..., M_N are all zero such that the update for t becomes

$$t^+ \;=\; t + h \sum_{i=1}^{m} \beta_i \;,$$

where the constants β_i are the coefficients in the polynomial $\mathcal{P}$ which correspond to the matrix powers M_1, M_1^2, ..., M_1^m. If we assume that the polynomial $\mathcal{P}$ is consistent, we must have $\sum_i \beta_i = 1$ and thus the update for the time t is by construction of the form $t^+ = t + h$. Finally, the derivative directions in which the matrix Z should be exact can again be obtained by a comparison of coefficients with the Taylor expansion of the exact solution.

7 Numerical Examples

In this section we present two numerical case studies. First, we consider in Section 7.1 a discretized PDE system comparing the proposed integrator $\mathcal{P}_3/\mathcal{P}_4$ from Algorithm 1 with both a state of the art explicit Runge-Kutta method and a state of the art BDF method. And second, we test our algorithm in Section 7.2 with a stiff and nonlinear test example.

7.1 A Two-Dimensional Advection-Diffusion Equation

The first case study considers a semi-discretized version of a two-dimensional advection-diffusion equation of the form

$$\frac{\partial}{\partial t} v(t, x_1, x_2) \;=\; \frac{\partial^2 v(t, x_1, x_2)}{\partial x_1^2} + \frac{1}{2} \frac{\partial v(t, x_1, x_2)}{\partial x_1} + \frac{\partial^2 v(t, x_1, x_2)}{\partial x_2^2} \;. \quad (18)$$

This differential equation is regarded on the domain $(x_1, x_2) \in [0, 1] \times [0, 2]$ in combination with zero Dirichlet boundary conditions. The simulation interval is $t \in [0, 1]$ while the initial profile is given by

$$v(0, x_1, x_2) \;=\; x_1 x_2 (1 - x_1)(2 - x_2) \exp(5 x_1 x_2) \;.$$

Note that this numerical case study is exactly coinciding with the tutorial advection-diffusion example which comes with the software package `SUNDIALS` [29,30]. For the spatial discretization we use a $(N + 2) \times (N + 2)$ mesh in

	RK 4/5	BDF (dense)	BDF (sparse)	INT $\mathcal{P}_3/\mathcal{P}_4$
$n_x = 100$	59 ms	10 ms	5 ms	4 ms
$n_x = 400$	525 ms	154 ms	29 ms	12 ms
$n_x = 2500$	10.8 s	13.2 s	0.77 s	0.13 s
$n_x = 10000$	$\approx$ 3 min	$\approx$ 30 min	7.8 s	1.1 s

Table 2 A run-time comparison of several integration methods for the discretized version of the PDE problem (18) with $N = 10$, $N = 20$, $N = 50$, and $N = 100$. The first column corresponds to the run-time of an explicit Runge-Kutta method of order 4/5 (Dormand-Prince). The run-times in the second and third column have been obtained with a BDF solver from `SUNDIALS`, named `CVODE`, using once a dense and once a direct banded solver within the Newton method. The last column reports the run-times which have been obtained with the proposed integrator $\mathcal{P}_3/\mathcal{P}_4$ from Algorithm 1.

combination with standard central differences which leads to a stiff ordinary differential equation with $n_x = N^2$ states.

In order to assess how the proposed algorithm performs in comparison to existing methods, we test the following implementations:

- Firstly, we test a standard explicit Runge-Kutta method of nominal order 4 and step-size control order 5. Here, we use a Butcher tableau from Dormand and Prince [18]. The corresponding code is implemented by using the programming language C/C++.
- Secondly, we employ the `SUNDIALS` solver `CVODE` (version 2.4.0) with user supplied Jacobian. Note that this code is based on a state of the art BDF method [30]. Here, we test both a dense linear algebra routine for Jacobian decomposition as well as a direct sparse solver.
- Thirdly, we implement the integrator integrator $\mathcal{P}_3/\mathcal{P}_4$ from Algorithm 1. Similar to the implementations of the existing methods our implementation is based on the programming language C/C++. The linear algebra operations in step 2) and 3) of Algorithm 1 are implemented by using the `LAPACK` routines "`dgels`" and "`dgetrf/dgetri`" (version 3.4.0), respectively.

All methods have been tested with a local error tolerance of TOL $= 10^{-6}$. The corresponding run-times for simulating the above advection-diffusion problem are reported in Table 2. Here, all computations have been obtained by running the corresponding implementations on a modern Laptop (Intel Pentium, 1.5GHz).

Let us first discuss the run-time of the explicit Runge-Kutta method. Unfortunately, the run-time of this solver does not increase proportionally with the system size $n_x = N^2$. This is due to the fact that increasing N does not only lead to a larger number of differential states, but also to a more stiff dynamics. For example, for $N = 10$ the Runge-Kutta integrator needs 360 steps to integrate the system. For $N = 20$ we need already 716 steps. Finally, for $N = 100$ the explicit Runge-Kutta integrator needs $1.2 * 10^4$ steps leading to a poor run-time performance ($\approx$ 3 min).

In contrast to the Runge-Kutta integrator, the BDF method needs always between 100 and 200 steps. However, if we use a dense linear algebra solver

the method scales even worse than the Runge-Kutta method. This is due to the fact that the cost for decomposing the Jacobian matrix scales with order $\mathbf{O}(N^6)$. Clearly, the Jacobian matrix of the semi-discretized advection diffusion equation is sparse and banded. Hence, it is not surprising that the performance of the BDF code `CVODE` is much better, if we link a suitable sparse linear algebra solver. The run-time of such a sparsity exploiting BDF code can be expected to increase linearly with the number n_x of differential states, as it can also approximately be observed from the run-time measurements in Table 2.

Finally, we test the proposed integrator $\mathcal{P}_3/\mathcal{P}_4$ which takes between 100 and 200 steps. In general, the integrator $\mathcal{P}_3/\mathcal{P}_4$ shows in comparison to the to the sparse BDF code a similar run-time performance. For increasing N we can even observe that the integrator $\mathcal{P}_3/\mathcal{P}_4$ is slightly faster than the BDF solver. This is mainly due to fact that the cost per step is in our implementation smaller than the corresponding cost of the BDF integrator. Note that this run-time performance is quite remarkable in the sense that the integrator $\mathcal{P}_3/\mathcal{P}_4$ does not exploit any structure or sparsity. This generic black-box integration is possible as the proposed scheme is purely explicit and does not require us to invert any large matrices. This is in contrast to state of the art BDF or implicit Runge-Kutta implementations which evaluate and decompose the full Jacobian of the right-hand side function at least once per simulation interval.

7.2 A Stiff Nonlinear Test Example

In this section we consider a nonlinear differential equation which originates from plant physiology and which has originally been introduced in [41]. The model is known to be stiff:

$$\begin{aligned}
\dot{x}_1 &= -1.71x_1 + 0.43x_2 + 8.32x_3 + 0.0007 \\
\dot{x}_2 &= 1.71x_1 - 8.75x_2 \\
\dot{x}_3 &= -10.03x_3 + 0.43x_4 + 0.035x_5 \\
\dot{x}_4 &= 8.32x_2 + 1.71x_3 - 1.12x_4 \\
\dot{x}_5 &= -1.745x_5 + 0.43x_6 + 0.43x_7 \\
\dot{x}_6 &= -280x_6x_8 + 0.69x_4 + 1.71x_5 - 0.43x_6 + 0.69x_7 \\
\dot{x}_7 &= 280x_6x_8 - 1.81x_7 \\
\dot{x}_8 &= -280x_6x_8 + 1.81x_7 \ . \qquad (19)
\end{aligned}$$

Here, the simulation interval is chosen to be $[0, 1000]$ while the initial value is given by

$$x(0) \ = \ (\,1,\,0,\,0,\,0,\,0,\,0,\,0,\,0.0057\,)^{\mathsf{T}} \ .$$

If we implement the above simulation problem with the Runge-Kutta solver of order 4 and step-size control order 5, we need 9582 integrator steps leading to a runtime of 98 ms. In contrast to this, the BDF solver needs 287 steps leading to a run-time of 2.4 ms. Here, we use a dense linear algebra routine,

as the system has only 8 states. Finally, the integrator $\mathcal{P}_3/\mathcal{P}_4$ needs 319 steps and 2.2 ms. In this example, we cannot observe a major difference between the run-times for the BDF and the integrator $\mathcal{P}_3/\mathcal{P}_4$, but the proposed explicit method performs again better than an explicit Runge Kutta integrator. Here, it is interesting to remark that also for a highly nonlinear and stiff dynamic equation our explicit method is able to simulate the system reliably, although our theoretical stability considerations apply to linear systems only.

8 Conclusions and Outlook

In this paper we have analyzed a novel class of nonlinear explicit integrators for stiff differential equations. Here, we have employed low-rank Jacobian approximations which are required to satisfy an adjoint directional condition which has been introduced in Definition 1. This adjoint stability condition allowed us to provide a proof of the stability of the integration scheme with respect to all linear-symmetric and dissipative test systems as summarized in Theorem 4. The main advantages in comparison to implicit integration schemes are first that we do not have to solve any implicit equations and second that the Jacobian approximations have to be accurate in a few directions only while they may be loose in other directions. Finally, we have provided an efficient implementation of the integration algorithm. Numerical tests with a discretized advection-diffusion equation as well as with a nonlinear stiff system have shown a promising performance of the method in comparison to both state of the art explicit Runge-Kutta and state of the art implicit BDF methods.

Future research might investigate the stability properties of the method for stable but possibly non-dissipative or even nonlinear test equations which was beyond the scope of this paper. Due to the fact that the method can easily be implemented, we expect many interesting applications. For example, the method might be suitable for multiple-shooting based optimal control algorithms, as the one-step nature of the integration scheme does not require any starting phase or initial Jacobian decomposition which is typically needed in implicit multi-step or Runge-Kutta methods. Finally, for frozen step-size meshes the memory requirements of the method are known in advance which allows, e.g., the automatic generation of static simulation code. This might be interesting for many engineering applications with embedded hardware systems such as autogenerated dynamic optimization algorithms for nonlinear model predictive control [31].

Acknowledgements The research was supported by the Research Council KUL via GOA/ 11/05 Ambiorics, GOA/ 10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES-4CHEM and PhD/postdoc/fellow grants, the Flemish Government via FWO (PhD/postdoc grants, projects G0226.06, G0321.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, G.0588.09, G.0377.09, research communities ICCoS, ANMMM, MLDM) and via IWT (PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare), the Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011), the IBBT, the EU

(ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC HIGHWIND (259 166)), the Contract Research (AMINAL), the Helmholtz Gemeinschaft via viCERP and the ACCM.

References

1. J. Albersmeyer. *Adjoint-based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems.* Ph.D. Thesis, Ruprecht-Karls-Universität Heidelberg, 2010.
2. U.M. Ascher and L.R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations.* SIAM, Philadelphia, 1998.
3. D. Barton, I.M. Willers, and R.V.M. Zahar. *The automatic solution of systems of ordinary differential equations by the method of Taylor series.* The Computer Journal, 14:243–248, 1971.
4. D. Barton. *On Taylor series and stiff equations.* ACM Trans. Math. Software 6:280–294, 1980.
5. F. Bashforth and J. C. Adams. *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid, with an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops.* Cambridge University Press, Cambridge, 1883.
6. I. Bauer. *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik.* PhD thesis, Universität Heidelberg, 1999.
7. H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations.* Birkhäuser, Boston, 1983.
8. J.P. Boyd. *Chebyshev and Fourier Spectral Methods.* Dover, New York, second edition, 2001.
9. K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1996.
10. K. Burrage and J.C. Butcher. *Stability criteria for implicit RungeKutta methods.* SIAM J. Numer. Anal. 16:46–57, 1979 .
11. J.C. Butcher. *A stability property of implicit RungeKutta methods* BIT 15:358–361, 1975.
12. Y. F. Chang and G. Corliss. *ATOMFT: solving ODEs and DAEs using taylor series.* Computers Math. Applic., 28:209–233, 1994.
13. G.F. Corliss and G. Kirlinger. *On implicit Taylor series methods for stiff ODEs.* Computer Arithmetic and Enclosure Methods, pp:371–379, Amsterdam, 1992.
14. S.M. Cox and P.C. Matthews. *Exponential Time Differencing for Stiff Systems*, J. Comput. Phys., 176:430–455, 2002.
15. C.F. Curtiss and J.O. Hirschfelder. Integration of stiff equations. *Proc. Nat. Acad. Sci*, 38:235–243, 1952.
16. G. Dahlquist. Convergence and stability in numerical integration of ordinary differential equations. *Math. Scand.*, 4:33–53, 1956.
17. G. Dahlquist. *Error analysis of a class of methods for stiff nonlinear initial value problems* Numerical Analysis, Dundee, Lecture Notes in Mathematics 506:60–74, 1976.
18. J.R. Dormand and P.J. Prince. *A family of embedded RungeKutta formulae.* J. Comput. Appl. Math., 6 (1980), 1926.
19. K. Eriksson, C. Johnson, and A. Logg. *Explicit Time Stepping for Stiff ODEs*, SIAM Journal of Scientific Computing, Vol. 25(4):1142–1157, 2003.
20. L. Euler. *De integratione aequationum differentialium per approximationem. De integratione aequationum differentialium per approximationem, In Opera Omnia*, volume 11. Teubner, Leipzig and Berlin, 1913.
21. C.W. Gear. *Numerical initial value problems in ordinary differential equations.* Prentice Hall, Englewood Cliffs, NJ, 1971.

22. C.W. Gear and I.G. Kevrekidis. *Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum.* SIAM J. Sci. Comp, 2001.
23. A. Griewank. *ODE Solving via Automatic Differentiation and Rational Prediction.* In Numerical Analysis 1995, Pitman Research Notes in Mathematics Series, D.F. Griffiths, G.A. Watson (eds.), Vol. 344, Addison-Wesley Longman Ltd, 1995.
24. A. Griewank. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.
25. E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations*, volume I of *Springer Series in Computational Mathematics.* Springer, Berlin, 2nd edition, 1993.
26. E. Hairer and G. Wanner. RADAU5 - an implicit Runge-Kutta code. Report, Université de Genève, Dept. de mathématiques, Genève, 1988.
27. E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems.* Springer, Berlin Heidelberg, 1991.
28. D. Henrion and J.B. Lasserre. *GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi.* ACM Transactions on Mathematical Software, 29(2):165–194, 2003.
29. A.C. Hindmarsh. LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM-SIGNUM Newsletter*, 15:10–11, 1980.
30. A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, and C.S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31:363–396, 2005.
31. B. Houska, H.J. Ferreau, and M. Diehl. *An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range*, Automatica, 47(10):2279–2285, 2011.
32. J.D. Lambert. *Nonlinear methods for stiff systems of ordinary differential equations*, Proceedings of the Conference on Numerical Solution of Ordinary Differential Equations 363, University of Dundee, pp: 75–88, 1973.
33. J.B. Lasserre. *Moments, Positve Polynomials and Their Applications*, Imperial College Press, 2009.
34. W. Kutta. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen.* Z. Math. Phys., 46:435–453 (1901).
35. R. E. Moore. Interval Arithmetic and Automatic Error Analysis in Digital Computing. PhD thesis, Department of Computer Science, Stanford University, 1962.
36. F.R. Moulton. *New Methods in Exterior Ballistics.* University of Chicago Press, 1926.
37. W.E. Milne. *A note on the numerical integration of differential equations.* J. Res. Nat. Bur. Stand., 43:537–542, 1949.
38. L.R. Petzold. A Description of DASSL: A Differential-Algebraic System Solver. In *Proc. 10th IMACS World Congress*, Montreal, 1982.
39. J.R. Rice. Mathematical Software, Chapter: *Taylor Series Methods for Ordinary Differential Equations – An Evaluation* (369–390) Academic Press, New York, 1971.
40. C. Runge. *Über die numerische Auflösung von Differentialgleichungen.* Math. Ann., 46:167–178, 1895.
41. E. Schäfer. *A new approach to explain the high irradiance responses of photomorphogenesis on the basis of phytochrome.* J. Math. Biol., 2:41–56, 1975.
42. R. Scherer and H. Türke. *Algebraic Characterization of A-Stable Runge-Kutta Methods.* J. Applied Numerical Mathematics, 5:133–144, 1989.
43. J.G. Verwer. *Explicit Runge-Kutta methods for parabolic partial differential equations.* Applied Numerical Mathematics, 22:359–379,1996.
44. M. Grant and S. Boyd. *CVX: Matlab software for disciplined convex programming*, http://stanford.edu/ boyd/cvx, 2009.