Construction of Adaptive Multistep Methods for Problems with Discontinuities, Invariants, and Constraints

Mohammadi, Fatemeh

2018

Document Version:
Publisher's PDF, also known as Version of record

Construction of Adaptive Multistep Methods for Problems with Discontinuities, Invariants, and Constraints

by Fatemeh Mohammadi

LUND
UNIVERSITY

Academic thesis which, with due permission of the Faculty of Science at Lund University, will be publicly defended on Friday, the 28th of September 2018 at 10:00 in lecture hall MH:Gårding at the Center for Mathematical Sciences, Sölvegatan18, Lund, for the degree of Doctor of Philosophy in Numerical Analysis.

Faculty opponent:
Prof. Ewa B. Weinmüller, Vienna University of Technology, Austria.

Organization
LUND UNIVERSITY

Center for Mathematical Sciences
Numerical Analysis
Box 118
221 00 Lund, Sweden

Document name
DOCTORAL DISSERTATION IN MATHEMATICAL SCIENCES

Date of disputation
2018-09-28

Sponsoring organization

Author(s)
Fatemeh Mohammadi

Title and subtitle Construction of Adaptive Multistep Methods for Problems with Discontinuities, Invariants, and Constraints

Abstract

Adaptive multistep methods have been widely used to solve initial value problems. These ordinary differential equations (ODEs) may arise from semi-discretization of time-dependent partial differential equations (PDEs) or may combine with some algebraic equations to represent a differential algebraic equations (DAEs).

In this thesis we study the initialization of multistep methods and parametrize some well-known classes of multistep methods to obtain an adaptive formulation of those methods. The thesis is divided into three main parts; (re-)starting a multistep method, a polynomial formulation of strong stability preserving (SSP) multistep methods and parametric formulation of $\beta-$blocked multistep methods.

Depending on the number of steps, a multistep method requires adequate number of initial values to start the integration. In the view of first part, we look at the available initialization schemes and introduce two family of Runge–Kutta methods derived to start multistep methods with low computational cost and accurate initial values. The proposed starters estimate the error by embedded methods.

The second part concerns the variable step-size $\beta-$blocked multistep methods. We use the polynomial formulation of multistep methods applied on ODEs to parametrize $\beta-$blocked multistep methods for the solution of index-2 Euler-Lagrange DAEs. The performance of the adaptive formulation is verified by some numerical experiments.

For the last part, we apply a polynomial formulation of multistep methods to formulate SSP multistep methods that are applied for the solution of semi-discretized hyperbolic PDEs. This formulation allows time adaptivity by construction.

Key words
Multistep methods, Initialization, Discontinuity, Time adaptivity, Strong stability preserving, Differential algebraic equations, $\beta-$blocking

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title
1404-0034

ISBN
978-91-7753-786-1 (print)
978-91-7753-787-8 (pdf)

Recipient's notes

Number of pages
162

Price

Security classification

Signature _____ Date 2018-08-24

Construction of Adaptive Multistep Methods for Problems with Discontinuities, Invariants, and Constraints

by Fatemeh Mohammadi

LUND
UNIVERSITY

Faculty of Science
Center for Mathematical Sciences
Numerical Analysis

Cover illustration front: The photo illustrates a stairway (multistep) of Persepolis, located in Iran, and shows a procession of people bringing tribute to the Achaemenid king (Copyright belongs to Gitty images).

Numerical Analysis
Center for Mathematical Sciences
Lund University
SE-221 00 Lund
Sweden
http://www.maths.lu.se

Doctoral Theses in Mathematical Sciences 2018:7
ISSN: 1404-0034

ISBN: 978-91-7753-786-1 (print)
ISBN: 978-91-7753-787-8 (pdf)
LUNFNA-1008-2018

Printed in Sweden by Media-Tryck, Lund University, Lund 2018

Dedicated to my parents
&
my beloved Toheed

‘ Not everything that counts
can be counted and
not everything that’s counted
truly counts ’

William Bruce Cameron

توانا بود هر که دانا بود

ز دانش دل پیر برنا بود

فردوسی

Knowledge is power

Ferdosi

Abstract

Adaptive multistep methods have been widely used to solve initial value problems. These ordinary differential equations (ODEs) may arise from semi-discretization of time-dependent partial differential equations (PDEs) or may combine with some algebraic equations to represent a differential algebraic equations (DAEs).

In this thesis we study the initialization of multistep methods and parametrize some well-known classes of multistep methods to obtain an adaptive formulation of those methods. The thesis is divided into three main parts; (re-)starting a multistep method, a polynomial formulation of strong stability preserving (SSP) multistep methods and parametric formulation of $\beta-$blocked multistep methods.

Depending on the number of steps, a multistep method requires adequate number of initial values to start the integration. In the view of first part, we look at the available initialization schemes and introduce two family of Runge–Kutta methods derived to start multistep methods with low computational cost and accurate initial values. The proposed starters estimate the error by embedded methods.

The second part concerns the variable step-size $\beta-$blocked multistep methods. We use the polynomial formulation of multistep methods applied on ODEs to parametrize $\beta-$blocked multistep methods for the solution of index-2 Euler-Lagrange DAEs. The performance of the adaptive formulation is verified by some numerical experiments.

For the last part, we apply a polynomial formulation of multistep methods to formulate SSP multistep methods that are applied for the solution of semi-discretized hyperbolic PDEs. This formulation allows time adaptivity by construction.

Acknowledgements

It is a pleasure to thank those who made this thesis possible. It is unbelievable how time flies. Six years ago I took the multibody dynamics course with Claus and that became the beginning of the new chapter in my life. I got the honor to be advised and accompanied by my academic parents, Carmen Arévalo and Claus Führer during the last five years of my life. I would like to thank my advisors for their supports, asking insightful questions, and offering invaluable advices. Carmen you are much more than an advisor for me and our meetings never feel like meetings. Our meetings feel like friends sharing things such as advice, experiences and stories.

Also, I owe my deepest gratitude to Gustaf Söderlind for his time, support, and patience. I would try to keep the logarithmic norm of my future career negative.

As I move through this chapter of my life. I feel so blessed for working in the numerical analysis group where colleagues care so much for one another. I want to express my deep appreciation to all my colleagues especially, Azahar, Dara, Tony, Erik, Christian, Lea, Peter and Julio for the help and support you have offered during these years. Also I would like to give a special thank to my friend Annika for her true friendship.

I am eternally grateful and blessed to have been gifted such a wonderful parents, Bashir and Akram. I have achieved so much and thanks to you, I can look for a bright future. Thanks to my siblings, Nasibeh and Hamidreza for showering me with their love. I also wish to thank my aunt, Zohreh for all her kindness and emotional support.

Last but not least, I cannot express my unfailing gratitude and love to my dear husband Toheed for his continuous supports, encouragements and invaluable assistance.

As I close this chapter, I would like to thank all my friends who supported me in the ups and downs of this journey, especially Najmeh, Masoomeh, Sara, and my KNTU friends.

Fatemeh Mohammadi
September 2018

Popular summary

The wide variety of physical phenomena, such as motion of objects, reaction among chemical substances, electricity flow in a circuit can be described by equations with quantities that vary along time. The rate at which a quantity is changing with respect to its independent variable (time), is represented by its derivative. Thus, these phenomena are modeled by equations with differential variables that are called differential equations.

It is often impossible or cumbersome to find the exact solution of a differential equation since either there is no analytical solution for the model or the system is huge. Thus numerical methods are developed to approximate the solution of differential equations. Numerical methods made it possible for human beings to fulfill their dream to travel to other planets by computing the trajectory of spaceship with the help of computers. Indeed the development in numerical methods is parallel to the growth in computer technology. On one hand the accuracy of the numerical solution is of high importance and on the other hand how fast the solution is calculated.

To uniquely determine the solution of an ordinary differential equation, some outside condition is needed, typically an initial value or a boundary value. Some numerical methods, in particular *multistep methods*, demand several initial values to start the calculation of the solution. We suggest some techniques to provide adequate number of high accurate initial values with least effort.

Often the numerical methods calculate the solution of differential equations at discrete time points. If these time points are equally spaced we have a *fixed step-size* numerical solution and a *variable step-size* one otherwise. In differential equations, variable step-size methods also called *adaptive methods* are of significant importance. The location of the time points has to be selected such that an accurate numerical

solution is obtained while keeping the number of points small. Adaptive methods take smaller step-sizes when needed while they allow for larger step-sizes when the accuracy is not affected by it.

We present an adaptive form of two classes of multistep methods. The first class is called $\beta-$*blocked multistep methods*. These are used for the solution of systems that contain both differential and non-differential equations. The second class is called *strong stability preserving methods* and Those are applied to the solution of models such as those of sea-waves that experience crashes.

List of Papers

This thesis is based on the following papers, listed in the order of publication.

I. F. Mohammadi, C. Arévalo and C. Führer.
Restarting algorithms for simulation problems with discontinuities
Proceedings of the 10 International Modelica Conference, March 10-12, 2014, Lund, Sweden.

II. F. Mohammadi, C. Arévalo and C. Führer.
Runge-Kutta restarters for multistep methods in presence of frequent discontinuities.
J. Computational and Applied Mathematics,V. 316: P. 287-297, 2017.

III. F. Mohammadi, C. Arévalo and C. Führer.
Construction of adaptive strong stability preserving multistep methods.
J. SIAM . Numerical methods, Reviewed and re-submitted, June 2018.

IV. Chapter 4 covers some unpublished results on adaptive $\beta-$blocked methods.

Author's contribution

I hereby describe my contribution to each of these papers.

Paper I. I implemented the method, and performed the numerical experiments.

Paper II. I developed and analyzed the methods and performed the numerical experiments.

Paper III. I designed, implemented, and analyzed the formulations and carried out the numerical results.

Chapter 4. The ideas presented here are of my own design.

Contents

Chapter 1

Introduction

Physical phenomena are often modeled by differential equations, but finding their exact solution is not always possible or desirable, often because of their huge size or complexity. To overcome this challenge, the first numerical method was introduced by Euler 250 years ago. Well-designed numerical methods approximate the solution of differential equations efficiently and with sufficient accuracy. Nowadays, a vast variety of numerical methods are at hand, many geared towards a specific application or a particular model structure. In order to have an efficient integration, the particularities of the differential equation have to be considered when choosing an appropriate numerical method.

Numerical methods are classified according to the memory they need from step to step in the integration process. One-step methods use information of the solution at time t_n to approximate the solution at time t_{n+1}. Thus at each step only the value of previous solutions determines the next solution. Multistep methods are time-stepping methods that do use information from several previous steps to approximate the next solution. A method that uses k previously computed solutions to approximate at a new point is called a $k-$*step method*.

Two features of a numerical method are important when choosing a numerical method: order and stability. Using smaller time steps, that is, a finer discretization, results in more computational effort. The payoff should be a higher accuracy in the solution. This leads to the notion of *order*: higher order methods allow for larger step-sizes to produce a required accuracy. Furthermore, a numerical method is called *stable* if it damps out the small error and perturbations in previous steps as

it approximates the solution of differential equations along an interval.

In the numerical solution of the equations of a mathematical model, both quantity and quality aspects have to be considered. The quantity aspect is the order of the numerical method, because it affects its efficiency, and the quality aspect is the accuracy of the approximation and it is related to the choice of the method in connection to the structure of the problem.

The step-size, or difference between two time points, $h_n = t_{n+1} - t_n$, plays an important role in the behavior of a numerical method. In practice, it is highly desirable to solve a problem by taking step-sizes as large as possible while controlling the estimated error, in order to obtain a reasonable accuracy for the solution [51]. The variation of the step-sizes should be dependent on the dynamics of the model. This can be illustrated by imagining you are driving on a winding road. In order to reach your destination safely and on time you need to reduce the speed in the turns and increase it when the road is straight. Time-adaptive numerical methods behave this way; the step-size is reduced when the dynamics of the model changes rapidly and it is increased when the changes are slow.

The subject of this thesis is adaptive linear multistep methods. In particular, we look at three different aspects, namely, how to restart a method after a discontinuity, how to introduce adaptivity for methods that solve differential equations with constraints, and how to formulate adaptive methods for differential equations arising from problems subject to conservation laws.

Industrial models often contain discontinuities because of friction, changes of degrees of freedom, impacts, and other physical factors. As a simple model, consider throwing a ball against a wall, so that it bounces back after hitting the wall. We can model the trajectory of the ball by equations that express the ball's position and velocity with respect to time, together with the position and velocity at the instant when the ball leaves the thrower's hands. Because the velocity changes its sign instantaneously as the ball hits the wall, a discontinuity is introduced. When a discontinuity is detected during integration, the method must be restarted and $k-1$ approximate solution values must be generated in order to initialize the k-step method. The overall performance of the simulation of models with discontinuities depends strongly on the restarting method for the integration after a discontinuity has been detected.

Differential algebraic equations (DAEs) are systems of differential equations combined with algebraic constraints. These equations arise mainly from modeling elec-

trical circuits and mechanical systems. For instance, an electrical circuit consists of differential equations that represent the dynamic of capacitors, resistors and inductors, and nonlinear algebraic equations that impose Kirchhoff laws. Also, the simulation of mechanical systems is required in robotics, as well as in the design and simulation of vehicles, including cars and trains. Here force laws are modeled by differential equations and joints impose algebraic equations. The numerical treatment of DAEs has some challenges that require special consideration. In particular, numerical methods for these type of problems have stability issues that require modifications of the standard methods for differential equations.

Another interesting class of ordinary differential equations occur in the solution of hyperbolic partial differential equations. These models have some monotonicity properties that must be preserved during the numerical integration. Multistep methods may be used for this purpose at an advantage, but how to construct methods with varying step-sizes for these problems is still an open question.

This thesis is based on four seminal papers: Schwerin and Bock [82], Arévalo and Söderlind [13], Arévalo et al. [10] and Hadjimichael et al. [40].

In Chapter 3 we study some (re-)starting methods. Schwerin and Bock introduced a third order Runge–Kutta (RK) starter to initialize multistep methods in the presence of frequent discontinuities. However, their methodology did not allow for higher order RK starters. We present two families of higher order RK restarters based on [82].

The $\beta-$blocked multistep methods were developed to solve a particular type of DAE systems. However, these methods were defined only for fixed step-sizes, and there was no successful attempt to formulate adaptive $\beta-$blocked methods. In Chapter 4 we review these methods [10] and present a parametric formulation that make them adaptive.

The adaptive formulation for strong stability preserving (SSP) multistep methods was first introduced in [40], for methods up to order three. In Chapter 5 we utilized the parametric formulation of multistep methods in [13], that is adaptive by construction, to obtain time adaptivity for higher order methods.

1.1 Thesis outline

The structure of the thesis is as following:

In **Chapter 2**, multistep methods and basic ideas behind them are reviewed.
In **Chapter 3**, three starting techniques to initialize multistep methods are introduced.
In **Chapter 4**, the first adaptive $\beta-$blocked multistep methods are constructed by developing a parametric formulation for these methods.
In **Chapter 5**, we explain the derivation of an adaptive parametric method for strong stability preserving multistep methods.
In **Chapter 6**, conclusions and future aspects of this research are discussed.

Chapter 2

Multistep methods

Multistep methods are an important class of numerical methods for solving initial value problems. While one step methods require the initial value at the previous time step to compute the value at the next, a $k-$step method utilizes the k previous approximated solution values to compute the next value.

Consider the initial value problem

$$\dot{y} = f(t, y), \quad y(t_0) = y_0, \quad t \in [t_0, t_f]. \tag{2.1}$$

A numerical method applied to Equation (2.1) approximates the solution $y(t)$ at discrete time points, t_i,

$$y_i \approx y(t_i), \quad t_i = t_0 + ih, \tag{2.2}$$

where h is the time step. A linear multistep method with constant step-size is defined by

$$\sum_{i=0}^{k} \alpha_{k-i} x_{n-i} = h \sum_{i=0}^{k} \beta_{k-i} f(t_{n-i}, y_{n-i}) \tag{2.3}$$

where α_j and β_j are method coefficients. For a $k-$step method, we need $\alpha_k \neq 0$ and $\alpha_0 \neq 0$ or $\beta_0 \neq 0$. The multistep method (2.3) is called implicit if $\beta_k \neq 0$ and explicit otherwise.

Given the multistep method (2.3), its generating polynomials are defined as

$$\rho(\zeta) := \sum_{i=0}^{k} \alpha_i \zeta^i \tag{2.4}$$

$$\sigma(\zeta) := \sum_{i=0}^{k} \beta_i \zeta^i \tag{2.5}$$

Using difference operator notation, we can define corresponding operators

$$\rho = E^{-k}\rho(E), \qquad \sigma = E^{-k}\sigma(E), \tag{2.6}$$

where E is the forward shift operator. Thus the linear multistep method (2.3) can be represented by a pair of its generating polynomials (ρ, σ) and

$$\rho x_n = h\sigma f(t_n, x_n). \tag{2.7}$$

2.1 Order and stability

Once the local error of a multistep method is defined, we can introduce the concept of order of consistency for multistep methods. The local residual of a linear multistep method is obtained by substituting the exact solution $y(t)$ evaluated at discrete times t_{n-i} for $i = 0, \ldots, k$ into (2.3),

$$l_n := \sum_{i=0}^{k} \alpha_{k-i} y(t_{n-i}) - h \sum_{i=0}^{k} \beta_{k-i} f(t_{n-i}, y(t_{n-i})). \tag{2.8}$$

A multistep method is said to be of order p, if for sufficiently smooth $y(t)$ we have $l_n = \mathcal{O}(h^{p+1})$. The equivalent conditions for a $k-$step method to have order p are

$$\begin{aligned} \sum_{i=0}^{k} \alpha_i &= 0 \\ \sum_{i=0}^{k} \alpha_i i^q &= q \sum_{i=0}^{k} \beta_i i^{q-1}, \quad q = 1, \ldots, p. \end{aligned} \tag{2.9}$$

Conditions (2.9) are equivalent to requiring that the numerical method is exact for polynomials of degree p. Furthermore, if the method (ρ, σ) has order p then its

variable step-size counterpart is consistent of the same order when $l_n = 0$ in (2.8) whenever the solution is a polynomial of degree p, that is,

$$\sum_{i=0}^{k} \alpha_{k-i} P(t_{n-i}) - h \sum_{i=0}^{k} \beta_{k-i} \dot{P}(t_{n-i}) = 0, \quad \forall P \in \Pi_p.$$

In particular, a method is consistent if p is greater or equal to 1. However, a consistent multistep method is not necessarily convergent. We also need to know that small changes in the initial values produce bounded changes in the numerical solution. This concept is called stability. The multistep method is called zero-stable if the roots of $\rho(\zeta)$ lie inside the unit circle and the roots on the unit circle are simple [42].

To have a convergent multistep method, the quantity of interest is the global error of the solution, defined as

$$e_n := y(t_n) - y_n, \tag{2.10}$$

where $y(t_n)$ and y_n are the exact and the numerical solution at a given time point t_n with $h = t_n/n$. We say a multistep method is convergent if, for exact initial values,

$$e_n \to 0 \quad \text{for} \quad h \to 0. \tag{2.11}$$

The local error of a consistent method contributes to its global error and zero-stable methods assure that by reducing the step-size, the global error decreases also. The Dahlquist equivalence theorem [52] guarantees that a consistent and zero-stable multistep method is convergent.

For a specific class of differential equations that consists of a set of ordinary differential equations and algebraic conditions, zero-stability is not enough to ensure a stable solution. Actually, the presence of algebraic equations impose a condition on the σ polynomial, namely, σ has to have all its roots inside the unit circle. This condition on σ is called *stability at infinity*.

2.2 Adaptive multistep methods

When solving particular systems of differential equations with multistep methods, varying the step-size will allow for a required precision of approximated values while avoiding unnecessary computational work.

There are two main approaches to variable step-size multistep methods [42]. The first one is based on using equally spaced points with a fixed time step, h, and then approximating a new solution by polynomial interpolation on a non-uniform grid. In the second approach methods are adjusted to variable step-sizes by altering the method's coefficients at every step.

An adaptive linear multistep method is defined by

$$\sum_{i=0}^{k} \alpha_{n,k-i} y_{n-i} = h_n \sum_{i=0}^{k} \beta_{n,k-i} f(t_{n-i}, y_{n-i}) \tag{2.12}$$

where the coefficients $\alpha_{n,k-i}$ and $\beta_{n,k-i}$ actually depend on the ratios $w_i = \frac{h_i}{h_{i-1}}$, $i = n-k+1, \ldots, n-1$.

A collocation formulation to construct variable step-size multistep methods was initially introduced in [7]. Arévalo and Söderlind [13] refined this approach to obtain a formulation where specific methods are defined by polynomials and characterized by a set of fixed parameters. In this formulation each k-step method of maximal order, i.e, $p \geq k$, is represented by a fixed set of parameters. The method uses these parameters at each step to construct the polynomial that will advance the solution. In Paper III we proposed an extension of this formulation for methods with $p < k$, which include strong stability preserving methods. In Chapter 4 we present a way to construct adaptive methods for index-2 DAEs with $\beta-$block stabilization [9], inspired by this formulation of multistep methods.

2.2.1 Parametric formulation of multistep methods

The parametric formulation of a $k-$step method is defined by a polynomial that interpolates the solution values y_{n-i} and their corresponding vector field values y'_{n-i} approximated at the time samples $t_{n-k}, \ldots, t_{n-1}$ where $y'_{n-i} = f(t_{n-i}, y_{n-i})$ and $h_{n-i} = t_{n+1-i} - t_{n-i}$. Let Π_p denote the space of polynomials of degree p. The method polynomial $P_n \in \Pi_p$ that approximates the solution $y(t)$ for $t > t_{n-1}$ gives

$$y_n := P_n(t_n). \tag{2.13}$$

The formulation of parametric multistep methods makes use of the state and derivative *slacks*, defined as follows.

Definition 1 *[13] Let the sequences* $\{y_{n-i}\}_{i=0}^{k}$ *and* $\{y'_{n-i}\}_{i=0}^{k}$ *be given for a fixed* n*. Further, let* $P_n \in \Pi_p$ *with* $p \leq k+1$*. The state slack* s_{n-i} *and the derivative slack* s'_{n-i} *at* t_{n-i} *are defined as*

$$s_{n-i} = P_n(t_{n-i}) - y_{n-i}, \qquad s'_{n-i} = \dot{P}_n(t_{n-i}) - y'_{n-i}, \qquad i = 0, \ldots, k. \tag{2.14}$$

According to the first Dahlquist barrier [52], it is not possible to interpolate all the state and vector field values so we leave slack on some of them.

Three types of multistep methods are discussed in [13], explicit and implicit $k-$step of order k and implicit $k-$step of order $k+1$. Each one is defined by a particular parametrization. These formulations are used as a basis for the parametrization of lower order explicit methods and $\beta-$blocked methods for DAEs. A parametrization is introduced for each of them. We only make use of explicit $k-$step of order k and implicit $k-$step of order $k+1$ methods.

It was demonstrated in [13] that every explicit k-step method of order p can be defined by $y_n = P_n(t_n)$, with the polynomial $P_n \in \Pi_k$ satisfying the conditions

$$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-i}\cos\theta_{i-1} + h_{n-i}s'_{n-i}\sin\theta_{i-1} = 0; \;\; i = 2, \ldots, k, \end{cases} \tag{2.15}$$

where $\theta_i \in (-\frac{\pi}{2}, \frac{\pi}{2}]$ are the *method parameters*. The first two conditions are called *structural conditions* and make the method explicit. The additional linear combinations of state and derivative slacks are called *slack balance conditions*, and specify the particular method. Arévalo et al. [14] showed that the following parametric equivalence holds between the coefficients of a classical, constant step-size, multistep formula of maximal order and the method parameters:

$$\tan\theta_{i-1} = \frac{\beta_{k-i}}{\alpha_{k-i}} \;\text{ for } i = 2, \ldots, k. \tag{2.16}$$

Further, every implicit $k-$step method of order $p = k+1$ can be defined by $P_n \in \Pi_{k+1}$, satisfying the conditions

$$\begin{cases} s'_n = 0 \\ s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-i}\cos\theta_{i-1} + h_{n-i}s'_{n-i}\sin\theta_{i-1} = 0; \;\; i = 2, \ldots, k, \end{cases} \tag{2.17}$$

with $y_n := P_n(t_n)$, where the first condition causes the implicitness of the method.

Note that for a variable step-size method the parameters θ_{i-1} are constants, even though the coefficients α and β vary from step to step. Thus, with conditions (2.17) a variable step-size k-step method is defined in terms of constants $\theta_1, \ldots, \theta_{k-1}$. The rest of this chapter is devoted to the extension of this formulation to strong stability preserving and $\beta-$blocked multistep methods. We refer to Paper III and Chapter 4 for more details.

It is natural to apply $k-$step methods with the highest possible consistency order. The first Dahlquist barrier implies that the maximal convergent order of a $k-$step method is at most $k+2$ for k even , and $k+1$ for k odd. If the method is also explicit, it cannot attain order greater than k. In this section we introduce two class of multistep methods that are applied in this thesis.

2.2.2 Multistep methods of maximal orders

There are three families of multistep methods that are commonly used, Adams-Moulton methods, Adams-Bashforth methods and BDF formuals. Further, $k-$step Adams-Moulton methods and BDFs are implicit methods with order of consistency $k+1$ and k respectively, while $k-$step Adams-Bashforth methods are explicit of order k. For stiff models, implicit multistep methods, especially BDF methods are suggested [44]. This is due to the severe restriction on the step-sizes to fulfill the numerical stability of the explicit methods. In addition, Adams-Moulton methods have bounded stability regions, hence they are intended for non-stiff integration.

A particularly interesting family of implicit $k-$step methods of order $k+1$ was introduced by Söderlind [78]. These methods, called difference-corrected BDF (*dcBDF*) methods, have $\rho = \rho^{\mathrm{BDF}}$ and their σ polynomial recovers the first term in the local truncation error of the BDF methods. The $k-$step dcBDF discretization of (2.1) can be written as

$$\rho^{\mathrm{BDF}} y_n = h(1 - \frac{\nabla^k}{k+1}) f(y_n) \tag{2.18}$$

where ρ^{BDF} is the generating polynomial of the corresponding BDF. Thus it only differs from BDF method, generating polynomilas $(\rho, 1)$, by the difference correction term $\frac{\nabla^k}{k+1}$.

ρ_i	σ_{ij} $\quad i=1,\ldots,6 \quad j=0,\ldots,6$						
∇	1	$-\frac{1}{2}\nabla$	$-\frac{1}{12}\nabla^2$	$-\frac{1}{24}\nabla^3$	$-\frac{19}{720}\nabla^4$	$-\frac{3}{160}\nabla^5$	$-\frac{863}{60480}\nabla^6$
$\frac{1}{2}\nabla^2$	1		$-\frac{1}{3}\nabla^2$	$-\frac{1}{12}\nabla^3$	$-\frac{17}{360}\nabla^4$	$-\frac{23}{720}\nabla^5$	$-\frac{143}{6048}\nabla^6$
$\frac{1}{3}\nabla^3$	1			$-\frac{1}{4}\nabla^3$	$-\frac{3}{40}\nabla^4$	$-\frac{11}{240}\nabla^5$	$-\frac{109}{3360}\nabla^6$
$\frac{1}{4}\nabla^4$	1				$-\frac{1}{5}\nabla^4$	$-\frac{1}{15}\nabla^5$	$-\frac{3}{70}\nabla^6$
$\frac{1}{5}\nabla^5$	1					$-\frac{1}{6}\nabla^5$	$-\frac{5}{84}\nabla^6$
$\frac{1}{6}\nabla^6$	1						$-\frac{1}{7}\nabla^6$

Table 2.1: An implicit difference correction method, IDCij is obtained by adding the ρ_i terms columnwise down to row i and sum terms up in row i up to column j. The $k-$step Adams-Moultom methods correspond to IDC$1k$, The $k-$step BDF is IDC$k0$ and IDCkk are $k-$step dcBDF methods.

All the mentioned methods in this section belong to the main class of multistep methods that are known as *Implicit Difference Correction* (IDC) methods. Table 2.1 [10] shows the IDC methods up to order 6. The IDCij methods are defined by (ρ_i, σ_{ij}) where ρ_i is the generating polynomial of the $i-$step BDF, and σ_{ij} is obtained by summing terms up in row i up to column j. The following example shows how to use Table 2.1 to find the formulation of a particular IDCij method.

Example 1 *The IDC34 method can be written by summing up ρ_i terms for $i = 1,\ldots,3$,*

$$\rho_3 = \nabla + \frac{1}{2}\nabla^2 + \frac{1}{3}\nabla^3$$

and summing up σ_{3j} for $i = 3$ and $j = 0,\ldots,4$,

$$\sigma_{34} = 1 - \frac{1}{4}\nabla^3 - \frac{3}{40}\nabla^4$$

Thus the IDC34 method is

$$\rho_3 y_n = h\sigma_{34} f(y_n).$$

The IDCij methods with $j \geq 1$ are implicit methods that are not suitable for stiff problems due to their restricted stability regions but they are useful in the context of $\beta-$blocked method for DAEs. Further, the $\beta-$blocked IDC methods [10] are marked by blue.

2.2.3 Multistep methods of lower orders

Although $k-$step methods of maximal order are the most popular multistep methods, often, in real-world applications, we favor a lower-order $k-$step method over a higher-order one if certain stability properties can be guaranteed.

In Section 5.1 we first introduce the application of a particular class of explicit multistep methods called *strong stability preserving* methods. It is well-known that these multistep methods with consistency order $p \geq 2$ have $p < k$ [58]. Thus these methods are a class of multistep methods of lower order. In Section 5.2 we first look for a polynomial formulation of multistep methods of lower order and then we show that SSP multistep methods can be described with a similar parametric formulation.

Chapter 3

Initialization of multistep methods

A numerical method for solving ordinary differential equations may be classified according to its memory requirement at each step in time. A Runge–Kutta method needs a single initial value to generate the next approximated solution, but the initialization of a k-step method requires k starting values. In the present chapter we are looking for a multistep starter which provides high order starting values with a minimal number of function evaluations.

We aim to introduce starters that can be used to start and especially to restart a multistep method after an interruption in the integration process. This interruption occurs because of the wrong error estimation of discontinuous systems where their right hand-side is not smooth enough [29, 26]. There are two main strategies to solve discontinuous ODEs. The first one is to ignore the discontinuity and apply an standard ODEs integration method. If a discontinuity is present in an integration step, the error estimation that forms the basis of step-size control techniques, becomes large and the step is rejected. After the step-size rejection, a smaller step-size is taken by the controller. This requires high computational effort since the step-size rejection often arise repeatedly until the discontinuity is passed. Further, the numerical solution efficiency and error estimation are based on the assumption that the solution and its derivatives are sufficiently differentiable. Thus if the assumption is not fulfilled the error estimation is not valid. The second approach is to stop the integration and after localizing the discontinuity, restart it. If the

model experiences frequent discontinuities, minimizing the computational effort of restarting a multistep method while obtaining efficient initial values becomes of a significant interest.

3.1 Background

Consider an initial value problem of the form

$$\dot{y} = f(t, y), \quad y_0 = y(t_0), \quad t \in [t_0, t_f], \tag{3.1}$$

with a sufficiently differentiable right-hand side function f. An s-stage explicit Runge–Kutta (RK) method with nodes $\{c_i\}_{i=1}^s$, weights $\{b_i\}_{i=1}^s$ and coefficients $\{a_{ij}\}$ with $i = 1, \ldots, s$, $j = 1, \ldots, i-1$ applied to problem (6.25) is defined by

$$\begin{aligned} Y_i &= y_0 + H \sum_{j=1}^{i-1} a_{ij} K_j, \\ K_i &= f(t_0 + c_i H, Y_i), \quad i = 1, \ldots, s, \end{aligned} \tag{3.2}$$

where H, Y_i and K_i are the RK step-size, stage values and the stage derivatives respectively, and the numerical solution at $t_0 + H$ is given by

$$y_1 = y_0 + H \sum_{j=1}^{s} b_j K_j. \tag{3.3}$$

RK methods are often represented by a *Butcher tableau*.

c	A
	b^{T}

Table 3.1: Butcher tableau for (6.26) with node vector c, weight vector b and coefficient matrix A.

The general form of a linear multistep method [29] applied to the differential equation in (6.25) is

$$\sum_{i=0}^{k} \alpha_{k-i,n} y_{n+1-i} = h_n \sum_{i=0}^{k} \beta_{k-i,n} f(t_{n+1-i}, y_{n+1-i}), \tag{3.4}$$

where coefficients α_{k-i} and β_{k-i} determine the method. The multistep method (3.4) applies a linear combination of past values y_{n+1-i} for $i = 1, \ldots, k$ to approximate the solution at y_{n+1}.

3.2 The Nordsieck vector

Instead of storing several previous solutions of the differential equation (6.25) with their state values y or its derivatives $\dot{y}$, Nordsieck [65] proposed to save them using the Nordsieck vector with higher degree scaled derivatives,

$$\frac{h^j}{j!} y^{(j)}(t_n),$$

for $j = 0, \ldots, p$, where p is the order of the integrator.

If we have numerical solution values

$$(t_n, y_n), (t_{n-1}, y_{n-1}), \ldots, (t_{n-p}, y_{n-p}),$$

and define the step-sizes $h_i = t_i - t_{i-1}$, we can compute the Nordsieck solution vector by solving the linear system [75],

$$\begin{pmatrix} y_n \\ y_{n-1} \\ y_{n-2} \\ \vdots \\ y_{n-p} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & -1 & 1 & \cdots & (-1)^p \\ 1 & -\xi_{2,n} & (-\xi_{2,n})^2 & \cdots & (-\xi_{2,n})^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & -\xi_{p,n} & (-\xi_{p,n})^2 & \cdots & (-\xi_{p,n})^p \end{pmatrix} \begin{pmatrix} y_n \\ h_n \dot{y}_n \\ \frac{h_n^2}{2!} \ddot{y}_n \\ \vdots \\ \frac{h_n^p}{p!} y_n^{(p)} \end{pmatrix},$$

where

$$\xi_{j,n} = \frac{t_n - t_{n-j}}{h_n} = \frac{1}{h_n} \sum_{i=0}^{j-1} h_{n-i}.$$

If the step-sizes are equidistant, then we have $\xi_{j,n} = \xi_n = j$ and the transformation matrix is

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & -1 & 1 & \cdots & (-1)^p \\ 1 & -2 & 4 & \cdots & (-2)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & -p & (-p)^2 & \cdots & (-p)^p \end{pmatrix} \tag{3.5}$$

Thus, it is possible to convert a vector of state values at consecutive grid points into a Nordsieck array and vice versa without loss of accuracy.

In this thesis we are especially interested in the solution of ordinary differential equations that contain discontinuities in the right-hand side function. It is crucial to be able to get all the necessary information in order to handle these discontinuities.

3.3 Event handling

Many problems in simulation and control are described by systems of ordinary differential equations (ODEs) of the form (6.25) having a right-hand side function that contains discontinuities in some of its components or higher derivatives.

Ignoring a discontinuity in the right-hand side function or its higher derivatives may cause a wrong error estimation or a drastic reduction of the step-size and possibly an order reduction of the method. There are several ways to handle the discontinuities while having an efficient integration over them. Our approach is based on four steps:

- Detecting the discontinuity
- Localizing the discontinuity
- Passing the discontinuity
- Restarting the integration

3.3.1 Detecting and localizing the discontinuity

The conditions under which a discontinuity occurs are governed by a set of algebraic equations known as the *switching function* [29],

$$g(t, y(t)) = (g_1(t, y(t)), \ldots, g_\nu(t, y(t)))^T.$$

The differential equation in (6.25) depends on the sign changes of the switching function and can be rewritten as

$$\dot{y} = f(t, y, \operatorname{sign} g).$$

The zeros of the switching function define discontinuities referred to as *events* in the simulation literature.

Example 2 *[29] The model*

$$\dot{x} = |x|$$

can be formulated by switching functions $q(t, x) = x$*, as*

$$\dot{x} = \begin{cases} x & \textit{for sign } q = 1 \quad (3.6) \\ -x & \textit{for sign } q = -1 \quad (3.7) \end{cases}$$

It is necessary to rewrite the ODEs with discontinuities as

$$\dot{x} = f(t, x, s) \text{ with } s = \text{sign } q$$

and then solve them numerically.

The detection of the possible presence of a discontinuity is done by checking the sign of the switching function. If it changes sign then the event needs to be localized by finding the root of the switching function, i.e. finding the $t^* \in [t_n, t_{n+1}]$ such that

$$g(t^*, y(t^*)) = 0.$$

Thus we need a continuous representation of the state values to determine the position of the event, not just in the discretization points, but also between them and with high accuracy. This continuous representation is available for Adams and BDF methods since they are based on polynomial interpolation. There are several algorithms for localizing the discontinuity, and here we take a look at the *Illinois algorithm* [27], a variant of the secant method that has been used in several ODE solvers to find the root of the switching function. The Illinois algorithm is an iterative procedure in the interval $[t_n, t_{n+1}]$ where $g(t_n) \cdot g(t_{n+1}) < 0$, that applies the secant method to modify one of the end points of the interval. The algorithm can be described as follows:

Let $x_0 = t_n$ and $x_1 = t_{n+1}$ where $g(x_0) \cdot g(x_1) < 0$. A new value x_{i+1} for $i = 1, 2, 3 \ldots$ is computed by

$$x_{i+1} = x_i - \frac{g(x_i)(x_i - x_{i-1})}{g(x_i) - g(x_{i-1})}.$$

Then $g(x_{i+1})$ is evaluated and

- if $g(x_i) \cdot g(x_{i+1}) < 0$, then $(x_{i-1}, g(x_{i-1}))$ is replaced by $(x_{i+1}, g(x_{i+1}))$.

- if $g(x_i) \cdot g(x_{i+1}) > 0$, then $(x_{i-1}, g(x_{i-1}))$ is replaced by $(x_{i-1}, \frac{g(x_{i-1})}{2})$.

The recursion is stopped when a suitable criterion is satisfied, for example, when $|x_{i+1} - x_i|$ is less than a prescribed tolerance. Figure 3.1 illustrates this algorithm.

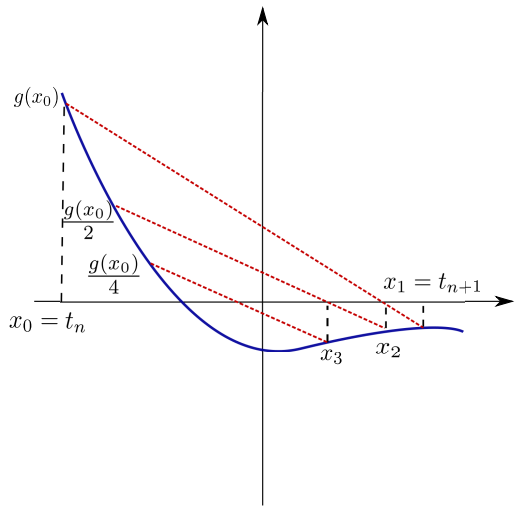

Figure 3.1: The *Illinois algorithm* in the interval $[t_n, t_{n+1}]$

After localizing the discontinuity the integrator must be restarted. A one-step method can be restarted without any difficulty as no past values are needed. For multistep methods, that depend on several previous values to compute the solution at the current time step, a restarting scheme must be specified.

3.4 (Re-)Starting a multistep method

There are three known approaches for initializing multistep methods [64]. The first approach makes repeated use of an RK method of the required order to generate the necessary k past values for the k-step method. The second one uses several multistep methods of increasing orders. The third one uses a single RK method of the desired order.

3.4.1 Several-step single-stage starter

In the 60's a common technique to generate the starting values for multistep methods was to apply an RK step repeatedly. (3.2) illustrates the idea of applying $k-1$ RK steps to start a k-step method.

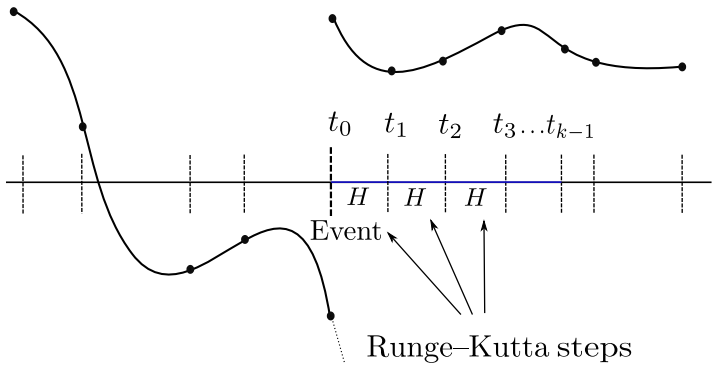

Figure 3.2: After detecting a discontinuity at t_0 the simulation is restarted and $k-1$ RK steps of step-size H are taken.

This starter is chosen in such a way that it generates initial values of the same order as the multistep method used thereafter. However, it needs at least $k(k-1)$ function evaluations.

3.4.2 Winding up states

Gear [35] developed a self-starting scheme that initially uses a one-step method, and then uses multistep methods of increasing order until the working order is achieved, as shown in (3.3). For example, for a three-step method we need to compute two points in addition to the initial value before entering the main time-stepping loop. This scheme starts with low order methods and very small step-sizes in order to gain accuracy and gradually increases the order and the step-size.

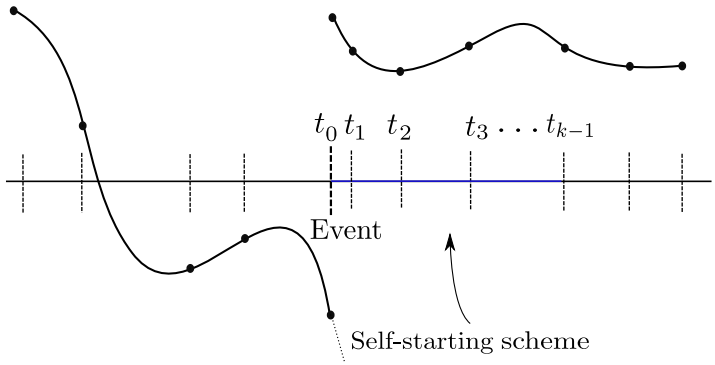

Figure 3.3: Self-starting scheme starting with a one-step method and a small step-size and successively increasing the step-size and the order of the method.

This self-starting scheme has been implemented in most current multistep ODE solvers.

3.4.3 Single-step several-stage starters

The idea of (re-)starting multistep methods with high order initial values on the one hand and a minimal number of function evaluations on the other, motivates the search for a family of explicit RK methods. Starters of different orders can be constructed, and all apply a single step of an RK method to generate an adequate number of starting values from the stage values of the method.

We introduced three families of single-step RK starters. The first family, denoted by $\mathscr{F}_1$, was developed by Gear [33] and consists of RK starters with an extrapolation technique that generates Nordsieck solution vectors. The other two families were developed in [64]. The second family, denoted by $\mathscr{F}_2$, uses the internal stages of an RK method to approximate the required starting values for the multistep methods. The last one, denoted by $\mathscr{F}_3$, uses the method's weight vectors to approximate the solution at distinct fractions of the RK step-size. Both families provide error estimation for the RK step.

A 5th order Runge–Kutta starter with high order internal values is presented here. This starter is not included in Paper II.

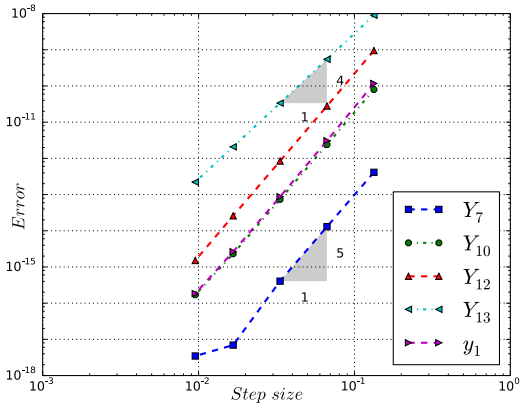

Figure 3.4: The accuracy plot of the solution of the linear test equation by 5th order RK starter (3.8) shows order 5 for internal stages Y_7, Y_{10}, Y_{12} and the approximated solution y_1. Y_{13} has order 4 and is used for error estimation.

3.4.4 A fifth order Runge–Kutta starter

We construct a 5th order RK starter with the same strategy applied to the 4th order RK starter of this family. It is known from [42] that we need at least six stages to obtain a 5th order RK method. Our starter would have thirteen stages in total to generate five state values of order 5 and one more of order 4 for error estimation. The coefficients of th Butcher tableau are the solution of the nonlinear system of all Runge–Kutta order equations. The elements of A and b were calculated as follows

$$
\begin{aligned}
a_{21} &= \tfrac{1}{20} & a_{93} &= \tfrac{94696591}{11720647} & a_{12,7} &= -\tfrac{87}{2} \\
a_{31} &= \tfrac{3}{160} & a_{94} &= -\tfrac{-352621205}{15003372} & a_{12,8} &= 0 \\
a_{32} &= \tfrac{9}{160} & a_{95} &= -\tfrac{316213811}{4964063} & a_{12,9} &= -\tfrac{6714159}{175827524} \\
a_{41} &= \tfrac{3}{40} & a_{96} &= -\tfrac{335891977}{6014773} & a_{12,10} &= \tfrac{5}{4} \\
a_{42} &= -\tfrac{9}{40} & a_{97} &= \tfrac{1531407208}{12995199} & a_{12,11} &= 0 \\
a_{43} &= \tfrac{6}{20} & a_{98} &= 0 & a_{13,1} &= -\tfrac{21793700}{146110639} \\
a_{51} &= -\tfrac{11}{216} & a_{10,1} &= -\tfrac{61233167}{119633322} & a_{13,2} &= 0 \\
a_{52} &= \tfrac{5}{8} & a_{10,2} &= 0 & a_{13,3} &= \tfrac{19145766}{113939551} \\
a_{53} &= -\tfrac{70}{108} & a_{10,3} &= \tfrac{12880742}{6250467} & a_{13,4} &= \tfrac{8434687}{36458574} \\
a_{54} &= \tfrac{35}{108} & a_{10,4} &= -\tfrac{13727525}{6430593} & a_{13,5} &= \tfrac{2012005}{39716421} \\
a_{61} &= \tfrac{1631}{221184} & a_{10,5} &= -\tfrac{7962931}{7810968} & a_{13,6} &= \tfrac{8409989}{57254530} \\
a_{62} &= \tfrac{175}{2048} & a_{10,6} &= -\tfrac{18723895}{33225736} & a_{13,7} &= \tfrac{10739409}{94504714} \\
a_{63} &= \tfrac{575}{55296} & a_{10,7} &= 0 & a_{13,8} &= 0 \\
a_{64} &= \tfrac{44275}{442368} & a_{10,8} &= \tfrac{303385}{143008499} & a_{13,9} &= -\tfrac{3321}{86525909} \\
a_{65} &= \tfrac{253}{16384} & a_{10,9} &= 0 & a_{13,10} &= -\tfrac{30352753}{150092385} \\
a_{71} &= \tfrac{37}{1512} & a_{11,1} &= -\tfrac{899776084}{7592411} & a_{13,11} &= 0 \\
a_{72} &= 0 & a_{11,2} &= \tfrac{104616515}{25909789} & a_{13,12} &= \tfrac{70257074}{109630355} \\
a_{73} &= \tfrac{250}{2484} & a_{11,3} &= \tfrac{151902626}{20177663} & b_1 &= \tfrac{33172492}{855554089} \\
a_{74} &= \tfrac{125}{2376} & a_{11,4} &= \tfrac{224901405}{19473086} & b_2 &= 0 \\
a_{75} &= 0 & a_{11,5} &= \tfrac{1053233659}{62474194} & b_3 &= \tfrac{3021245}{89251943} \\
a_{76} &= \tfrac{512}{7084} & a_{11,6} &= \tfrac{3149087129}{204549519} & b_4 &= \tfrac{5956469}{58978530} \\
a_{81} &= \tfrac{161343841}{5000000} & a_{11,7} &= \tfrac{275362480}{17214843} & b_5 &= \tfrac{851373}{32201684} \\
a_{82} &= -\tfrac{197478323}{5000000} & a_{11,8} &= \tfrac{74188771}{3079109} & b_6 &= \tfrac{11559106}{149527791} \\
a_{83} &= -\tfrac{207849921}{10000000} & a_{11,9} &= \tfrac{58704791}{27211802} & b_7 &= \tfrac{11325471}{112382620} \\
a_{84} &= \tfrac{55574077}{2500000} & a_{11,10} &= \tfrac{205250753}{9483265} & b_8 &= 0 \\
a_{85} &= \tfrac{454543383}{10000000} & a_{12,1} &= \tfrac{584323646}{57380211} & b_9 &= -\tfrac{12983}{235976962} \\
a_{86} &= \tfrac{103992363}{2000000} & a_{12,3} &= -\tfrac{121871270}{3367729} & b_{10} &= \tfrac{17692261}{82454251} \\
a_{87} &= -\tfrac{381650311}{4186219} & a_{12,4} &= \tfrac{529441805}{13610864} & b_{11} &= 0 \\
a_{91} &= -\tfrac{69897325}{2928599} & a_{12,5} &= \tfrac{252783117}{13775503} & b_{12} &= \tfrac{38892959}{120069679} \\
a_{92} &= \tfrac{281012770}{6692133} & a_{12,6} &= \tfrac{208565843}{19322161} & b_{13} &= \tfrac{11804845}{141497517}
\end{aligned}
\tag{3.8}
$$

and the error coefficients are

$$\begin{aligned}
e_1 &= \tfrac{846370458425881}{4503599627370496} & e_8 &= 0\\
e_2 &= 0 & e_9 &= -\tfrac{306889540921203}{18446744073709551616}\\
e_3 &= -\tfrac{4834476178104325}{36028797018963968} & e_{10} &= \tfrac{3754180563579103}{9007199254740992}\\
e_4 &= -\tfrac{4696571107073997}{36028797018963968} & e_{11} &= 0\\
e_5 &= -\tfrac{6981083844601939}{288230376151711744} & e_{12} &= -\tfrac{5709380784211703}{18014398509481984}\\
e_6 &= -\tfrac{5014032849404753}{72057594037927936} & e_{13} &= \tfrac{601161594016284 7}{72057594037927936}\\
e_7 &= -\tfrac{463433739290933}{36028797018963968}
\end{aligned} \tag{3.9}$$

The main objective in Paper I is to implement the multistep starters introduced by Gear in Hindmarsh's code LSODAR. We tasted their performance when used to restart ODE solvers for problems with discontinuities. In Paper II we constructed other families of (re-)starters and compared their efficiency and performance.

3.5 Initial step-size

The integrator needs an initial step-size to restart after each discontinuity. The choice of the initial step-size has particular importance when a high order integrator is applied to the model. Although a bad starting choice for H would be repaired by the step-size control often, it can generate large error and as a consequence lost of accuracy and computational effort that cannot be repaired. An algorithm for computing the initial step-size is introduced in [13] and the algorithm of this starter is represented in Algorithm 1.

Algorithm 1: Autostart algorithm used to calculate the initial step-size

Data: f, t_0, y_0, t_f
Result: Initial step-size

- Approximate the Lipschitz constant
  $L_0 = ||f(t_0, y_0 + \Delta y) - f(t_0, y_0)||/\Delta y$,
  where Δy is a small perturbation of y_0
- Set $H_0 = 0.1/L_0$
- Take a single Euler step forward $y_1 = y_0 + H_0 f(t_0, y_0)$
- Take a single Euler step backward in time from y_1,
  $\tilde{y}_0 = y_1 - H_0 f(t_0 + h_0, y_1)$
- Compute a better approximation of the Lipschitz constant
  $L = ||f(t_0, \tilde{y}_0) - f(t_0, y_0)||/||\tilde{y}_0 - y_0||$
- Calculate logarithmic norm
  $M = (\tilde{y}_0 - y_0)^T (f(t_0, \tilde{y}_0) - f(t_0, y_0))/||\tilde{y}_0 - y_0||^2$
- Compute the scaling factor
  $\kappa = \frac{1}{(\tilde{y}_0 - y_0)} + \frac{1}{H_0(L + M/2)}$
- Set the initial step-size as
  $H = \frac{H_0 \kappa (\text{Tol})^{\frac{1}{p+1}}}{2}$,
  where p is the order of the method
- Define $H = \min(H, 10^{-3}(t_f, t_0))$,
  where t_f is the final time of the simulation

3.6 Implementation

In this section we discuss implementation issue of the RK starters. Once a starter is implemented, it is added to an ODE solver which handles events. There are several ODE solver packages with event handling such as the SUNDIALS codes CVODE and IDA, [49] and the FORTRAN program LSODAR, [48]. These codes are available in Assimulo, a Python wrapper for ODE software, which permits a

description of the model together with the discontinuity handling in Python and allows access to a large variety of professional and experimental solvers [5]. We have chosen to implement our algorithms in Python, and to incorporate them in LSODAR.

3.6.1 LSODAR features

The choice of LSODAR as a solver for implementing all our RK starters, is based on the fact that it uses the Nordsieck vector of the solution at each step. This vector is provided by the $\mathscr{F}_1$ family of RK starters.

LSODAR is written in FORTRAN. Thus it is possible to use F2PY, a tool that makes it possible to call Fortran subroutines and to access Fortran COMMON blocks from Python.

LSODAR [68] is a code for solving explicit ordinary differential equations with the following features:

- It chooses either an Adams-Moulton method together with fixed-point iteration or a BDF method together with Newton iteration depending on a heuristic stiffness test made at every integration step.
- It is a variable step-size solver, that selects the step-size depending on a local error estimate and a given error tolerance.
- It is a variable order method, that selects the order by efficiency considerations.
- Given a user defined switching function, it returns the control to the user once a discontinuity is detected by finding the root of the switching function within the current time interval (see (3.3.1)).

Our goal is to add the RK starters $\mathscr{F}_1$, $\mathscr{F}_2$, and $\mathscr{F}_3$ (see Paper I and Paper II) without modifying the original code. These starters are implemented and tested for the restarting phases after discontinuities. The code for the RK starters is prototyped in Assimulo [5].

3.6.2 Code organization

The work flow of LSODAR within Assimulo is depicted in (3.5).

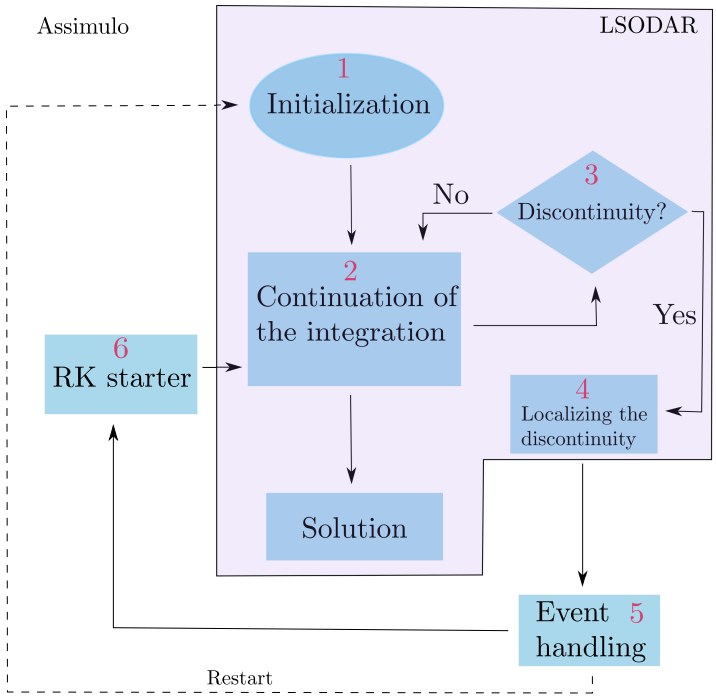

Figure 3.5: Code organization of LSODAR.

The LSODAR code structure has the following parts:

1. The initialization of the integration is done by calculating the initial step-size and loading the initial value of the problem in the Nordsieck solution vector.
2. The solutions are approximated by one of the numerical methods implemented in LSODAR.
3. After completion of each step, the switching functions are checked for the possible presence of a discontinuity.
4. A discontinuity is localized by applying the Illinois algorithm.
5. After localization of the discontinuity, the control of the simulation is given to a user-specified method to handle the event in Assimulo. Finally, the

integration is reinitialized either by using a classical starter or one of the RK starters. Classical reinitialization is preformed by restarting from step 1.

6. Alternatively, restart by applying an RK starter. This is done by modifying and updating the Nordsieck solution vector in internal arrays while the other saved data is kept unchanged.

Our contribution in the code structure is in the reinitialization of the integration with one of the RK starters at step 6, all other steps are left unaltered.

The data communication between different calls to LSODAR is done by passing data in two work arrays, RWORK and IWORK, and through a COMMON block that contains information of the integration procedure. RWORK is an array that contains real-valued information such as step-sizes and the Nordsieck history array at the current time. IWORK is an integer array that contains data such as the current method's order and the number of function evaluations. The COMMON block gives global access to variables whose values must be preserved between calls e.g. methods coefficients.

When LSODAR is used in the classical way it is expected that RWORK, IWORK and the COMMON block are not modified outside of LSODAR. The way we implemented the RK starter requires these arrays to be modified according to the Nordsieck history data obtained from the starter and to possibly alter the method coefficients in the COMMON block (see Box 6 in (3.5)).

3.6.3 Numerical experiments

We solved three problems using the RK starters previously described. Our focus was on the performance of the RK starters. We evaluated the strengths and weaknesses of each starter in comparison to other starting schemes. Numerical results of two models, bouncing ball and pendulum are presented in Paper I and Paper II.

In the bouncing ball example as well as the pendulum, Using the last step-size is clearly a good idea as half of the occurred events are not a discontinuity. They are just technical stops to activate the switching function. Thus, taking the old step-size and order is the best choice.

As a third example we present a woodpecker toy [57] gliding down a bar that is another model of motion with discontinuities. We consider the model [3] with

impacts and without friction.

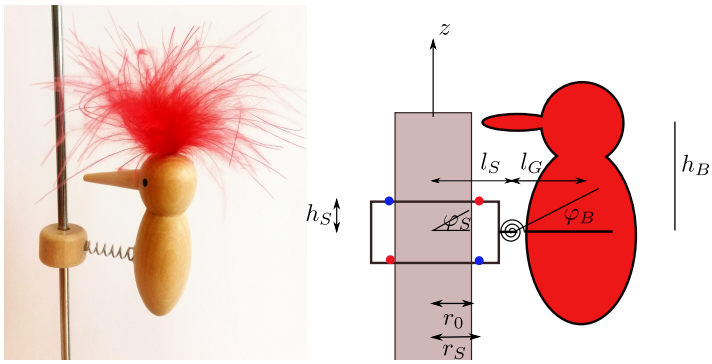

Figure 3.6: The woodpecker toy

The toy consists of a sleeve, a spring and the woodpecker (see (3.6)). The hole in the sleeve is slightly larger than the diameter of the bar and leads to the constraints, which are activated and deactivated under certain geometrical conditions. The bird is connected to the sleeve by a spring without damping. The sleeve has two degrees of freedom, its rotation ϕ_S and its vertical translation z.

The bird has one degree of freedom, the rotation relative to the sleeve, ϕ_B. Finally the bird's beak can hammer the bar and cause an impact. The model can be formulated in three states. The sleeve can fall down freely in State I, so there is no blocking. In State II the sleeve blocks at lower right and upper left corner thus, the constraint forces λ_1 and λ_2 get nonzero value. Finally in State III, the sleeve gets blocked again at the lower left and upper right corner, so λ_1 and λ_2 are nonzero.

The initial conditions of the model are,

$$z = 0, \quad \phi_S = 0, \quad \phi_B = -0.6, \quad \dot{z} = 0, \quad \dot{\phi}_S = 0, \quad \dot{\phi}_B = 0,$$

and the sleeve is in State I, so it can move down freely.

Starter	Classic	RK $\mathscr{F}_1$	RK $\mathscr{F}_2$	RK $\mathscr{F}_3$
# steps	2943	2906	3011	3065
# function evals	6368	6161	6070	5926
# event function evals	3795	3740	3887	3966
# jacobian evals	57	13	31	17

Table 3.2: Run time statistics for the woodpecker model with relative tolerance set to 10^{-8} and the initial step equal to the last successful step before the event.

The number of detected events for all the starters in Table (6.8) is 93. The simulation result in Table (3.2) reveals that the number of function evaluations for RK starter $\mathscr{F}_1$, $\mathscr{F}_2$ and $\mathscr{F}_3$ are slightly less than the self-starting scheme of LSODAR.

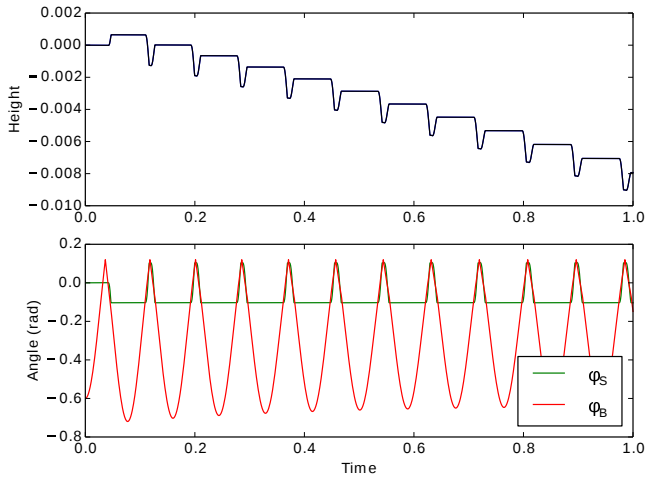

Figure 3.7: Simulation results of the woodpecker problem [3] for RK starter $\mathscr{F}_2$ and self-starting algorithms in LSODAR. The results are almost identical. The absolute and relative tolerance is set to 10^{-8} and simulation time is 1 second.

The RK starter $\mathscr{F}_2$ often uses order 3 or higher and takes mostly larger step-sizes compared to the other starters $\mathscr{F}_1$ and $\mathscr{F}_3$ (see Figure (3.9) and 3.8).

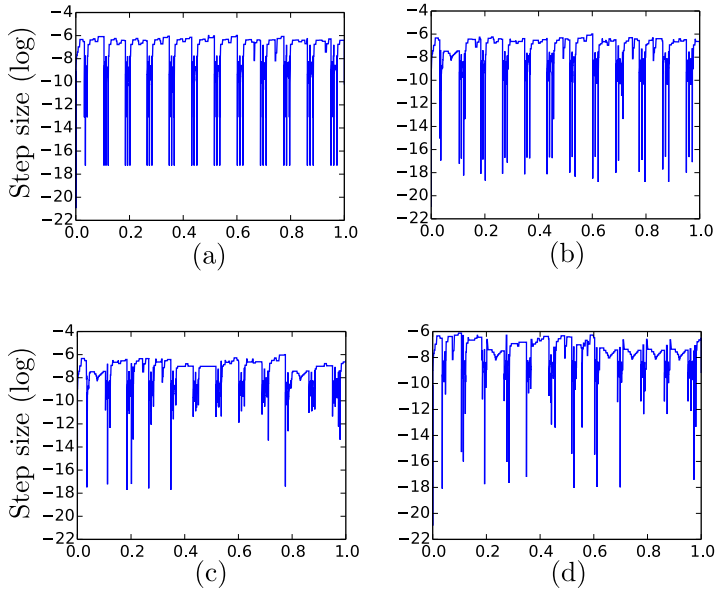

Figure 3.8: Comparison of the step-size vs. simulation time (in seconds) in logarithmic scale for (a) Classical starter, (b) RK starter $\mathscr{F}_1$, (c) RK starter $\mathscr{F}_2$ and (d) RK starter $\mathscr{F}_3$. the simulation time is 1s and the initial step-sizes for restarting in (b), (c) and (d) is taken from the last successful step before the events.

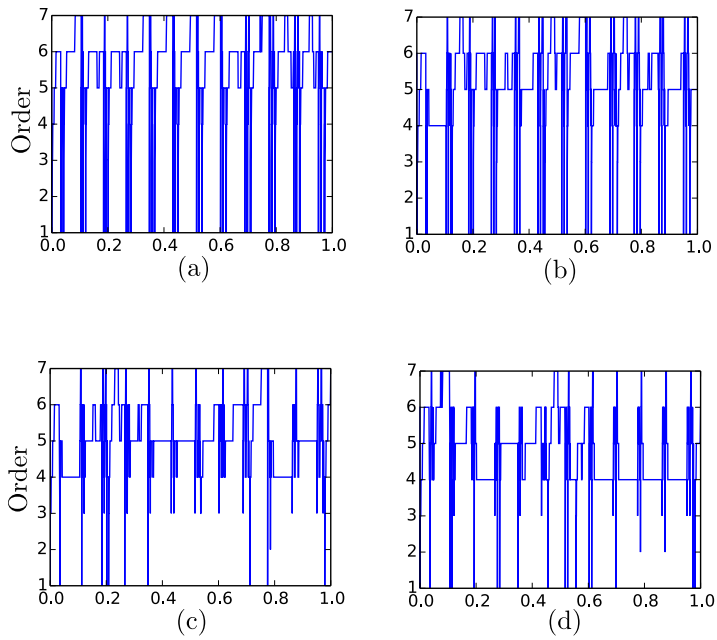

Figure 3.9: Comparison of the order vs. simulation time (in seconds) for (a) Classical starter, (b) RK starter $\mathscr{F}_1$, (c) RK starter $\mathscr{F}_2$ and (d) RK starter $\mathscr{F}_3$. the simulation time is 1s and the order for restarting in (b), (c) and (d) is taken from the last successful order before the events.

Chapter 4

Adaptive $\beta-$blocked multistep methods

The theory behind the numerical solution of ODEs is well-understood 250 years after the first attempt by Euler in *Institutionum calculi integralis* [30] and now robust software is available to solve a wide variety of ODE models. Many problems in constrained mechanical systems and electrical circuits lead to systems of ordinary differential equations (ODEs) where the simultaneous solution of algebraic equations is required each time that the differential part is to be evaluated. Theses systems, that are a combination of differential and algebraic equations are called *differential algebraic equations* (DAEs). In a mechanical systems context they are often called ODEs with constraints.

The early idea of solving DAEs with numerical methods was introduced by Gear [32]. He observed that BDF methods can be applied to solve DAEs. Later, because of numerical problems in the solution of some mechanical systems modeled by DAEs [12], new methods and techniques were investigated.

In the present chapter, we briefly explain one of these techniques, called $\beta-$*blocking*, and then we introduce a polynomial formulation of $\beta-$blocked multistep methods. This new formulation allows variable step-size implementation by construction. The idea of $\beta-$blocking multistep methods was first introduced about twenty years ago and since then, there was no successful attempt to introduce a variable step-size formulation of $\beta-$blocked methods. Thus our goal is to construct adaptive singular and regular $\beta-$blocked multistep methods for the solution of index-2

Euler-Lagrange differential algebraic equations.

In Section 4.1 we first introduce some basic theory on differential algebraic equations and in Section 4.2 we look at the $\beta-$blocking technique that makes it possible to apply integrators other than BDF to solve specified classes of DAEs. Finally in Section 4.4, we present a technique to construct variable step-size $\beta-$blocked multistep methods based on a parametric formulation.

4.1 Background

A differential algebraic equation in its fully implicit form is

$$F(t, x, \dot{x}) = 0 \tag{4.1}$$

with $\frac{\partial F}{\partial \dot{x}}$ singular along the solution. It is difficult to find an encompassing technique for solving such general problems. The particular structure of a DAE must be taken into account in order to develop successful methods. There are several engineering applications that lead to an autonomous semi-explicit DAEs model of the form

$$\begin{aligned} \dot{x} &= f(x, y) \\ 0 &= g(x, y) \end{aligned} \tag{4.2}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$. The derivatives of some variables are not expressed explicitly in DAEs. In fact, derivatives of some of the dependent variables, denoted here by y, typically do not appear in the (4.2). Such variables are usually called algebraic variables while the others, denoted here by x, are called differential variables.

Several thorough studies have been done in the numerical treatment of DAEs. Among many others, Runge–Kutta methods are discussed by Petzold [67], Hairer et al. [41], Kvaerno [55], Small [76] and multistep methods are studied by Lötstedet et al. [60, 69], Brenan et al. [19],Führer et al. [31], Arévalo [6], Söderlind [77], März [63], Gupta et al. [34] and Arnold [16]. Some of these methods are implemented in extensively used solvers such as IDA/DASSL (BDF methods) [66] and RADAU5 (implicit Runge–Kutta methods) [43].

The differentiation index quantifies the level of difficulty involved in solving a given DAE. This index measures the minimal number of differentiations required to

transform a DAE into an explicit ODE. It is known that DAEs can be difficult to solve when their index is greater than one [29].

Example 3 *[29] Consider the following equations*

$$\dot{x}_1 = x_3 \tag{4.3}$$
$$\dot{x}_2 = x_4 \tag{4.4}$$
$$\dot{x}_3 = -yx_1 \tag{4.5}$$
$$\dot{x}_4 = -yx_2 - 9.81 \tag{4.6}$$
$$0 = x_1^2 + x_2^2 - 1 \tag{4.7}$$

The differential equations (4.3), (4.4), (4.5) *and* (4.6) *with constraint* (4.7) *form an index 3 DAE system. If we replace constraint* (4.7) *by its derivative* (4.8) *we obtain the index-2 DAEs,*

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= -yx_1 \\
\dot{x}_4 &= -yx_2 - 9.81 \\
0 &= x_1x_3 + x_2x_4
\end{aligned} \tag{4.8}$$

Also if we replace constraint (4.7) *by its second derivative* (4.9) *we obtain the index 3 DAEs,*

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= -yx_1 \\
\dot{x}_4 &= -yx_2 - 9.81 \\
0 &= x_3^2 + x_4^2 - y(x_1^2 + x_2^2) - 9.81x_2
\end{aligned} \tag{4.9}$$

Note that (4.9) *implicitly defines* y*, and differentiating this expression leads to a differential equation in* y*.*

Thus, an obvious remedy to high index DAEs would be to reduce the index of the model by differentiating the constraints. However, index reduction may cause an instability of the analytical solution because the constraints may not be fulfilled. This instability is called *drift-off* effect. During the numerical integration, roundoff and truncation errors will accumulate and grow like a polynomial so that the solution is no longer on the constraint manifold [34].

Some remedies were suggested by Baumgarte [17] and others [72, 28] to overcome this instability due to drift-off effect. Baumgarte proposed stabilizing the problem by replacing the constraint with a linear combination of all three of them in index 1, index-2 and index-3 formulations.

Let us define an autonomous DAE system of differential index at least 2,

$$\begin{aligned} \dot{x} &= f(x, y) \\ 0 &= g(x) \end{aligned} \tag{4.10}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$.

The Baumgarte stabilization for index-2 DAEs uses

$$0 = \dot{g} + \gamma g \tag{4.11}$$

instead of constraint in index-2 form that is $\dot{g}(x) = 0$. Then it is required to find γ such that the characteristic polynomial $x + \gamma = 0$ has negative roots with negative real parts in order to have an stable underlying ODE [56, 20]. Similarly, Baumgarte stabilized index 3 DAEs with constraint

$$0 = \ddot{g} + \gamma_1 \dot{g} + \gamma_2 g \tag{4.12}$$

where γ_1 and γ_2 have to be determined such that the polynomial $x^2+\gamma_1 x+\gamma_2 = 0$ has roots with negative real parts.

We are going to look at the appropriate numerical methods to solve high index DAEs without changing the problem while the Baumgarte stabilization changes the model and add some parameters to the DAE model.

Coordinate projection is another technique [72, 28] that avoids the drift-off effect. The idea is to project the attained solution back to the constraints manifold after completing each step. Consider a consistent solution (x_{n-1}, y_{n-1}) of an index-2 DAE (4.10) at time t_{n-1} and suppose the numerical method advances the solution of the index reduced system from (x_{n-1}, y_{n-1}) to $(\hat{x}_n, y_n)$. Then the projected values are defined as the solution of

$$\begin{aligned} &\min_{x_n} \|\hat{x}_n - x_n\|_2 \\ &\quad \text{s.t.} \quad g(x_n) = 0 \end{aligned} \tag{4.13}$$

The projected values, (x_n, y_n) now satisfy the constraint, Figure 4.1. This technique can be applied to any integration method.

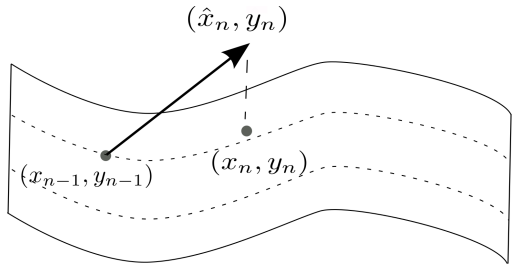

Figure 4.1: The solution $(\hat{x}_n, y_n)$ is projected to (x_n, y_n) on the constraint manifold.

The equations of motion of a mechanical system in index-2 Euler-Lagrange form are

$$\begin{aligned} \dot{p} &= v \\ M\dot{v} &= f(t,p,v) - G(p)^{\mathrm{T}}\lambda \\ 0 &= g(p) \end{aligned} \tag{4.14}$$

where p, v are position and velocity variables; λ is a Lagrange multiplier. Furthermore, $G(p) = \partial g/\partial p$ and $G(p)G(p)^{\mathrm{T}}$ is invertible. System (4.14) has a semi-explicit form where the algebraic variables, λ, appear linearly in the equations.

We can rewrite system (4.14) as an autonomous system in condensed form

$$\begin{aligned} \dot{x} &= f(x) - G(x)^{\mathrm{T}}\lambda \\ 0 &= g(x) \end{aligned} \tag{4.15}$$

where $x = [p \;\; v]$ and $G(x) = [0 \;\; G(p)]$. A system of ODEs can always be initialized by giving initial values at the starting time. However, the initial values for DAEs such as equations (4.2) have to satisfy the algebraic and differential equations simultaneously [23, 22]. The initial values that satisfy equations (4.2) are then called consistent initial values. Furthermore, high index DAEs have hidden algebraic constraints that affect the choice of initial values for the model. Thus the consistent initial values must be calculated before relaxing the constraints by index reduction.

Example 4 *[29] Consider the following DAE*

$$\begin{aligned} \dot{x}_1 &= f_1 - x_3 \\ \dot{x}_2 &= f_2 - x_1 \\ 0 &= x_2 - f_3 \end{aligned} \tag{4.16}$$

The consistent initial value has to satisfy the constraint so that $x_2(t) = f_3(t)$. *This implies the hidden constraints on* x_1 *and* x_3.

$$\begin{aligned} f_2 - x_1 &= \dot{f}_3 \\ f_1 - x_3 &= f_2 - \ddot{f}_3 \end{aligned} \tag{4.17}$$

The solution of (4.16) *is given as*

$$\begin{aligned} x_1 &= f_2 - \dot{f}_3 \\ x_2 &= f_3 \\ x_3 &= f_1 - \dot{f}_2 + \ddot{f}_3 \end{aligned} \tag{4.18}$$

Thus, there is no freedom to choose the initial values.

4.2 Regular and singular $\beta-$blocked multistep methods

It is known from Theorem 3.6 in [44] that the BDF discretization of index-2 DAEs (4.15) is stable while the Adams-Moulton discretization is not [44]. The crucial stability issue arises from the roots of the σ polynomial (2.5), that must lie inside the unit circle. Arévalo et al. [9] suggested to block this instability by modifying the discretization of the Lagrange multipliers and called the technique $\beta-$*blocking*. For a linear index-2 Euler-Lagrange DAE discretized by multistep methods, the $\beta-$blocking technique moves the eigenvalues of its one-step form of the discretization to the unit disc. This is done by introducing an additional polynomial $\tau(\zeta) = \sum_{i=0}^{k} \gamma_i \zeta^i$ acting only on algebraic variables,

$$\begin{aligned} \rho x_n &= h\sigma(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n) - hG^{\mathrm{T}}(x_n)\tau\lambda_n \\ 0 &= g(x_n). \end{aligned} \tag{4.19}$$

The stabilizing operator τ is chosen so that $\sigma + \tau$ satisfies the strict root condition and also $\tau\lambda_n = \mathcal{O}(h^k)$. By choosing $\tau = c\nabla^k$ both objectives may be met. The free parameter c must be chosen so that the polynomial $\sigma(\zeta) + \tau(\zeta)$ has all its roots inside the unit circle. Depending on whether $\sigma + \tau$ is an implicit or an explicit operator, a $\beta-$blocking method is regular or singular. It is shown [8] that

Method	AM1	AM2	AM3	dcBDF1	dcBDF2	dcBDF3	dcBDF4	dcBDF5
c	0.5	0.146	0.092	0.5	0.33	0.25	0.2	0.16

Table 4.1: Suggested value for parameter c in $\tau = c\nabla^k$ for Adams-Moulton(AM) and dcBDF(IDC) methods.

for $k-$step Adams-Moulton methods with $k \leq 3$, the parameter c that satisfies the strict root condition may be chosen within an interval. The suggested parameter c that locates the roots of $\sigma + \tau$ closest to the origin is shown in Table (4.1), see [10].

Unfortunately, it is not possible to find such an interval for c to $\beta-$block $k-$step Adams-Moulton methods with $k \geq 4$. However, we can choose a particular value $c = \beta_k$ such that $\sigma + \tau$ becomes of one order less. With $c = \beta_k$ the algebraic variables are treated by an explicit method and thus this technique is called singular $\beta-$blocking [10]. Singular $\beta-$blocking allows the stabilization of methods such as Adams-Moulton methods up to $k \leq 6$.

The stability of the overall discretization is recovered but the price is a reduced order of convergence of the algebraic variables [8]. Theorem 1 gives the convergence result for $\beta-$blocked multistep methods. Later, this theorem will be used to construct the polynomial formulation of $\beta-$blocked maximal order multistep methods.

Theorem 1 *[8] Consider the index-2 problem* (4.15) *discretized by* (4.19) *with starting values* $x_i - x(t_i) = \mathcal{O}(h^p)$ *and* $\lambda_i - \lambda(t_i) = \mathcal{O}(h^k)$ *for* $i = 0, \ldots, k-1$*, where* $p = k$ *or* $p = k+1$ *is the order of the pair* (ρ, σ)*. Furthermore, let* τ *be chosen such that* $\sigma + \tau$ *has roots strictly inside the unit circle and* $\tau y(t_n) = \mathcal{O}(h^k)$ *for any sufficiently smooth function. Then the discretization* (4.19) *is convergent, i.e. for* $t_n = nh$ *constant,*

$$x_n - x(t_n) = \mathcal{O}(h^p) \text{ and } \lambda_n - \lambda(t_n) = \mathcal{O}(h^k).$$

As a result, a $k-$step Adams-Moulton methods have order $p = k+1$ in the differential variables, and $p = k$ in the algebraic ones.

The $\beta-$blocked multistep discretization of problem (4.15) can be written as

$$\begin{aligned} h^{-1}\rho x_n &= \beta_k(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n) + \tilde{\sigma}(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n) - G^{\mathrm{T}}(x_n)\tau\lambda_n \\ 0 &= g(x_n) \end{aligned} \tag{4.20}$$

where $\tilde{\sigma}x_n = (\sigma - \beta_k)x_n$ or

$$\begin{aligned} h^{-1}\rho x_n &= \beta_k(f(x_n) - G^{\mathrm{T}}(x_n)(1+\beta_k^{-1}\tau)\lambda_n) + \tilde{\sigma}(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n) \\ 0 &= g(x_n) \end{aligned} \tag{4.21}$$

To illustrate the difference between regular and singular $\beta-$blocking, we consider the one step Adams-Moulton method as an example.

Example 5 *The trapezoidal scheme applied on* (4.15) *has the form*

$$\begin{aligned} \nabla x_n &= \frac{h}{2}\big(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n + f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1}\big) \\ 0 &= g(x_n) \end{aligned} \tag{4.22}$$

It is obvious that the generating polynomial $\sigma(\zeta) = \frac{1}{2}(\zeta+1)$ has a root on the unit disc. For regular $\beta-$blocking with $\tau = \frac{1}{2}\nabla$ we have

$$\begin{aligned} \nabla x_n &= \frac{h}{2}\big(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n + f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1}\big) - \frac{h}{2}G^{\mathrm{T}}(x_n)\nabla\lambda_n \\ 0 &= g(x_n) \end{aligned} \tag{4.23}$$

We note that given x_{n-1} and λ_{n-1}, we can compute x_n and λ_n simultaneously.

Singular $\beta-$blocking makes $\sigma+\tau$ explicit, and thus takes $\tau = -\frac{\nabla}{2}$. We get

$$\begin{aligned} \nabla x_n &= \frac{h}{2}(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_{n-1} + f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1}) \\ 0 &= g(x_n) \end{aligned} \tag{4.24}$$

The effect of singular $\beta-$blocking is that the algebraic variable is treated by an explicit method since there is no λ_n in system (4.24)*. In the first step with a given x_0, we solve the system simultaneously for x_1 and λ_0 and do the same for the coming steps, that is, solve for x_n and λ_{n-1}.*

The former stabilizing techniques are only studied for fixed step-size multistep methods [8, 9, 10]. Adaptive $\beta-$blocked methods may reduce the computational efforts in the presence of a good error estimator and controller. As a proof of concept, we formulate $\beta-$blocked methods in a polynomial form that makes such methods adaptive. Our goal is to represent the variable step-size polynomial formulation of the discretized system (4.19). Thus, we start by looking at a polynomial formulation for DAEs.

4.3 A polynomial formulation for index-2 DAEs

The new polynomial formulation of multistep methods in Section 2.2.1 is developed for solving a system of ODEs [13]. Thus, to applying this formulation to solve DAEs some modification of the slack conditions is needed. In the following section, we look for a parametric formulation (see Section 2.2.1) of $(k, k+1)$ methods suitable for solving index-2 DAEs.

The straightforward polynomial formulation of a $k-$step method of order $k+1$ applied to (4.15) is

$$\begin{cases} P_n'(t_n) = f(P_n(t_n)) - G^{\mathrm{T}}(P_n(t_n))\lambda_n \\ P_n(t_{n-1}) = x_{n-1} \\ P_n'(t_{n-1}) = f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1} \\ \cos\theta_{j-1}(P_n(t_{n-j}) - x_{n-j}) + h_{n-j}\sin\theta_{j-1} \\ \qquad (\dot{P}_n(t_{n-j}) - f(x_{n-j}) - G^{\mathrm{T}}(x_{n-j})\lambda_{n-j}) = 0 \\ g(P_n(t_n)) = 0 \end{cases} \tag{4.25}$$

where $j = 2, \ldots, k$ and $x_n := P_n(t_n)$. The unknowns of the system (4.25) at each step are the coefficients of the polynomial P_n, and the Lagrange multiplier λ_n. The $(k+1)n_x + n_\lambda$ unknowns that may be determined by solving system (4.25).

Example 6 *Consider the* $1-$*step Adams-Moulton method of order* 2*, AM1, applied to* (4.15) *with fixed step-size. The AM1 in parametric form is constructed as a polynomial* $P_n \in \Pi_2$ *with*

$$P_n(t) = \frac{at^2}{h^2} + \frac{bt}{h} + c$$

over $[t_{n-1}, t_n]$ *that defines* $x_n := P_n(t_n)$ *with the following conditions,*

$$\begin{cases} P_n'(t_n) = f(P_n(t_n)) - G^{\mathrm{T}}(P_n(t_n))\lambda_n \\ P_n(t_{n-1}) = x_{n-1} \\ P_n'(t_{n-1}) = f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1} \\ g(P_n(t_n)) = 0 \end{cases} \tag{4.26}$$

By solving system (4.26) *for the unknown coefficients* a, b *and* c *we get*

$$c = x_{n-1} \tag{4.27}$$

$$b = f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1} \tag{4.28}$$

$$a = \frac{f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n - (f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1})}{2h} \tag{4.29}$$

which by substituting these coefficients of polynomial P_n in $x_n := P_n(t_n)$ we obtain

$$x_n = x_{n-1} + \frac{h}{2}(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n + (f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1}) \tag{4.30}$$

$$0 = g(x_n) \tag{4.31}$$

that is equivalent to the classical formulation of AM1 for (4.15).

Adams-Moulton methods are well known zero-stable $k-$steps method of order $k+1$ with formulation (4.25) when $\theta_i = \frac{\pi}{2}$ for $i = 1, \ldots, k-2$. However their resulting discretization of index-2 DAEs is unstable, see Section 4.2.

The main questions are how to parametrize $\beta-$blocked multistep methods and how to find a polynomial corresponding to the stabilizer τ.

4.4 Parametrized $\beta-$blocked multistep methods

In order to parametrize a $k-$step method of order $k+1$ (see Theorem 1), we construct for every interval $[t_{n-1}, t_n]$ a pair of polynomials, $P_n \in \Pi_{k+1}$ and $Q_n \in \Pi_k$ that interpolate differential and algebraic variables, respectively.

Formulation (4.21) suggests a modification of the slack condition that interpolates the vector field value at t_n, that is, $s'_n = 0$. Further, the stabilizer τ acts only on algebraic variables so it is enough to modify the polynomial Q_n in order to obtain this effect. As we now wish to consider τ as a variable step-size operator, we replace $c\nabla^k\lambda_n$ with $ch_{n-1}^k Q_n^{(k)}(t_n)$, but note that $Q_n^{(k)}(t)$ is constant as $Q_n \in \Pi_k$.

4.4.1 Parametrized regular $\beta-$blocked methods

The method polynomials P_n and Q_n for problem (4.15) are constructed by solving the system

$$\begin{cases} P_n'(t_n) = f(P_n(t_n)) - G^{\mathrm{T}}(P_n(t_n))(Q_n(t_n) + \hat{c}\, h_{n-1}^k Q_n^{(k)}(t_n)) \\ P_n(t_{n-1}) = x_{n-1} \\ P_n'(t_{n-1}) = f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})\lambda_{n-1} \\ Q_n(t_{n-1}) = \lambda_{n-1} \\ \cos\theta_{j-1}(P_n(t_{n-j}) - x_{n-j}) + h_{n-j}\sin\theta_{j-1} \\ \qquad (\dot{P}_n(t_{n-j}) - f(x_{n-j}) + G^{\mathrm{T}}(x_{n-j})\lambda_{n-j}) = 0 \\ Q_n(t_{n-j}) = \lambda_{n-j} \\ g(P_n(t_n)) = 0 \end{cases} \tag{4.32}$$

for $j = 2, \ldots, k$. The new values are obtained by setting $x_n := P_n(t_n)$ and $\lambda_n := Q_n(t_n)$. Note that we have no derivative slack for polynomial Q_n in formulation (4.32) since Q_n interpolates the algebraic variables.

Obtaining the value of parameter $\hat{c}$

In order to solve system (4.32), we have to determine the value of $\hat{c}$. For fixed step-size regular $\beta-$blocking, the value of c was determined so that $\sigma + \tau$ satisfies the strict root condition (see Table 4.1). for this formulation we take $\hat{c} = c\beta_k^{-1}$. We can find an expression for the leading term of the σ polynomial, β_k, in terms of method parameters θ_j.

Example 7 *Consider a $2-$step method of order 3. We can obtain an expression as a function of θ_1 for β_2. The order conditions and the parametric equivalence equation* (6.39) *result in the system $j = 2, \ldots, k$.*

$$\begin{cases} \tan\theta_1\ \alpha_0 - \beta_0 = 0 \\ \alpha_0 + \alpha_1 + 1 = 0 \\ \alpha_1 + 2 - \beta_0 - \beta_1 - \beta_2 = 0 \\ \alpha_1 + 4 - 2(\beta_1 + 2\beta_2) = 0 \\ \alpha_1 + 8 - 3(\beta_1 + 4\beta_2) = 0 \end{cases} \tag{4.33}$$

where we get $\beta_2 = \frac{5\sin\theta_1 + 2\cos\theta_1}{12\sin\theta_1 + 5\cos\theta_1}$ and thus $\hat{c} = c\frac{12\sin\theta_1 + 5\cos\theta_1}{5\sin\theta_1 + 2\cos\theta_1}$. For AM2, $c = 0.146$ (see Table 4.1) and $\theta_1 = \frac{\pi}{2}$, therefore $\hat{c} = 0.35$.

Note that the parameter $\hat{c}$ is not unique since the value of c may be chosen from an interval such that $\sigma + \tau$ satisfies the strict root condition.

In general, for all fixed step-size $k-$step methods of order $p = k + 1$ the leading coefficient of σ, namely β_k, is obtained by solving a linear system $Lv = w$ where matrix L consists of 4 sub-matrices

$$L = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{4.34}$$

of the forms

$$A_{(p+1)\times(p-2)} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & 1 & 2 & \dots & k-1 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1^p & 2^p & \dots & (k-1)^p \end{pmatrix}, \tag{4.35}$$

$$B_{(p+1)\times(p-1)} = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & -1 & -1 & \dots & -1 \\ 0 & -2 & -2 \cdot 2 & \dots & -2k \\ 0 & -3 & -3 \cdot 2^2 & \dots & -3k^2 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & -p & -p \cdot 2^{p-1} & \dots & -p(k-1)^{p-1} \end{pmatrix}, \tag{4.36}$$

$$C_{(p-2)\times(p-2)} = \begin{pmatrix} \sin\theta_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & \sin\theta_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \sin\theta_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & 0 \\ 0 & 0 & 0 & \dots & \sin\theta_{k-1} & 0 \end{pmatrix}, \tag{4.37}$$

$$D_{(p-2)\times(p-1)} = \begin{pmatrix} -\cos\theta_1 & 0 & \dots & 0 & 0 & 0 \\ 0 & -\cos\theta_2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & 0 & 0 \\ 0 & 0 & \dots & -\cos\theta_{k-1} & 0 & 0 \end{pmatrix}, \tag{4.38}$$

with a vector of unknowns $v = (\alpha_0, \alpha_1, \dots, \alpha_{k-1}, \beta_0, \dots, \beta_k)^{\mathrm{T}}$ and right hand-side vector $w = (-1, -k, \dots, -k^p, 0, \dots, 0)^{\mathrm{T}}$.

We use the parameter c that is derived for fixed step-size formulation such that $\sigma(\zeta) + c\nabla^k(\zeta)$ satisfies the strict root condition. In the variable step-size formulation with smooth changes of step-sizes we have $\frac{h_n}{h_{n-1}} \approx 1$. Thus the multistep

coefficients, that are continuously dependent on the step-size ratios are close to the fixed step-size coefficients. Stability at infinity will be preserved if the change of step-sizes is small and smooth [15].

4.4.2 Parametrized singular $\beta-$blocked methods

The singular $\beta-$blocked multistep methods are parametrized so that the algebraic variables are treated by an explicit method. This causes some notable differences with regard to the formulation of regular stabilized methods.

As in the case of parametrized regular $\beta-$blocked methods, we construct for every interval $[t_{n-1}, t_n]$ a pair of polynomials, $P_n \in \Pi_{k+1}$ and $Q_n \in \Pi_k$ that interpolate differential and algebraic variables, respectively (see Theorem 1). We take $\tau = -\beta_k \nabla^k$ for the singular $\beta-$blocking stabilizer (see Section 4.2). By substituting this τ in the discretization (4.21), we get

$$\begin{aligned} h^{-1}\rho x_n &= \beta_k(f(x_n) - G^{\mathrm{T}}(x_n)(1-\nabla^k)\lambda_n) + \tilde{\sigma}(f(x_n) - G^{\mathrm{T}}(x_n)\lambda_n) \\ 0 &= g(x_n) \end{aligned} \tag{4.39}$$

The interpolation conditions require that $Q_n(t_{n-j}) = \lambda_{n-j}$ for $j = 2, \ldots, k$. The term $(1-\nabla^k)\lambda_n$ in (4.39) has the latest value λ_{n-1} thus the polynomial Q_n is degenerated to $Q_n \in \Pi_{k-1}$ while the consistency order of algebraic variables remains the same as for regular $\beta-$blocking.

The singular $\beta-$blocked method polynomials P_n and Q_n for problem (4.15) are constructed by solving the system

$$\begin{cases} P_n'(t_n) = f(P_n(t_n)) - G^{\mathrm{T}}(P_n(t_n)x_n)Q_n(t_n) \\ P_n(t_{n-1}) = x_{n-1} \\ P_n'(t_{n-1}) = f(x_{n-1}) - G^{\mathrm{T}}(x_{n-1})Q_n(t_{n-1}) \\ \cos\theta_{j-1}(P_n(t_{n-j}) - x_{n-j}) + h_{n-j}\sin\theta_{j-1} \\ \qquad (\dot{P}_n(t_{n-j}) - f(x_{n-j}) - G^{\mathrm{T}}(x_{n-j})\lambda_{n-j}) = 0 \\ Q_n(t_{n-j}) = \lambda_{n-j} \\ g(P_n(t_n)) = 0 \end{cases} \tag{4.40}$$

for $j = 2, \ldots, k$. The new values are obtained by setting $x_n := P_n(t_n)$ and $\lambda_{n-1} := Q_n(t_{n-1})$.

4.5 Numerical results

In this section we investigate the performance of parametric regular and singular $\beta-$blocked multistep methods. We consider two examples and different tests are carried out.

4.5.1 Linear model

Example 8 *Consider the following index-2 system*

$$\begin{aligned} \dot{x}_1 &= 1 - \lambda \\ \dot{x}_2 &= -x_2 + \lambda \\ 0 &= x_1 - x_2 \end{aligned} \tag{4.41}$$

with initial values $x_1(0) = 0$, $x_2(0) = 0$, $\lambda(0) = 0.5$ *and analytic solution* $x_1(t) = 1 - e^{-t/2}$, $x_2(t) = 1 - e^{-t/2}$, $\lambda = 1 - \frac{e^{-t/2}}{2}$.

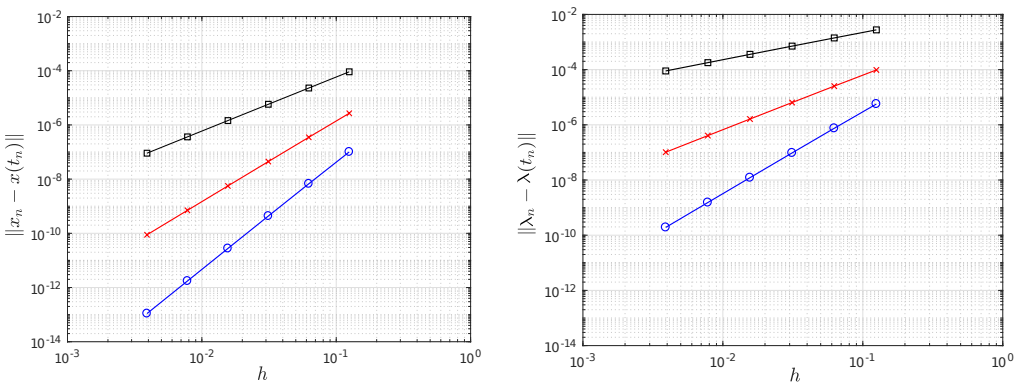

Figure 4.2: Global error versus step-size for regular $\beta-$blocked AM1 (□), AM2 (×) and AM3 (○)

For the first experiment, we want to show that the parametric formulation of $\beta-$*blocked multistep methods has the expected consistency order in the fixed step-size case. Thus, we solve problem* (4.41) *with fixed step-size regular* $\beta-$*blocked AMk methods, taking* $h = 1/N$ *with* $N = 8, 16, 32, 64, 128$ *and* 256. *The global error in non-algebraic variables* x_1, x_2 *and algebraic variable* λ *were measured in* L^1 *norm on the interval* $[0, 5]$ *by taking the mean global error over all steps. This norm flattens the minor fluctuation that often occur when errors are measured at a single point. Figure* (4.2) *shows that the order is at least* $k + 1$, *for differential variables and* k *for algebraic one.*

In the next test, we apply two adaptive $\beta-$blocked multistep methods of order 4, the $3-$step dcBDF method and the $3-$step AM method to solve problem (4.41). *Although the step-sizes that are taken by AM method are often larger than the steps taken by dcBDF method, see Figure 4.4, the global errors of the numerical solution x_1, obtained by AM3 is around 10^{-8} while for dcBDf method, it is around 10^{-7}. It is well known that AM methods have smaller error constant compare to dcBDF methods. The error is computed at each step for variables x_1 (the solution of x_1 is equal to x_2) and λ by In*

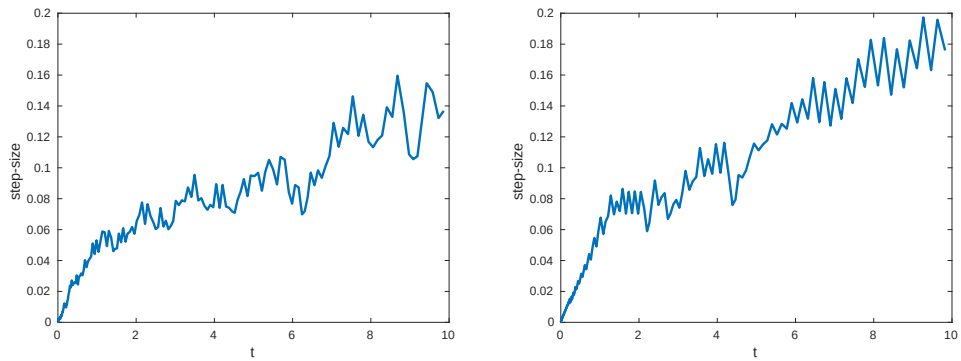

Figure 4.3: The step-size changes versus time of $3-$step dcBDF method(left plot) and $3-$step AM method (right plot), both of order 4 applied on model (4.41).

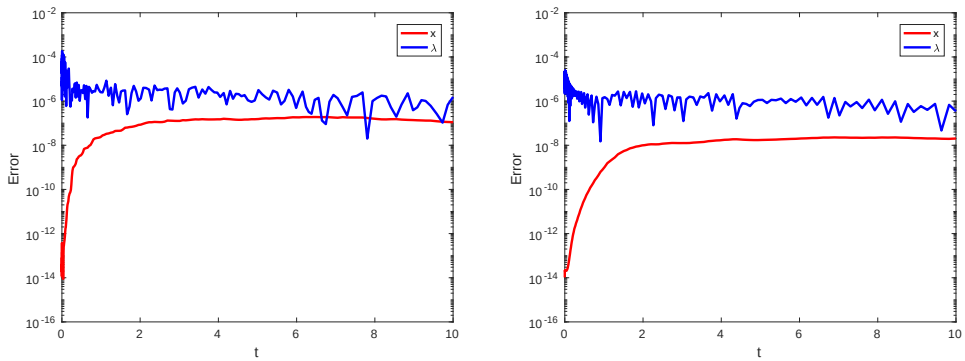

Figure 4.4: Global error of the solution of model (4.41) that is obtained by $3-$step dcBDF method(left plot) and $3-$step AM method (right plot).

the classical approach of controlling the step-sizes, the new step-size is calculated by,

$$h_n = \left(\frac{\text{Tol}}{e_{n-1}}\right)^{\frac{1}{p+1}} h_{n-1} \tag{4.42}$$

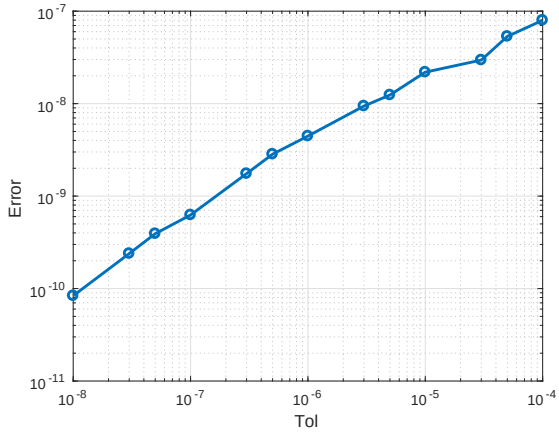

Figure 4.5: Changes of mean L^1 norm of error in x_1 versus tolerance is proportional for adaptive regular $\beta-$blocked AM3 applied on model (4.41).

where Tol *is a given tolerance and* e_{n-1} *is the local error of the current time step and* p *is the method's order. By taking logarithms from Equation* (4.42)*, we get*

$$\log e_{n-1} = \log \text{Tol} + (p+1)\log \frac{h_{n-1}}{h_n}. \tag{4.43}$$

If the step-size changes are small then their ratio $\frac{h_n}{h_{n-1}}$ *is close to one, so the second term in* (4.43) *becomes negligible. Hence in practice we can observe an error that is proportional to the tolerance. Figure* (4.5) *shows the tolerance proportionality to the global error in differential variables. The error is measured by taking the mean global error over all steps.*

4.5.2 Nonlinear model

Example 9 (Pendulum) *The equation of the pendulum in index-2 Euler-Lagrange form [29] is*

$$\begin{aligned} \dot{p}_1 &= v_1 \\ \dot{p}_2 &= v_2 \\ \dot{v}_1 &= -\lambda p_1 \\ \dot{v}_2 &= -\lambda p_2 - g \\ 0 &= p_1 v_1 + p_2 v_2 \end{aligned} \tag{4.44}$$

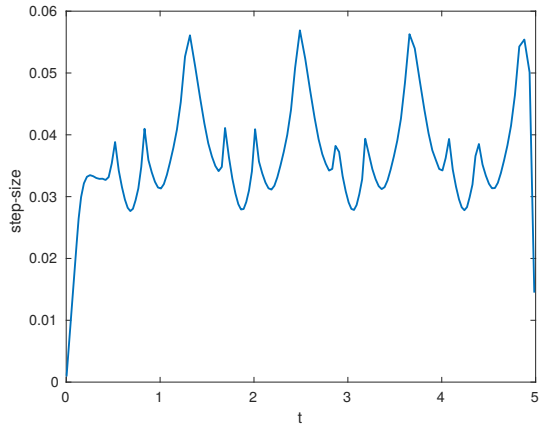

Figure 4.6: Step-size changes of the pendulum during the simulation exhibit a periodic pattern.

where $p_1,\ p_2$ *and* $v_1,\ v_2$ *are position and velocity variables, respectively. Here we assume that the mass of the pendulum and its length are one and the gravitational constant is* $g = 9.81$. *Consider the consistent initial values* $p_1(0) = 1, p_2(0) = v_1(0) = v_2(0) = \lambda = 0$.

The problem was solved by singular $\beta-$*blocked AM4 with* $Tol = 10^{-3}$, *initial step-size* $Tol = 10^{-3}$ *and simulation time* $5s$. *The step-size changes is illustrated in Figure* (4.6). *The simulation starts with the initial step-size* 10^{-3} *and controller increase the step-sizes gradually until it varies between* 0.03 *and* 0.06. *The step-sizes are varying periodically as the pendulum has the same dynamic at each period. Figure 4.7 shows that the tolerance is proportional to the absolute error of the solution to the pendulum equations. Although the analytic solution of the pendulum is not known, we know that the solution is periodic. Thus we compute the error at a specific time point where the pendulum completes a full period,* $t_f = 2.367841947461$. *The error for the state variables* $x = (p_1, p_2, v_1, v_2)$ *is approximated by* $\|x(t_f) - x_{t_f}\|$ *where* $x(t_f)$ *is equal to the initial value and* x_{t_f} *is the numerical solution for the given tolerance.*

4.6 Implementation

We implemented our parametric formulation of SSP and $\beta-$ blocked multistep methods in the adaptive multistep solver MODES [14]. We have briefly looked at MODES in Section 5.3. We have added two functions PolDAEr (regular $\beta-$blocking) and PolDAEs (Singular $\beta-$blocking) to MODES so that, for given

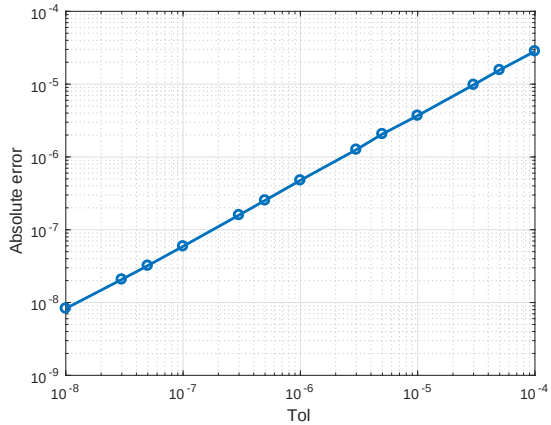

Figure 4.7: Changes of absolute error versus tolerance is proportional for the solution of pendulum that is obtained by adaptive singular $\beta-$blocked AM4.

parameters θ and c the solution at t_n is computed by solving the system of conditions, (4.32) and (4.40) for the coefficients of P_n. Furthermore, these two functions contain an iterative corrector, as is required it in the classical formulation of implicit methods.

For a given initial step-size, the adequate k initial values for initializing the $\beta-$blocked multistep methods are calculated by lower order $\beta-$blocked methods. The local error at each step is computed by comparing the value of the previous polynomial $P_{n-1}(t_n)$ and the value of the current one $P_n(t_n)$, only for the differential variables. Note that the algebraic variables have one order less than the state variables.

In the integration phase, we apply either PolDAEr with the given parameters θs and $\hat{c}$ or PolDAEs with the given parameters θs to compute the coefficients of polynomials P_n and Q_n. The other steps are similar to those we explained in Section 5.3.

Chapter 5

Parametrized multistep methods of lower orders

In this section we present a parametric formulation of a general explicit $k-$step method of order p where $p < k$. From now on, we denote a $k-$step method of order p by (k, p) method.

Although the stability region of explicit methods is smaller than the implicit methods, they allow for a direct computation of the solution from the known quantities. Even though implicit methods can take larger step-sizes due to larger stability region, extrapolating to the next step for the system that has changed in that time step while the numerical solvers information hasn't, may cause more computational efforts. Usually explicit methods are combined with the implicit solvers called *predictor-corrector methods* to counter this problem. The explicit method is used to predict the state of the system at the next time step. Then the implicit methods can use the predicted solution to evaluate the derivative and correct the predicted value.

In the context of SSP methods, explicit multistep methods are especially interesting because of direct approximation of the solution since applying iterative methods to solve a large nonlinear system of equations that arises from implicit time discretization of hyperbolic models would be expensive.

In order to formulate a general explicit (k, p) method with $p < k$, we need $p + 1$ interpolation conditions. There must be at least one slack present for each non-zero

pair (α_i, β_i), but having one slack condition for each point at $t_{n-i}, i = 1, \ldots, k$, might result in an overdetermined system. Therefore, we consider using linear combinations of the slack conditions. In the classical formulation of explicit multistep methods, a $k-$step, order p method is defined by its coefficients $\alpha_1, \ldots, \alpha_k$ (α_0 is normalized to 1), and $\beta_1, \ldots, \beta_k$, and satisfies $p + 1$ order conditions. We are looking for a parametric formulation of (k, p) methods that includes methods that can be obtained by fixing $2k - p - 1$ parameters, so that there is a one-to-one correspondence between the classical coefficients of a method and its parameters in the polynomial formulation. It is important to notice that the slack s_{n-k} must be present in the formulation to guarantee the prescribed number of steps.

We extended a fixed step-size method with given coefficients $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k$, in a way to preserve the pattern of zero coefficients. This implies that for any coefficients satisfying $\alpha_i = 0$ or $\beta_i = 0$, the corresponding state, y_{n-i}, or vector field, y'_{n-i}, values are not present in the formulation, respectively. Thus the interpolation conditions that define P_n cannot include s_{n-i} and s'_{n-i} and we must observe the following rules:

Procedure 1: Formulation of explicit methods of order $p < k$

To define the method polynomial $P_n \in \Pi_p$, use the following rules to set up the interpolation conditions:

- If $(\alpha_i, \beta_i) = (0, 0)$, do not include either s_{n-i} or s'_{n-i}.
- If $\alpha_i = 0, \beta_i \neq 0$, include only $s'_{n-i} = 0$.
- If $\alpha_i \neq 0, \beta_i = 0$, include only $s_{n-i} = 0$.
- For each $\alpha_i \neq 0$ and $\beta_i \neq 0$, include one of the following:

$$\begin{cases} s_{n-i} = 0 \\ s'_{n-i} = 0, \end{cases} \tag{5.1}$$

$$\text{or } s_{n-i} + h_{n-i}\tau_i s'_{n-i} = 0, \tag{5.2}$$

$$\text{or } \sum \left(\lambda_i s_{n-i} + h_{n-i}\tau_i s'_{n-i}\right) + s_{n-k} + h_{n-k}\tau_k s'_{n-k} = 0, \tag{5.3}$$

so that the total number of conditions adds up to $p + 1$.

We may have some freedom in choosing the interpolation conditions that define

P_n, but the following theorem, proved in Paper III, gives formulas that represent the equivalence relations between the method parameters and the coefficients of a classical multistep formula.

Theorem 2 *For a (k, p) method defined by $p + 1$ slack conditions chosen according to Procedure 1, the method parameters can be defined as follows:*

$$\begin{aligned} \tau_i &= \frac{\beta_i}{\alpha_i}, \quad && \textit{when (5.2) is used} \\ \tau_i &= \frac{\beta_i}{\alpha_k}, \quad \lambda_i = \frac{\alpha_i}{\alpha_k}, \quad && \textit{when (5.3) is used} \end{aligned} \tag{5.4}$$

Although it is not common to apply (k, p) methods with $p < k$, an important class of these methods that are designed to solve specific ODE models is introduced in Section 5.1.

5.1 Strong stability preserving methods

In the current section we study strong stability preserving (SSP) multistep methods that are a class of $k-$step methods with $p < k$ for $p \geq 2$.

In solving time-dependent partial differential equations (PDEs),

$$u_t = f(u)_x$$

it is common to discretize the spatial variables x, first to obtain a semi-discrete ODE model of the form

$$u_t = L(u), \quad u(t_0) = u_0$$

where L is a difference operator that arises from semi-discretizations of PDEs.

In fact, some special methods for time dependent hyperbolic PDEs have been developed to insure stability and avoid spurious oscillations of the numerical solutions during shocks. These methods were first developed by Shu and Osher [73] as *total variation diminishing* (TVD) methods. SSP methods are designed to preserve the structure of the models and the conservation laws. A standard objective is to construct semi-discretizations of hyperbolic conservation laws so that the total variation of discrete solutions does not increase in time.

These methods are also known in the literature as *contractivity preserving*[58], or *monotonicity preserving methods*[50].

For the sake of shock-capturing, the hyperbolic models are naturally semi-discretized by TVD finite difference of finite volume methods [62]. Unfortunately the order of these methods is limited to one or two [36]. Thus Essentially Non-Oscillatory (ENO) methods were developed [74, 46] that minimize the numerical oscillation around shocks and attain high order accuracy outside the discontinuities. The WENO scheme [45] is a variant of ENO methods especially suited for the problems containing both shocks and complicated smooth solution structures, such as compressible turbulence simulations. In this thesis our focus is on the time-stepping methods and we consider a semi-discretized hyperbolic model.

Gottlieb and Shu [38] highlighted the necessity of applying SSP methods to hyperbolic PDEs through the following example.

Consider Burgers equation,

$$u_t = (\frac{u^2}{2})_x \tag{5.5}$$

with initial function,

$$u(x,0) = \begin{cases} 1, & \text{if } x \leq 0 \\ -0.5, & \text{if } x > 0 \end{cases} \tag{5.6}$$

The model (5.5) is semi-discretized with a second order upwind spatial discretization, Monotone Upstream-centered Schemes for Conservation Laws (MUSCL) [81]. The MUSCL discretization is TVD under some restriction on time steps. Two second order time integrators are applied to solve the ODE arises from semi-discretization of Burgers equation.

In Figure 5.1, we can clearly see that the non-TVD result is oscillatory.

Strong stability preserving methods are a class of ODE solvers that can be one of two types: one-step or multistep methods. An important class of one-step SSP methods, Runge–Kutta methods, have been well-studied, see e.g, [37, 47, 39]. SSP multistep methods with uniform step-size are also well-studied in [58, 71, 37]. Nevertheless, there is only one study on variable step-size SSP multistep methods and it is on SSP explicit multistep methods of orders two and three [40]. The adaptive SSP explicit multistep methods of orders at least 3 have a complicated structure in the classical formulation of linear multistep methods. In [40] it is suggested to look for a new formulation that has a simple structure in order to develop higher order SSP explicit

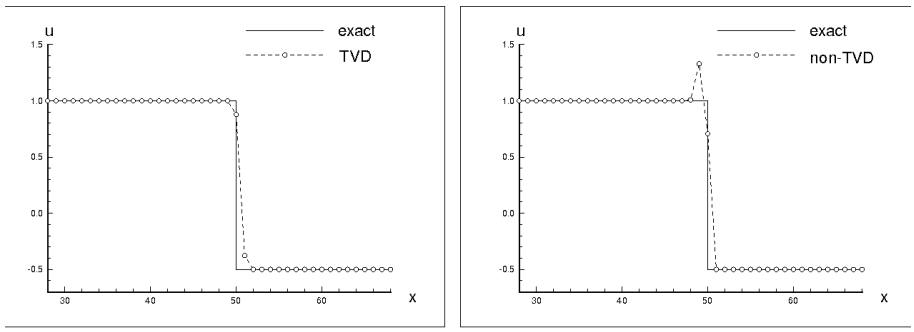

Figure 5.1: Second order TVD MUSCL spatial discretization. solution after 500 time steps. Left: TVD time discretization; Right: non-TVD time discretization [38].

multistep methods. In Paper III we focused our study on a new formulation of SSP explicit multistep methods supporting variable step-size by construction. The center part of this study is on the construction of particular explicit (k, p) methods with $p < k$ and the supplementary issues such as their efficiency or the required bounds on step-size ratios are not part of this study.

The setting for SSP methods can be summarized as follows.

Let us consider ODEs of the form

$$\dot{y} = F(y), \;\; y(t_0) = y_0, \;\; t \in [t_0, t_{\mathrm{f}}], \tag{5.7}$$

where $F : \mathcal{R}^m \to \mathcal{R}^m$.

The vector field F is assumed to have the property (5.8) for a given norm and a fixed step-size h.

$$\|y + hF(y)\| \leq \|y\| \;\; \text{for all} \;\; y \;\; \text{and} \;\; h \leq h^*. \tag{5.8}$$

This implies $F(0) = 0$, which obviously holds for linear vector fields, but also for many interesting nonlinear vector fields associated with PDEs. If F is linear we denote it by L, and (5.8) can be expressed in terms of the logarithmic norm, [24, 61, 79], as $\mu[L] \leq 0$, where

$$\mu[L] = \lim_{h \to 0^+} \frac{\|I + hL\| - 1}{h}. \tag{5.9}$$

The condition $\mu[L] \leq 0$ implies that $\|y(t)\|$ is a non-increasing function of t, and thus $y = 0$ is a stable solution. For nonlinear vector fields satisfying $F(0) = 0$,

using the operator norm

$$\|F\| = \sup_{\|y\|\neq 0} \frac{\|F(y)\|}{\|y\|},$$

the corresponding logarithmic norm $\mu[F]$ can be readily defined as in (5.9); this is a straightforward special application of the general extension of the logarithmic norm to nonlinear maps, [79]. Thus (5.8) implies $\mu[F] \leq 0$, and once again $\|y(t)\|$ is a non-increasing function of t. Morover, if $\mu[F] < 0$, then $\|y + hF(y)\| \leq \|y\|$ in a nonempty interval, $0 < h < h^*$, since $\|y + hF(y)\|$ is a convex function of h.

The objective is to find methods that reproduce this behavior. A numerical method whose solution is non-increasing for all vector fields F satisfying (5.8) for step-sizes $0 < h \leq c\, h^*$ is said to be *strong stability preserving* (SSP). Obviously, the explicit Euler method is SSP with $c = 1$ and h^* is determined by Courant-Friedrichs-Lewy (CFL) condition.

Whether the vector field F satisfies $\mu[F] < 0$ depends on the choice of norm, but the choice is not restricted to specific norms, such as inner product norms. A standard objective is to construct semi-discretizations of hyperbolic conservation laws so that the total variation of discrete solutions does not increase in time [74], where the *total variation* in space is defined as

$$\|u\|_{TV} = \sum_{j=1}^{m} |u_{j+1} - u_j|, \quad u \in \mathcal{R}^m.$$

This is easily seen to be a semi-norm (referred to as the TV norm). If a semi-discretization satisfies $\mu[F] < 0$ with respect to the TV norm, and the time stepping method is SSP, the resulting method will overcome deficiencies that may occur in other methods, such as numerically induced oscillations in space, which are not present in the exact solutions of conservation laws. Thus the SSP method will produce a total variation diminishing (TVD) scheme for time steps $h \leq c\, h^*$.

To investigate linear multistep methods reducing total variation, further definitions are needed.

Definition 2 *[37] Let problem* (5.7) *satisfy condition* (5.8). *A k-step method given by formula* (6.32) *is an SSP method if there is a constant c such that the method applied to problem* (5.7) *with $0 < h \leq c\, h^*$ produces a sequence $\{y_i\}$ satisfying*

$$\|y_n\| \leq \max\{\|y_{n-1}\|, \|y_{n-2}\|, \ldots, \|y_{n-k}\|\}. \tag{5.10}$$

The maximal value of c is called the SSP constant of the method and is denoted by C.

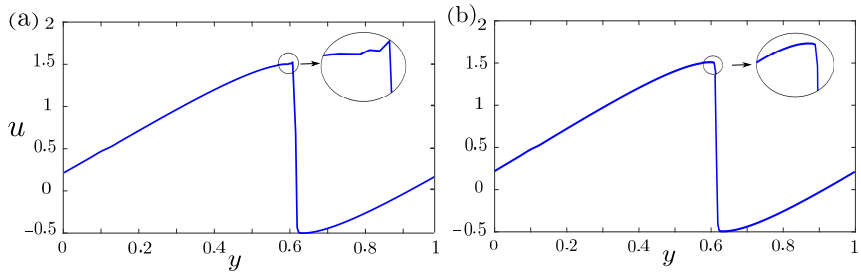

Figure 5.2: Solutions of Burgers' equation with fixed step-sizes, with initial function (6.75) and the optimal $(3, 2)$ SSP method. The solution was captured during the shock at $t = 0.23s$. (a) A step-size violating the step-size limit produces overshoot. (b) No oscillations are observed for a smaller step-size.

Remark 1 *[37] Consider an explicit method defined by formula* (2.3) *with* $\alpha_i \geq 0$, $\beta_i \geq 0$ *for all* i, *and let* $\gamma = \min_i \left\{ \frac{\alpha_i}{\beta_i} \;\middle|\; \beta_i \neq 0 \right\}$. *If* $\gamma > 0$, *the method is SSP with* $C = \gamma$.

We say a (k, p) SSP multistep method is optimal if it gives the largest SSP constant C. These methods are of particular interest [58, 54] so as to allow for largest step-sizes.

To illustrate the use of SSP methods, we consider the inviscid Burgers' equation with periodic boundary conditions

$$\begin{aligned} &u_t + uu_y = 0, \\ &u(y, 0) = g(y), \quad y \in [0, 1], \end{aligned} \tag{5.11}$$

using the smooth initial function

$$g(y) = \frac{1}{2} + \sin(2\pi y). \tag{5.12}$$

Furthermore, the model is semi-discretized with the fifth order Weighted Essentially Non Oscillatory (WENO) scheme [59, 53]. WENO is one of the spatial discretizations that are often combined with an SSP time integrator to preserve contractivity properties.

Figure 5.2(a) illustrates the solution of Burgers' equation with 256 grid points in space and the optimal explicit $(3, 2)$ SSP method, and with a fixed step-size that violates the step-size limit condition $h \leq Ch^*$ (see Definition 2 and Remark 1),

where the SSP constant of the method is $C = 0.5$. An undesired oscillation is visible during the shock formation. On the other hand, Figure 5.2(b) shows that for a fixed step-size satisfying the step-size limit condition, no spurious oscillations are observed.

5.2 Adaptive strong stability preserving multistep methods

An aim of Paper III is to develop a variable step-size formulation of SSP methods. Earlier, we discussed a procedure to formulate explicit (k, p) multistep methods where $p < k$. Further, it is known that for $k \geq 2$ there is no explicit (k, p) SSP multistep methods of order $p = k$ with all $\beta_i \geq 0$ [58]. Thus we can apply similar polynomial formulation of low order explicit multistep methods, Procedure1 to parametrize (k, p) SSP multistep methods. Because for SSP methods, we have $\alpha_i = 0 \Rightarrow \beta_i = 0$ (see Remark 1), we remove the second rule in Procedure 1 (If $\alpha_i = 0, \beta_i \neq 0$, include only $s'_{n-i} = 0$). We refer to Paper III for more details.

Consider the pairs of constant step-size coefficients (α_i, β_i) of an explicit SSP methods given by (2.3). The optimal SSP methods with $(k, p) \neq (6, 3)$ or $(16, 5)$, $k \leq 20$ satisfy following conditions:

1. $\alpha_1 \neq 0$, $\beta_1 \neq 0$, $\alpha_k \neq 0$.
2. If p is even, $\beta_k = 0$ and there are p pairs of non-zero coefficients.
3. If p is odd, $\beta_k \neq 0$ and there are $p - 1$ pairs of non-zero coefficients.

The rules to obtain optimal SSP methods are described in Procedure 2.

Note that the optimal $(6, 3)$ and $(16, 5)$ methods have p pairs of non-zero coefficients, although they have the odd orders.

Procedure 2: Formulation of optimal SSP methods

- *Take structural conditions at the point* $t = t_{n-1}$.
- *Take slack balance conditions at the intermediate points* $t = t_{n-j}$ *with* $\alpha_j \neq 0$, $1 < j < k$. *The method parameters are* $\tau_j = \beta_j/\alpha_j$.
- *Add the state slack* $s_{n-k} = 0$. *If* p *is odd, also add the derivative slack* $s'_{n-k} = 0$.
  (For the $(6, 3)$ *method, take a slack balance condition at* t_{n-6}.*)*

Example 10 *The optimal explicit* $(8, 5)$ *SSP method is constructed as follows,*

$$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-4} + h_{n-4}\tau_4 s'_{n-4} = 0, \\ s_{n-5} + h_{n-5}\tau_5 s'_{n-5} = 0, \\ s_{n-8} = 0, \\ s'_{n-8} = 0. \end{cases}$$

The coefficients of the optimal variable step-size (8,5) method are rational functions of degree 10 of four different step-size ratios, $\Omega_7, \Omega_5, \Omega_4, \Omega_3$, *and thus it is very difficult to find precise bounds that define positivity. The effect of increasing or decreasing the step-size at a constant rate was studied by considering*

$$h_{n-i} = (1 + \varepsilon)h_{n-i-1} \quad i = 1, \ldots, 7,$$

where ε *is a number close to 0. As for* $\varepsilon = 0$ *all coefficients are positive, and the SSP constant is 0.1451. By continuity, there is an interval where the step-sizes can vary while the method remains SSP. If* $\varepsilon > 0$ *the step-size is increased, and if* $\varepsilon < 0$ *it is decreased. All coefficients remain positive in the interval* $\varepsilon \in (-0.055, 0.035)$, *that is, a persistent increase of* 3.5% *or a decrease of* 5.5% *may be allowed. As for the variable SSP coefficient , some calculations turn out to be possible if these simplifications are made. For instance, when the step-size is repeatedly reduced at a constant rate, the SSP coefficient is given by a simple formula,* $\dfrac{0.1451}{\Omega_5 - \Omega_4}$. *For the constant step-size, the SSP constant is 0.1451; with a constant step-size increase of* 1%, *the SSP coefficient drops to 0.1064, and with a constant decrease of* 1% *it drops to 0.1395.*

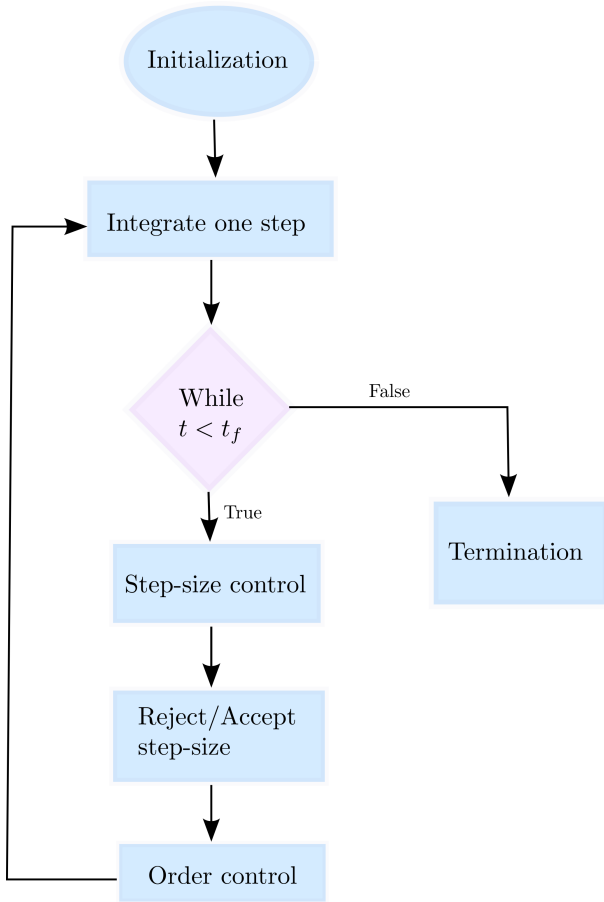

Figure 5.3: Structure of MODES.

These calculations suggest that as long as the step-sizes are varied slowly and smoothly, the variable step-size extension of an SSP method will remain SSP.

5.3 Implementation

We implemented our SSP formulation in the adaptive multistep solver MODES [14]. First, we briefly look at the structure of MODES. In Figure 5.3 the six main parts of MODES are illustrated.

At the initialization phase, the initial step-size and the k initial values for a $k-$step method are provided. To determine the initial step-size Algorithm 1 is used and to

calculate adequate number of initial values one of the initialization techniques that are explained in Chapter 3 are applied. Further, a predication of the solution at the first time step is obtained as a remedy to calculate the error estimate.

The coefficients of the polynomial in parametric formulation are computed in integration step. Then we evaluate the polynomial to find the solution value $y_n = P_n(t_n)$. This solution has to pass the step-size control phase in order to be saved as an accepted solution. For each class of methods with k steps and order p there is a function that computes the coefficients of the method polynomial of degree p by solving the derived parametric formulation for that class. For instance, a $2-$step explicit multistep method of order 2 is characterized by a second degree polynomial P_n,

$$P_n(t) = c_2(t - t_{n-1})^2 + c_1(t - t_{n-1}) + c_0,$$

satisfying the slack conditions in (6.38),

$$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-2}\cos\theta_1 + h_{n-2}s'_{n-2}\sin\theta_1 = 0 \end{cases} \tag{5.13}$$

where these three conditions, together with the parameter value θ_1, uniquely determine the polynomial coefficients. Then the solution at time t_n is obtained as $P_n(t_n) = c_2h_{n-1}^2 + c_1h_{n-1} + c_0$.

In the step-size control phase, the error estimation is computed by evaluating the polynomial method from previous step, P_{n-1}, to the new one, P_n at time t_n. If the error is less than a supplied tolerance, the step will be accepted otherwise it will be rejected in the next step of the flow chart 5.3. Several step-size controllers designed both for general and for more specific needs are implemented in this part.

The last step is the order control of multistep method. Here, the algorithm checks how large a step can be taken if the order is untouched or increased or decreased. Then a comparison among these three scenarios indicates if the changing order is favorable or not. In this thesis, we consider methods of fixed order and this step of flow chart is irrelevant for us.

Finally the integration is terminated when $t \geq tf$. We have to be sure that the integration hits the last time point thus a new solution is calculated at time point t_f.

The implementation in MODES was made by adding a function `polElow` which, given the method parameters λ and τ, computes the solution at t_n by solving system

for the coefficients of P_n, and then evaluates $y_n = P_n(t_n)$. We also have the option of calling some optimal (k, p) SSP methods for $p \leq 5$ by name, without giving their parameters explicitly.

The step-size controllers in MODES provide an estimate of the local error at each step. By monitoring the error estimation, the controllers increase or decrease the step-size when the error estimate is below or above the specified tolerance, and in particular they reduce the step-size when a numerical instability is detected. This is particularly important for explicit SSP integrators that cannot operate with step-sizes above their stability limit.

Using an adaptive implementation of these methods eliminates the need of calculating the SSP constant for each particular method, although care must be taken to set appropriate bounds for step-size changes. These bounds will be conservative and the method will be costlier than when the greedy step-size selection is used, but on the other hand, the error will be monitored, keeping it below a given tolerance. Our implementation retains MODES's step-size controllers and its mechanism for calculating the starting values of the multistep methods, which are provided by Runge-Kutta methods. As MODES allows the user to set upper and lower limits on the step-size ratios, it is a particularly suited platform for maintaining these ratios bounded, as required by SSP methods.

Chapter 6

Summary and main results

In the first part of this thesis we studied RK starters for restarting multistep methods. Two RK starter families equipped with error estimation are constructed and compared to the classical RK starter developed in 80's. The performance of the RK starters on restarting the multistep methods after a discontinuity shows their advantage both for problems with mild discontinuities, e.g., the pendulum and the bouncing ball, and for problems where the integration is interrupted to do a re-parametrization. The RK starters utilized the last order and step-size of the integration before the discontinuity. Some tests were carried out with an autostart algorithm to determine the initial step-size, because the last attempted step-size can be too small. The starting order and step-size are not studied in this thesis.

In the second part of the thesis, we parametrized $k-$step methods of order $p < k$ and suggested a polynomial formulation for SSP multistep methods that allows variable step-sizes. Although the suggested adaptive formulation of SSP multistep methods has a simple structure, it is not easy to compute the restriction on the step-sizes to guarantee positivity of the method coefficients for methods with order higher than 3. Further, the step-size controllers in MODES take smooth (small) step-size changes that enables us to retain the positivity of coefficients by restricting the allowable changes in step-size ratios.

Finally, we derived a parametric formulation for $\beta-$blocked multistep methods that allows us to construct the first adaptive $\beta-$blocked multistep methods solver for the solution of index-2 Euler-Lagrange DAE system. These methods were implemented as a plug-in to MODES and extended it from an adaptive ODE solver

to adaptive ODE/DAE solver. As a proof of concept, some numerical experiments were carried out. In our experiments, the $\beta-$blocked multistep methods are initialized by a one step $\beta-$blocked method with a small initial step-size given by the user. Choosing a small enough initial step-size and a starting scheme for initializing the numerical methods applied on DAEs models remain to be studied.

Bibliography

[1] J. Åkesson, M. Gäfvert, and H. Tummescheit. JModelica—an open source platform for optimization of modelica models. In *Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, Vienna, Austria, 2009. TU Wien.

[2] C. Andersson. *Assimulo: A new Python based class for simulation of complex hybrid DAEs and its integration in JModelica.org*. Lund University, 2011.

[3] C. Andersson. *A software framework for implementation and evaluation of co-simulation algorithms*. PhD thesis, Lund university, 2013.

[4] C. Andersson, J. Åkesson, C. Führer, and M. Gäfvert. Import and export of functional mock-up units in JModelica.org. In *8th International Modelica Conference 2011*, Dresden, Germany, mar 2011.

[5] C. Andersson, C. Führer, and J. Åkesson. Assimulo: A unified framework for ODE solvers. *Mathematics and Computers in Simulation*, 116:26 – 43, 2015.

[6] C. Arévalo. *Matching the structure of DAEs and multistep methods.* Ph.D. thesis, Lund University, 1993.

[7] C. Arévalo, C. Führer, and M. Selva. A collocation formulation of multistep methods for variable step-size extensions. *Applied Numerical Mathematics*, 42(1-3):5–16, 2002.

[8] C. Arévalo, C. Führer, and G. Söderlind. Stabilized multistep methods for index 2 Euler-Lagrange DAEs. *BIT Numerical Mathematics*, 36(1):1–13, 1996.

[9] C. Arévalo, C. Führer, and G. Söderlind. β-blocked multistep methods for Euler-Lagrange DAEs: Linear analysis. *ZAMM-Journal of Applied Mathematics and Mechanics/ Zeitschrift für Angewandte Mathematik und Mechanik*, 77(8):609–617, 1997.

[10] C. Arévalo, C. Führer, and G. Söderlind. Regular and singular β-blocking of difference corrected multistep methods for nonstiff index-2 DAEs. *Applied numerical mathematics*, 35(4):293–305, 2000.

[11] C. Arevalo, E. Jonsson-Glans, J. Olander, M. Selva-Soto, and G. Söderlind. MODES. `http://www.maths.lu.se/staff/carmen-arevalo/downloads/`, 2016.

[12] C. Arévalo and P. Lötstedt. Improving the accuracy of BDF methods for index 3 differential-algebraic equations. *BIT Numerical Mathematics*, 35(3):297–308, 1995.

[13] C. Arévalo and G. Söderlind. Grid-independent construction of multistep methods. *Journal of Computational Mathematics*, 35(5):670–690, 2017.

[14] C. Arévalo, G. Söderlind, E. Jonsson-Glans, J. Olander, and M. Selva-Soto. MODES: A software platform for adaptive high order multistep methods. *Lund University, Preprints in Mathematical Sciences*, 2017:1.

[15] C. Arévalo, G. Söderlind, and J. López Diaz. Constant coefficient linear multistep methods with step density control. *Journal of computational and applied mathematics*, 205(2):891–900, 2007.

[16] M. Arnold. The stabilization of linear multistep methods for constrained mechanical systems. *Applied numerical mathematics*, 28(2-4):143–159, 1998.

[17] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer methods in applied mechanics and engineering*, 1(1):1–16, 1972.

[18] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, et al. The functional mockup interface for tool independent exchange of simulation models. In *Modelica 2011 Conference, March*, pages 20–22, 2011.

[19] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14. SIAM, 1996.

[20] B. Burgermeister, M. Arnold, and B. Esterl. DAE time integration for real-time applications in multi-body dynamics. *ZAMM-Journal of Applied Mathematics and Mechanics/ Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics*, 86(10):759–771, 2006.

[21] J. C. Butcher. On fifth order Runge-Kutta methods. *BIT Numerical Mathematics*, 35(2):202–209, 1995.

[22] S. L. Campbell. Consistent initial conditions for linear time varying singular systems. *Frequency Domain and State Space Methods for Linear Systems, Elsevier Science (North-Holland), Amsterdam*, 1986.

[23] S. L. Campbell. A computational method for general higher index nonlinear singular systems of differential equations. *IMACS Transactions on Scientific Computing*, 1989.

[24] G. Dahlquist. *Stability and error bounds in the numerical integration of ordinary differential equations*. PhD thesis, Almqvist & Wiksell, 1958.

[25] P. Deuflhard and F. Bornemann. *Scientific computing with ordinary differential equations*, volume 42. Springer Science and Business Media, 2012.

[26] L. Dieci and L. Lopez. A survey of numerical methods for IVPs of ODEs with discontinuous right-hand side. *Journal of Computational and Applied Mathematics*, 236(16):3967–3991, 2012.

[27] M. Dowell and P. Jarratt. A modified regula falsi method for computing the root of an equation. *BIT Numerical Mathematics*, 11(2):168–174, 1971.

[28] E. Eich. Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints. *SIAM Journal on Numerical Analysis*, 30(5):1467–1482, 1993.

[29] E. Eich-Soellner and C. Führer. *Numerical methods in multibody dynamics*, volume 45. Teubner Stuttgart, 1998.

[30] L. Euler. Institutionum calculi integralis. *Petropoli: Impensis Academiae Imperialis Scientiarum*, 1768.

[31] C. Führer and B. J. Leimkuhler. Numerical solution of differential-algebraic equations for constrained mechanical motion. *Numerische Mathematik*, 59(1):55–69, 1991.

[32] C. W. Gear. Simultaneous numerical solution of differential-algebraic equations. *IEEE Transactions on Circuit Theory*, 18(1):89–95, 1971.

[33] C. W. Gear. Runge–Kutta starters for multistep methods. *ACM Transactions on Mathematical Software (TOMS)*, 6(3):263–279, 1980.

[34] C. W. Gear, B. Leimkuhler, and G. K. Gupta. Automatic integration of Euler-Lagrange equations with constraints. *Journal of Computational and Applied Mathematics*, 12:77–90, 1985.

[35] C William Gear. Algorithm 407: DIFSUB for solution of ordinary differential equations. *Communications of the ACM*, 14(3):185–190, 1971.

[36] J. B. Goodman and R. J. LeVeque. On the accuracy of stable schemes for 2d scalar conservation laws. *Mathematics of computation*, pages 15–21, 1985.

[37] S. Gottlieb, D. I. Ketcheson, and C. Shu. *Strong stability preserving Runge-Kutta and multistep time discretizations*. World Scientific, 2011.

[38] S. Gottlieb and C. Shu. Total variation diminishing Runge–Kutta schemes. *Mathematics of computation of the American Mathematical Society*, 67(221):73–85, 1998.

[39] S. Gottlieb, C. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43(1):89–112, 2001.

[40] Y. Hadjimichael, D. I. Ketcheson, L. Lóczi, and A. Németh. Strong stability preserving explicit linear multistep methods with variable step size. *SIAM Journal on Numerical Analysis*, 54(5):2799–2832, 2016.

[41] E. Hairer, C. Lubich, and M. Roche. *The numerical solution of differential-algebraic systems by Runge–Kutta methods*, volume 1409. Springer, 2006.

[42] E. Hairer, S. P. Norsett, and G. Wanner. *Solving ordinary differential equations I.* 1993.

[43] E. Hairer and G. Wanner. Radau5- An implicit Runge–Kutta code. *Report, Université de Geneve, Dept. de mathématiques, Geneve*, 1988.

[44] E. Hairer and G. Wanner. Solving ordinary differential equations II: Stiff and differential-algebraic problems second revised edition, 1996.

[45] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, III. In *Upwind and high-resolution schemes*, pages 218–290. Springer, 1987.

[46] A. Harten, S. Osher, B. Engquist, and S. R. Chakravarthy. Some results on uniformly high-order accurate essentially nonoscillatory schemes. *Applied Numerical Mathematics*, 2(3-5):347–377, 1986.

[47] I. Higueras. On strong stability preserving time discretization methods. *Journal of Scientific Computing*, 21(2):193–223, 2004.

[48] A. C. Hindmarsh. ODEPACK, a systematized collection of ODE solvers. In R. S. Stepleman et al., editor, *IMACS Transactions on Scientific Computation*, volume 1, pages 55–64. 1983.

[49] 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. (3):363–396, 2005.

[50] W. H. Hundsdorfer, S. J. Ruuth, and R. J. Spiteri. *Monotonicity-preserving linear multistep methods*. Centrum voor Wiskunde en Informatica, 2002.

[51] S. Ilie, G. Söderlind, and R. M. Corless. Adaptivity and computational complexity in the numerical solution of ODEs. *Journal of Complexity*, 24(3):341–361, 2008.

[52] A. Iserles. *A first course in the numerical analysis of differential equations*. Number 44. Cambridge university press, 2009.

[53] G. Jiang and C. Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996.

[54] D. I. Ketcheson. Computation of optimal monotonicity preserving general linear methods. *Mathematics of Computation*, 78(267):1497–1513, 2009.

[55] A. Kværnø. Runge–Kutta methods applied to fully implicit differential-algebraic equations of index 1. *Mathematics of Computation*, 54(190):583–625, 1990.

[56] R. Lamour, R. März, and R. M. Mattheij. On the stability behaviour of systems obtained by index-reduction. *Journal of Computational and Applied Mathematics*, 56(3):305–319, 1994.

[57] R. I. Leinea, C. Glockerb, and D. H. van Campena. Nonlinear dynamics of the woodpecker toy. In *ASME 2001 Design Engineering Technical Conference and Computers and Information in Engineering Conference*, Pittsburg, PA, September 9-12 2001.

[58] H. W. Lenferink. Contractivity preserving explicit linear multistep methods. *Numerische Mathematik*, 55(2):213–223, 1989.

[59] X. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.

[60] P. Lötstedt and L. Petzold. Numerical solution of nonlinear differential equations with algebraic constraints. I. convergence results for backward differentiation formulas. *Mathematics of computation*, 46(174):491–516, 1986.

[61] S. M. Lozinskii. Error estimate for numerical integration of ordinary differential equations. I. *Izvestiya Vysshikh Uchebnykh Zavedenii. Matematika*, (5):52–90, 1958.

[62] E. S. Maciel and A. P. Pimenta. Comparison among MUSCL, ENO, and WENO procedures as applied to reentry flows in 2D. *Computational and Applied Mathematics Journal*, 1(5):355–377, 2015.

[63] R. März. *Multistep methods for initial value problems in implicit differential-algebraic equations*. Humboldt-Universität zu Berlin. Sektion Mathematik, 1981.

[64] F. Mohammadi, C. Arévalo, and C. Führer. Runge–Kutta restarters for multistep methods in presence of frequent discontinuities. *Journal of Computational and Applied Mathematics*, 316:287–297, 2017.

[65] A. Nordsieck. On numerical integration of ordinary differential equations. *Mathematics of Computation*, 16(77):22–49, 1962.

[66] L. R. Petzold. A description of DASSL: A differential/algebraic system solve. *Scientific Computing, eds. R.S. Stepleman et al.*, pages 65–68, 1983.

[67] L. R. Petzold. Order results for implicit Runge–Kutta methods applied to differential/algebraic systems. *SIAM Journal on Numerical Analysis*, 23(4):837–852, 1986.

[68] L. R. Petzold and A. C. Hindmarsh. LSODAR. *Computing and Mathematics Research Division, I-316 Lawrence Livermore National Laboratory, Livermore, CA*, 94550, 1997.

[69] L. R. Petzold and P. Lötstedt. Numerical solution of nonlinear differential equations with algebraic constraints II: Practical implications. *SIAM Journal on Scientific and Statistical Computing*, 7(3):720–733, 1986.

[70] A. Ralston. Runge-Kutta methods with minimum error bounds. *Mathematics of computation*, 16(80):431–437, 1962.

[71] S. J. Ruuth and W. Hundsdorfer. High-order linear multistep methods with general monotonicity and boundedness properties. *Journal of Computational Physics*, 209(1):226–248, 2005.

[72] L. F. Shampine. Conservation laws and the numerical solution of odes. *Computers and Mathematics with Applications*, 12(5-6):1287–1296, 1986.

[73] C. Shu. Total-variation-diminishing time discretizations. *SIAM Journal on Scientific and Statistical Computing*, 9(6):1073–1084, 1988.

[74] C. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77:439–471, 1988.

[75] A. Sjö. *Analysis of computational algorithms for linear multistep methods*. PhD thesis, Lund University, 1999.

[76] S. J. Small. *Runge–Kutta type methods for differential-algebraic equations in mechanics*. The University of Iowa, 2011.

[77] G. Söderlind. DASP3- A program for the numerical integration of partitioned stiff ODE: s and differential-algebraic systems. Technical report, KTH Royal Institute of Technology, 1980.

[78] G. Söderlind. A multi-purpose system for the numerical integration of ODE's. *Applied Mathematics and Computation*, 31:346–360, 1989.

[79] G. Söderlind. The logarithmic norm. History and modern theory. *BIT Numerical Mathematics*, 46(3):631–652, 2006.

[80] H. J. Stetter. Asymptotic expansions for the error of discretization algorithms for non-linear functional equations. *Numerische Mathematik*, 7(1):18–31, 1965.

[81] B. Van Leer. Towards the ultimate conservative difference scheme. *Journal of Computational Physics*, 135(2):229–248, 1997.

[82] R. von Schwerin and H. G. Bock. A Runge–Kutta starter for a multistep method for differential-algebraic systems with discontinuous effects. *Applied Numerical Mathematics*, 18(1):337–350, 1995.

Paper I

10th international Modelica conference, Lund, 2014

Restarting algorithms for simulation problems with discontinuities

Fatemeh Mohammadi Carmen Arévalo Claus Führer

Numerical Analysis, Center for Mathematical Sciences, Lund University,
Sölvegatan 18, SE-22100 Lund, Sweden

Abstract

Modelica's language support includes so-called events for describing discontinuities. Modern integrating environments, like Assimulo, provide elaborate event detection and event handling methods. In addition, the overall performance of a simulation of models with discontinuities (hybrid models) depends strongly on the methods for restarting the integration after an event detection. The present paper reviews two restarting methods for multistep methods, both based on Runge–Kutta starters, and presents preliminary first experiments with Assimulo and LSODAR as a proof of concept, which motivates to apply the technique to hybrid systems described in Modelica and simulated by JModelica.org/PyFMI and Assimulo [1, 3, 2].

Keywords: events, discontinuities, hybrid systems, multistep method, Runge–Kutta method, simulation restart

1 Introduction

When dealing with hybrid systems, i.e. dynamic systems with state or time discontinuities, much emphasis has been put on the modeling aspect. Attempts to standardize the formulation of events and algorithms for event detection were in the focus of development and research, e.g. [4]. On the other hand, the question of restarting complex integration methods like multistep methods, with their sophisticated internal error and order control algorithms and internal data representations, did not attract much attention. In this paper we want to take up and review two early ideas for restarting and to present some experiments using the JModerlica.org - PyFMI - Assimulo toolchain.

A multistep method is classically started by step-wise increasing the order of the method, starting with a first order method (implicit Euler method) and leading to a method having the operational order of the problem at hand. Simulations are often done for a set of parameterized models for which the operational order and also good guesses of initial step sizes are available from other simulation runs. Thus, a goal for improving the integration performance is to avoid costly starting phases and directly start the integrator with a method already having the operational order. To start such a higher order method several internal values are required. Here we consider two ideas for providing these values. In both cases the starting values are obtained from the stage values of a single Runge–Kutta step of a specially designed method. One of them uses state values, while the other is geared to Nordsieck based multistep methods like LSODAR.

2 Runge–Kutta starter with state values

We demonstrate the principle by constructing a Runge–Kutta starter for a third order multistep method, [8].

Furthermore, we construct two error estimates for determining the starting step of both the Runge–Kutta starter and a class of multistep methods, i.e. Adams methods.

Such a Runge–Kutta starter has to have an internal stage of order 3 as soon as possible and all subsequent stages need to be at least of order 3. In addition; the final result should be of order 4 for the purpose of error estimation. It is well-known

that to get third-order accuracy at least three internal stages are necessary, and to conserve this accuracy for subsequent stages we need to aim for a Runge–Kutta method with at least six stages, [5]. We will thus consider a 6-stage Runge–Kutta method.

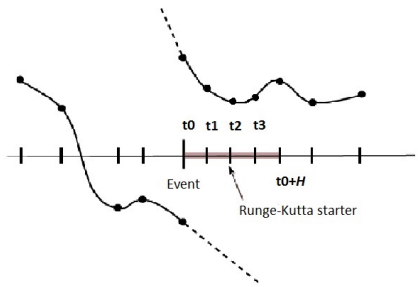

Figure 6.1: Runge--Kutta starter after a discontinuity

For the initial value problem

$$y' = f(t, y), \qquad y(0) = y_0, \tag{6.1}$$

an s-stage explicit Runge–Kutta method can be written in the form,

$$\begin{aligned} k_i :=& f(t_0 + c_i H, g_{i-1}), \\ g_i :=& y_0 + H \sum_{j=1}^{i} a_{ij} k_j, \qquad i = 1, \ldots, s, \\ y_1 :=& y_0 + H \sum_{j=1}^{s} b_j k_j, \end{aligned} \tag{6.2}$$

where y_1 is the numerical solution at $t_1 = t_0 + H$, H is the Runge–Kutta step size and k_i are stage derivatives.

f may be discontinuous but it is assumed to be piecewise smooth.

In the construction of an order 4, 6-stage explicit Runge–Kutta method, order

conditions up to order four need to be satisfied. Let

$$\begin{aligned} b &:= (b_1, b_2, \cdots, b_s)^T, \\ a_i &:= (a_{i1}, a_{i2}, \cdots, a_{ii}), \\ C_i &:= \mathrm{diag}(c_1, \cdots, c_i), \\ A_i &:= (a_{jk})_{j,k=1}^{i}, \\ e_i &:= (1, 1, \cdots, 1)^T. \end{aligned} \tag{6.3}$$

When deriving the stage order conditions, we make use of the fact that in Eq. (6.2), the stage values g_i and y_i have structurally the same form. Therefore, we can derive the order conditions for internal stages in the same way.
The order conditions for a fourth-order Runge–Kutta method are

- order 1
$$b^T e_s = 1.$$
- order 2
$$b^T C_s e_s = \frac{1}{2}.$$
- order 3
$$\begin{aligned} b^T C_s^2 e_s &= \frac{1}{3}, \\ b^T A_s C_s e_s &= \frac{1}{6}. \end{aligned}$$
- order 4
$$\begin{aligned} b^T C_s^3 e_s &= \frac{1}{4}, \\ b^T C_s A_s C_s e_s &= \frac{1}{8}, \\ b^T A_s C_s^2 e_s &= \frac{1}{12}, \\ b^T A_s^2 C_s e_s &= \frac{1}{24}. \end{aligned} \tag{6.4}$$

Additionally we require

$$\sum_{j=1}^{s} a_{ij} = c_i. \tag{6.5}$$

The remaining order conditions for internal stages $i = 4, 5, 6$ are

- order 2

$$a_i^T C_i e_i = \frac{1}{2} c_i^2.$$

- order 3

$$\begin{aligned} a_i^T C_i^2 e_i &= \frac{1}{3} c_i^3, \\ a_i^T A_i C_i e_i &= \frac{1}{6} c_i^3. \end{aligned} \tag{6.6}$$

We want to both obtain third-order accuracy and minimize the truncation error bound. Raltson [6] showed that the third-order Runge-Kutta method which has the minimal error bound among all third-order Runge–Kutta methods is

$$\begin{aligned} k_1 &= hf(t_n, y_n), \\ k_2 &= hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1), \\ k_3 &= hf(t_n + \frac{3}{4}h, y_n + \frac{3}{4}k_2), \\ y_{n+1} &= y_n + \frac{2}{9}k_1 + \frac{1}{3}k_2 + \frac{4}{9}k_3. \end{aligned} \tag{6.7}$$

This implies the Butcher tableau for the first four stages is

0	0		
$\frac{1}{2}c_4$	$\frac{1}{2}c_4$	0	
$\frac{3}{4}c_4$	0	$\frac{3}{4}c_4$	0
c_4	$\frac{2}{9}c_4$	$\frac{1}{3}c_4$	$\frac{4}{9}c_4$

From condition (6.5) we find

$$a_{21} = c_2, \qquad a_{31} = (1-\theta)c_3, \;\; a_{32} = \theta c_3. \tag{6.8}$$

We now substitute (6.6) and (6.5) in (6.4) to calculate the values b_i for the six stage Runge–Kutta method.

$$\begin{aligned} b_2c_2^3 + b_3c_3^3 + b_4c_4^3 + b_5c_5^3 + b_6c_6^3 &= \frac{1}{4}, \\ b_3c_3\theta c_3c_2 + \frac{1}{2}b_4c_4^3 + \frac{1}{2}b_5c_5^3 + \frac{1}{2}b_6c_6^3 &= \frac{1}{8}, \\ b_3c_3\theta c_2^2 + \frac{1}{3}b_4c_4^3 + \frac{1}{3}b_5c_5^3 + \frac{1}{3}b_6c_6^3 &= \frac{1}{12}, \\ \frac{1}{6}b_4c_4^3 + \frac{1}{6}b_5c_5^3 + \frac{1}{6}b_6c_6^3 &= \frac{1}{24}. \end{aligned} \tag{6.9}$$

For the first three stages we require $c_2 \neq 0, c_3 \neq 0$ and $\theta \neq 0$, otherwise a third-order Runge–Kutta method cannot be obtained. So $b_2 = b_3 = 0$ and Equations (6.9) reduce to a single equation

$$b_4c_4^3 + b_5c_5^3 + b_6c_6^3 = \frac{1}{4}.$$

We repeat this process for order 2 and 3 conditions, getting

$$b_4c_4^2 + b_5c_5^2 + b_6c_6^2 = \frac{1}{3},$$

and

$$b_4c_4 + b_5c_5 + b_6c_6 = \frac{1}{2}.$$

respectively.

Here we have a system of equations for given c_4, c_5, c_6,

$$\begin{pmatrix} c_4 & c_5 & c_6 \\ c_4^2 & c_5^2 & c_6^2 \\ c_4^3 & c_5^3 & c_6^3 \end{pmatrix} \begin{pmatrix} b_4 \\ b_5 \\ b_6 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{4} \end{pmatrix}.$$

We obtain a Vandermonde type matrix, which has, for distinct c_4, c_5 and c_6, a

unique solution:

$$
\begin{aligned}
b_1 &= 1 - b_4 - b_5 - b_6, \\
b_4 &= \frac{3 - 4c_5 - 4c_6 + 6c_5c_6}{12c_4(c_4 - c_5)(c_4 - c_6)}, \\
b_5 &= \frac{3 - 4c_4 - 4c_6 + 6c_4c_6}{12c_5(c_4 - c_5)(c_5 - c_6)}, \\
b_6 &= \frac{3 - 4c_4 - 4c_5 + 6c_4c_5}{12c_6(c_4 - c_6)(c_5 - c_6)}.
\end{aligned}
$$

In order to obtain an equidistant grid for starting multistep methods, we can choose

$$
c_4 = \frac{1}{4}, \qquad c_5 = \frac{1}{2}, \qquad c_6 = \frac{3}{4},
$$

which gives

$$
b_1 = b_2 = b_3 = 0, \quad b_4 = \frac{2}{3}, \quad b_5 = -\frac{1}{3}, \quad b_6 = \frac{2}{3}.
$$

Finally, by solving the equations that guarantee the remaining order conditions, we obtain the Butcher tableau for the Runge–Kutta starter:

$$
\begin{array}{c|cccccc}
0 & 0 & & & & & \\
\frac{1}{8} & \frac{1}{8} & 0 & & & & \\
\frac{3}{16} & 0 & \frac{3}{16} & 0 & & & \\
\frac{1}{4} & \frac{1}{18} & \frac{1}{12} & \frac{1}{9} & 0 & & \\
\frac{1}{2} & \frac{5}{12} & -\frac{1}{3} & -\frac{4}{9} & 1 & 0 & \\
\frac{3}{4} & -\frac{1}{4} & \frac{3}{4} & 1 & -\frac{3}{2} & \frac{3}{4} & 0 \\
\hline
 & 0 & 0 & 0 & \frac{2}{3} & -\frac{1}{3} & \frac{2}{3}
\end{array}
$$

We can apply the explicit Runge–Kutta starter to start $k = 3$-step Adams methods. We need k data points $(t_i, f_i), i = n - k + 1, \ldots, n$ to compute the respective polynomials for either the Adams–Moulton corrector or the Adams–Bashforth predictor.

3 Error estimation and step size control

The error of the numerical solution depends on the function f and on the step size H. The step size influences the size of the global error increment. Thus, for a given tolerance the step size is chosen in such a way that the global error increment meets a user-supplied tolerance bound.

We use an embedded formula to obtain an error estimate for the Runge–Kutta starter of the Adams method. The estimation can be done by reusing the available stages to produce a formula of different order. To do so, we apply stages k_4, k_5, k_6 of the Runge–Kutta method in Section 2 and obtain $\hat{y}_1$ by the third-order Adams–Bashforth method. We generate the difference table

$$\begin{array}{lll} c_4 = h & k_4 & \\ & & \nabla k_5 \\ c_5 = 2h & k_5 & \qquad \nabla^2 k_6 \\ & & \nabla k_6 \\ c_6 = 3h & k_6 & \end{array}$$

where $h = \frac{H}{4}$ and H is the Runge–Kutta step size. The third-order Adams–Bashforth method is

$$\hat{y}_1 = g_6 + h \sum_{i=1}^{3} \gamma_{i-1} \nabla^{i-1} k_6 = g_6 + h \sum_{i=1}^{3} \gamma_i^\star k_{i+3}, \tag{6.10}$$

The latter is the Lagrange form of the Adams–Bashforth method and

$$\begin{aligned} \gamma_1^\star &= \gamma_2 = \frac{5}{12}, \\ \gamma_2^\star &= -\gamma_1 - 2\gamma_2 = -\frac{4}{3}, \\ \gamma_3^\star &= \gamma_0 + \gamma_1 + \gamma_2 = \frac{23}{12}. \end{aligned}$$

As we have

$$g_6 = y_0 + H \sum_{j=1}^{5} a_{6j} k_j, \tag{6.11}$$

we can rewrite equation (6.10) as

$$\hat{y}_1 = y_0 + H \sum_{j=1}^{6} \hat{b}_j k_j$$

Thus, the error estimate is

$$y_1 - \hat{y}_1 = y_0 + H \sum_{j=1}^{6} b_j k_j - \left(y_0 + H \sum_{j=1}^{6} \hat{b}_j k_j \right) = h \sum_{j=1}^{6} \hat{e}_j k_j, \quad (6.12)$$

giving the following coefficients

j	1	2	3	4	5	6
$\hat{b_j}$	$-\frac{1}{4}$	$\frac{3}{4}$	1	$-\frac{67}{48}$	$\frac{5}{12}$	$\frac{23}{48}$
$\hat{e_j}$	$\frac{1}{4}$	$-\frac{3}{4}$	-1	$\frac{99}{48}$	$-\frac{3}{4}$	$\frac{3}{16}$

This error estimation is the difference of a third-order predictor and the fourth-order result of the Runge–Kutta method that is applied to determine the step size for the Runge–Kutta starter.

We will now develop a second error estimate, to determine a step size for Adams method. We evaluate the right-hand side function f at the solution value y_1 and call it k_7. Then we generate the third-order Adams–Moulton corrector using k_5, k_6, k_7,

$$\begin{array}{lll} c_5 = h & k_5 & \\ & & \nabla k_6 \\ c_6 = 2h & k_6 & \nabla^2 k_7 \\ & & \nabla k_7 \\ c_7 = 3h & k_7 & \end{array}$$

The third-order approximation by the Adams–Moulton method is

$$\tilde{y_1} = g_6 + h\sum_{i=1}^{3} \beta_{i-1}\nabla^{i-1}k_7 = g_6 + h\sum_{i=1}^{3} \beta_i^\star k_{i+4}, \tag{6.13}$$

where the latter is the Lagrange form of the Adams–Moulton corrector and

$$\begin{aligned}
\beta_1^\star &= \beta_2 = -\frac{1}{12},\\
\beta_2^\star &= -\beta_1 - 2\beta_2 = \frac{2}{3},\\
\beta_3^\star &= \beta_0 + \beta_1 + \beta_2 = \frac{5}{12}.
\end{aligned}$$

From Equation (6.11) we can rewrite the third-order corrector in Runge–Kutta form

$$\tilde{y_1} = y_0 + H\sum_{j=1}^{7} \tilde{b}_j k_j.$$

resulting in the following table:

j	1	2	3	4	5	6	7
$\tilde{b}_j$	$-\frac{1}{4}$	$\frac{3}{4}$	1	$-\frac{3}{2}$	$\frac{35}{48}$	$\frac{1}{6}$	$\frac{5}{48}$
$\tilde{e}_j$	$\frac{1}{4}$	$-\frac{3}{4}$	-1	$\frac{13}{6}$	$-\frac{51}{48}$	$\frac{1}{2}$	$-\frac{5}{48}$

The error estimate is used in determining the step size for starting the third-order Adams-Moulton method.

4 Runge–Kutta starter as an extrapolation method

The starting values of a multistep method can also be stored as a differentiation array, which constitutes the Nordsieck vector of scaled derivatives $\frac{h^i y^{(i)}}{i!},\ i = 0, \ldots, p$. It is possible to convert a vector of state values at consecutive grid points into a

Nordsieck array and vice versa without loss of accuracy. Classical multistep codes like LSODAR are based on Nordsieck formulations.

Based on such a Nordsieck formulation an alternative way of constructing a Runge–Kutta starter was developed by Gear, [5]. Here, the asymptotic expansion of the global error of a base method is used to construct a Runge-Kutta method with higher order stage values.

We use the explicit Euler method as a base method to compute y_i^m(the super-script m refers to the corresponding step size, $h_m = \frac{H}{m}$) for $i = 1, \ldots, m$, $m = p, p-1, \ldots, 1$. From these values the terms in the asymptotic expansion, [7],

$$y_i^m \quad = \quad y(ih_m) \quad + \quad \sum_{q=1}^{p} e_q(ih_m)h^q \quad + \quad \mathcal{O}(H^{p+1}). \tag{6.14}$$

can successively be eliminated by an extrapolation technique until a method of a required order is obtained. The resulting method is known to be a Runge–Kutta method.

We exemplify the approach by aiming for third-order accurate Nordsieck values and restricting ourselves to autonomous differential equations for simplicity. The same process can be employed to obtain higher order accuracy.

Let $h = \frac{H}{m}$, and integrate the autonomous form of the differential equation (6.1) on the interval $[y_0, y_0 + H]$ with Euler's method, using m steps of size $\frac{H}{m}$ for $m = 3, 2, 1$.

For $m = 3$

$$\begin{array}{ll} y_1^3 = y_0 + hf(y_0) = y_0 + k_1, & k_1 = hf(y_0), \\ y_2^3 = y_1^3 + hf(y_1^3) = y_0 + k_1 + k_2, & k_2 = hf(y_1^3), \\ y_3^3 = y_2^3 + hf(y_2^3) = y_0 + k_1 + k_2 + k_3, & k_3 = hf(y_2^3). \end{array} \tag{6.15}$$

For $m = 2$

$$\begin{aligned} y_1^2 &= y_0 + \frac{3}{2}hf(y_0) = y_0 + \frac{3}{2}k_1, \\ y_2^2 &= y_1^3 + \frac{3}{2}hf(y_1^3) = y_0 + \frac{3}{2}k_1 + \frac{3}{2}k_4, \quad k_4 = hf(y_1^2). \end{aligned} \tag{6.16}$$

For $m = 1$

$$y_1^1 = y_0 + 3hf(y_0) = y_0 + 3k_1. \tag{6.17}$$

with $h = \frac{H}{3}$. We use approxmation formulas for higher derivatives

$$h_m^k y^{(k)}(\frac{H}{2}) = \sum_{i=0}^{m} d_{ik} y(ih_m) + \sum_{s=k+1}^{p} c_{sk} h_m^s y^s(\frac{H}{2}) + \mathcal{O}(H^{p+1}), \quad m \geq k$$

and (6.14) to obtain, after some algebraic manipulations,

$$\begin{aligned}
D_3^3 &= y_3^3 - 3y_2^3 + 3y_1^3 - y_0 = h^3 y^{(3)}, \\
D_2^3 &= y_3^3 - y_2^3 - y_1^3 + y_0 = 2h^2 y^{(2)} + 2h^3 e_1^{(2)}, \\
D_2^2 &= y_2^2 - 2y_1^2 + y_0 = (\frac{3h}{2})^2 y^{(2)} + (\frac{3h}{2})^3 e_1^{(2)}, \\
D_1^3 &= y_2^3 - y_1^3 = hy^{(1)} + h^2 e_1^{(1)} + h^3 e_2^{(1)} + \frac{h^3}{24} y^{(3)}, \\
D_1^2 &= y_2^2 - y_0 = 3hy^{(1)} + \frac{9}{2} h^2 e_1^{(1)} + \frac{27}{4} h^3 e_2^{(1)} + \frac{9}{8} h^3 y^{(3)}, \\
D_1^1 &= y_1^1 - y_0 = 3hy^{(1)} + 9h^2 e_1^{(1)} + 27h^3 e_2^{(1)} + \frac{27}{24} h^3 y^{(3)}.
\end{aligned} \tag{6.18}$$

All derivatives are evaluated at $\frac{H}{2}$ and $\mathcal{O}(h^4)$ terms are dropped. Estimates of the derivatives can be derived at any point within a constant multiple of the interval H with the same accuracy. We solve the system (6.18) to remove the error terms for $h^k y^{(k)}(\frac{H}{2})$, for $k = 1, \ldots, m$, to get

$$\begin{aligned}
h^3 y^{(3)}(\frac{H}{2}) &= D_3^3 + \mathcal{O}(h^4), \\
h^2 y^{(2)}(\frac{H}{2}) &= \frac{3}{2} D_2^3 - \frac{8}{9} D_2^2 + \mathcal{O}(h^4), \\
hy^{(1)}(\frac{H}{2}) &= \frac{9}{2} D_1^3 - \frac{4}{3} D_1^2 + \frac{1}{6} D_1^1 + \frac{9}{8} D_3^3 + \mathcal{O}(h^4).
\end{aligned} \tag{6.19}$$

The D_k^m can be expressed as combinations of stage values k_i. All $\mathcal{O}(h^4)$ terms are

neglected.

$$\begin{aligned} D_3^3 &= k_1 - 2k_2 + k_3, \\ D_2^3 &= -k_1 + k_3, \\ D_2^2 &= -\frac{3}{2}k_1 + \frac{3}{2}k_4, \\ D_1^3 &= k_2, \\ D_1^2 &= \frac{3}{2}k_1 + \frac{3}{2}k_4, \\ D_1^1 &= 3k_1. \end{aligned} \tag{6.20}$$

From (6.19) and (6.20),

$$\begin{aligned} h^3 y^{(3)}(\frac{H}{2}) &= k_1 - 2k_2 + k_3 + \mathcal{O}(h^4), \\ h^2 y^{(2)}(\frac{H}{2}) &= -\frac{1}{6}k_1 + \frac{3}{2}k_3 - \frac{4}{3}k_4 + \mathcal{O}(h^4), \\ h y^{(1)}(\frac{H}{2}) &= -\frac{3}{8}k_1 + \frac{9}{4}k_2 + \frac{9}{8}k_3 - 2k_4 + \mathcal{O}(h^4), \\ y(\frac{H}{2}) &= y_0 + \frac{3}{16}k_1 + \frac{18}{8}k_2 + \frac{9}{16}k_3 - \frac{12}{8}k_4 + \mathcal{O}(h^4). \end{aligned} \tag{6.21}$$

The first element of the Nordsieck vector, $y(\frac{H}{2})$, is computed by Taylor expansion.

It follows that

$$\Gamma_{\frac{H}{2}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{3}{16} & -\frac{3}{8} & -\frac{1}{6} & 1 \\ \frac{18}{8} & \frac{9}{4} & 0 & -2 \\ \frac{9}{16} & \frac{9}{8} & \frac{3}{2} & 1 \\ -\frac{12}{8} & -2 & -\frac{4}{3} & 0 \end{pmatrix}. \tag{6.22}$$

If the derivatives are instead computed at the origin, the matrix above becomes

$$\Gamma_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\frac{5}{3} & 1 \\ 0 & 0 & 3 & -2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{4}{3} & 0 \end{pmatrix}. \tag{6.23}$$

The matrix A of coefficients $\alpha_{i,j}$ in equation (6.2) is obtained from Equations (6.15) and (6.16)

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \frac{3}{2} & 0 & 0 & 0 \end{pmatrix}. \tag{6.24}$$

For a fourth-order method we need at least six function evaluations, [5], and the relevant matrices Γ_0 and A are

$$\Gamma_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\frac{5}{6} & \frac{4}{9} & -\frac{1}{9} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{4}{9} & \frac{1}{9} \\ 0 & 0 & \frac{7}{3} & -\frac{19}{9} & \frac{7}{9} \\ 0 & 0 & -3 & \frac{10}{3} & -\frac{4}{3} \\ 0 & 0 & 1 & -\frac{11}{9} & \frac{5}{9} \end{pmatrix},$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ \frac{3}{4} & 0 & \frac{9}{4} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} & 2 & 0 & 0 \\ \frac{1}{12} & 2 & \frac{1}{4} & \frac{2}{3} & 2 & 0 \end{pmatrix}.$$

The cost of this process in terms of function evaluations is $1 + \frac{p(p-1)}{2}$, since the interval H is integrated by Euler's method m times with step size $\frac{H}{m}$ for $m = p, p-1, \ldots, 1$. For the first value of m we have p function evaluations because the initial value of y' has to be evaluated once, so for the next value of m we have $p-2$ function evaluations, and so on.

We constructed a Nordsieck vector with third-order accuracy. To do this we used four stages k_1, k_2, k_3, k_4 as in (6.15) and (6.16) with lower order and an extrapolation technique. It can be shown that there exists no method of the same order with less stages and thus less function evaluations.

5 Order tests

To verify that the starter indeed achieves the expected order we consider the harmonic oscillator

$$y'' = -4y, \qquad y_0 = 1, \;\; y_0' = 0.$$

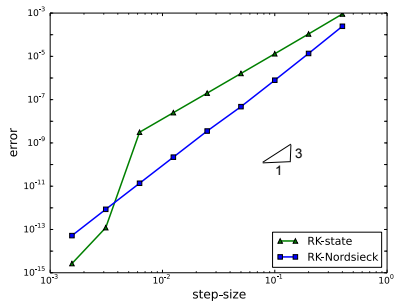

Figure 6.2: Both Runge--Kutta starters achieved third-order accuracy when solving the harmonic oscillator problem with the 3-step Adams-Moulton method

6 The bouncing ball test example

In this section we demonstrate the method on the example of a bouncing ball with linear damping $d = 0.1$:

$$\begin{aligned} \dot{y_1} &= y_2 \\ \dot{y_2} &= -dy_1 + 9.81 \end{aligned}$$

The bounces are modeled using a coefficient of restitution was chosen to be $c = 0.88$ to give the system sufficiently many impacts to be able to make a statement about the effect of restarting, see Fig. 6.12. The model includes two events, one to trigger bouncing and a second one which triggers the upper turning point. At the upper turning point the differential equation and its states remain unaltered, and only the switch to control the bouncing event becomes activated. At the bouncing event the velocity $\dot{y}(t^-)$ is altered to $\dot{y}(t^+) = -c\dot{y}(t^-)$.

Figure 6.3: A simulation of a bouncing ball (damping: $d = 0.1$, coefficient of restitution $c = 0.88$).

In Fig. (6.13) the step size and order history of both restarting techniques is compared. The classical starting procedure clearly shows a drop in order and step size. The method recovers quite quickly from the reduced step size as LSODAR allows exceptionally big step size changes during the initialization phase.

The run statistics, cf. Tab. 6.8 show the effect of the Runge–Kutta starter in Assimulo. The gain in the number of function evaluations for this example is about 58%.

7 Conclusions

The aim of this paper is to study the effect of Runge–Kutta restarting techniques on the performance of the simulation of hybrid systems. Tests were made on a system with relatively small numbers of discontinuities. The tests give a clear indication that investigating a more sophisticated restarting procedure like the fourth-order Runge–Kutta starter presented here has a potential impact on the overall performance of an simulator.

The flexibility in selecting the order of the restarter as well as doing error control of the restarter is the topic of future research.

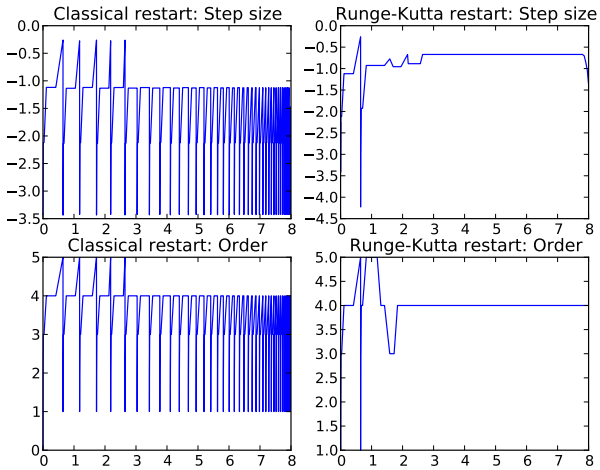

Figure 6.4: Comparison of the step size and order history for the two restarting approaches. A logarithmic scale is used for the step size plot.

	Classic starter	Runge–Kutta starter
# steps	455	129
# function evals	1027	428
# event function evals	919	538
# events	38	37

Table 6.1: Run time statistics for the bouncing ball example with absolute and relative tolerance set to 10^{-8}.

Bibliography

[1] J. Åkesson, M. Gäfvert, and H. Tummescheit. JModelica—an open source platform for optimization of modelica models. In *Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, Vienna, Austria, 2009. TU Wien.

[2] C. Andersson. *Assimulo: A new Python based class for simulation of complex hybrid DAEs and its integration in JModelica.org*. Lund University, 2011.

[3] C. Andersson, J. Åkesson, C. Führer, and M. Gäfvert. Import and export of functional mock-up units in JModelica.org. In *8th International Modelica Conference 2011*, Dresden, Germany, mar 2011.

[4] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, et al. The functional mockup interface for tool independent exchange of simulation models. In *Modelica 2011 Conference, March*, pages 20–22, 2011.

[5] C. W. Gear. Runge–Kutta starters for multistep methods. *ACM Transactions on Mathematical Software (TOMS)*, 6(3):263–279, 1980.

[6] A. Ralston. Runge-Kutta methods with minimum error bounds. *Mathematics of computation*, 16(80):431–437, 1962.

[7] H. J. Stetter. Asymptotic expansions for the error of discretization algorithms for non-linear functional equations. *Numerische Mathematik*, 7(1):18–31, 1965.

[8] R. von Schwerin and H. G. Bock. A Runge–Kutta starter for a multistep method for differential-algebraic systems with discontinuous effects. *Applied Numerical Mathematics*, 18(1):337–350, 1995.

Paper II

Journal of Computational and Applied Mathematics 316 (2017) 287–297

Runge–Kutta Restarters for Multistep Methods in Presence of Frequent Discontinuities

Fatemeh Mohammadi Carmen Arévalo Claus Führer

Numerical Analysis, Center for Mathematical Sciences, Lund University,
Sölvegatan 18, SE-22100 Lund, Sweden

Abstract

Differential equations with discontinuities or differential equations coupled to discrete systems require frequent re-initializations of the numerical solution process. The classical starting process of multistep methods, based on winding-up the order in the initialization phase, is computationally expensive when frequent discontinuities occur. Instead we propose to use the stage values or weight vectors of these specially constructed explicit Runge–Kutta methods for starting processes. Two practical examples demonstrate these methods.

Keywords: Multistep methods, Runge–Kutta methods, Discontinuities, Error estimation

1 Introduction

The overall performance of the simulation of models with discontinuities depends strongly on the methods for restarting the integration after an event detection. Initialization of k-step methods requires k starting values.

There are three alternative starting strategies for obtaining the initial values; however, only two of these starting schemes have been implemented and tested. The first strategy is to apply a high order Runge–Kutta (RK) method repeatedly to generate the starting values. This requires several costly evaluations of the equation. The second one is to approximate the required starting values by applying multistep methods of successively higher order. The particular methods chosen for the starting processes can affect the overall order of the algorithm.

In this article we will look at the third alternative for starting multistep methods. This one provides high order starting values with a minimal number of function evaluations. Gear [3] proposed a starter for multistep methods that is based on the Nordsieck vector containing the scaled derivatives of the approximated solution values at a given time point. This scheme needs approximately half the number of function evaluations of the first method and provides more accurate initial values than the second scheme. It uses a representation of RK methods, based on extrapolation techniques. In this paper we take up this idea and present two other families of the RK starters of order 2 to 4. In addition, we construct an embedded RK error estimator for the starting step.

We first review the aforementioned schemes for starting multistep methods and then introduce our two families of RK starters followed by a calculation of the error estimator for the first step. Finally, we present some numerical results of the implementation of our starters in a modification of LSODAR [5], which is a variant of the ODE solver LSODE equipped with event detection.

Consider an initial value problem of the form

$$\dot{y} = f(t, y), \quad y_0 = y(t_0), \tag{6.25}$$

with a sufficiently smooth function f. An s-stage explicit RK method with nodes $\{c_i\}_{i=1}^{s}$, weights $\{b_{1i}\}_{i=1}^{s}$ and coefficients $\{a_{ij}\}_{1\leq j<i\leq s}$ applied to (6.25) is defined

by

$$Y_i = y_n + H \sum_{j=1}^{i-1} a_{ij} K_j,$$
$$K_i = f(t_n + c_i H, Y_i), \quad i = 1, \cdots, s, \tag{6.26}$$

where Y_i and K_i are the stage values and the stage derivatives respectively. The numerical solution at $t_n + H$ is approximated by

$$y_{n+1} = y_n + H \sum_{j=1}^{s} b_{1j} K_j. \tag{6.27}$$

The coefficients of the RK method can be described using a *Butcher tableau*,

$\mathbf{c}$	A
θ_1	$\mathbf{b_1}^{\mathrm{T}}$

where the stage vector is $\mathbf{c} = (c_1, c_2, \ldots, c_s)^{\mathrm{T}}$, the weight vector is $\mathbf{b_1}^{\mathrm{T}} = (b_{1,1}, b_{1,2}, \ldots, b_{1,s})$, and the coefficient matrix is $A = \{a_{ij}\}$. In this paper we will introduce several weight vectors and define $\theta_i := \sum_{j=1}^{s} b_{ij}$ where index i refers to the i^{th} weight vector. The consistency condition of an RK method gives $\theta_1 = 1$, but we introduce the notation here as we will later use weight vectors with $\theta_i \neq 1$.

2 Several-step single-stage starter

During the 60's a common technique to generate the starting values for multistep methods was to apply an RK step of order k repeatedly. Figure 6.5 illustrates the idea of applying $k - 1$ RK steps to start a k-step method.

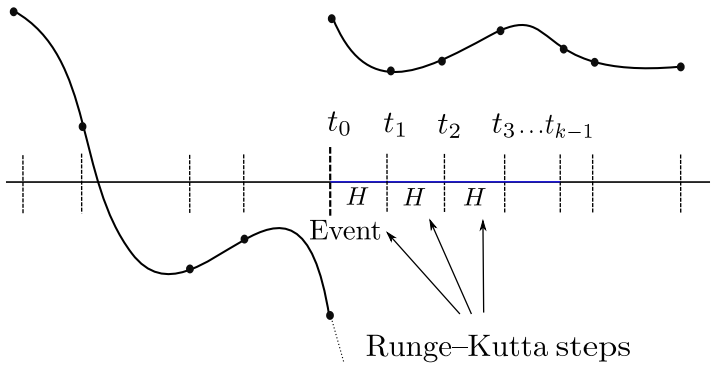

Figure 6.5: $k-1$ RK steps with fixed step size, H, taken to restart the integration after detecting a discontinuity (Event).

This starter is chosen in such a way that it generates initial values of the same order as the multistep method used thereafter. However, it needs at least $k(k-1)$ function evaluations which is far more than other starting schemes for large value of k.

3 Winding-up states

Most current multistep ODE solvers, such as LSODAR, DASSL and CVODE, use a self-starting scheme developed by Gear [4] in 1971. The scheme is started by a one-step method with a small step size, and then the order and the step size are consecutively increased until the working order and the desired accuracy are reached (see Figure 6.6). For instance, when a 3-step method is started, two more initial solution points must be computed in addition to the initial value before entering the main time-stepping loop.

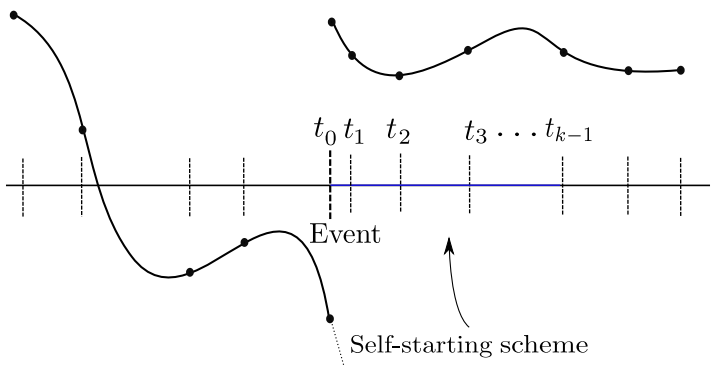

Figure 6.6: Self-starting scheme beginning with a one-step method and a small step size. Both the order and step size are successively increased.

4 Single-step several-stage starter

The idea of (re-)starting multistep methods with highly accurate initial values on the one hand and a minimal number of function evaluations on the other hand, motivated us to look for a family of explicit RK methods with embedded error estimation. These RK starters embrace the preferences of the first two alternatives for starting the multistep methods, i.e., the minimal number of function evaluations of winding-up schemes and the highly accurate initial values of several-step single-stage starters.

We introduce two families of single-step RK starters. The first family, $\mathscr{R}_1$, uses the internal stages of an RK method to obtain the starting values required by the multistep method. The second one, $\mathscr{R}_2$, uses the method's weight vectors to approximate the solution at several points within a single RK step. Both families provide error estimation for the RK step.

4.1 $\mathscr{R}_1$: Runge–Kutta starters with special internal stages

For the purpose of starting a k-step method we can construct RK methods with internal stage values of order k. For simplicity, these stages are chosen at equidistant time points. Figure 6.7 illustrates one of these starters.

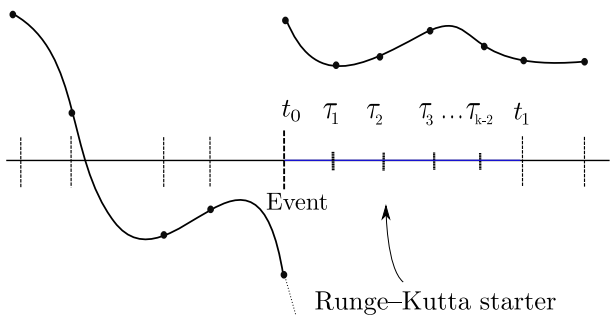

Figure 6.7: A single-step RK starter with equidistant k^{th} order stage values at τ_i used for starting the k-step method.

To construct this multistep starter we need to solve a nonlinear system of order conditions, that are derived by comparing the Taylor series of the exact solution to that of the numerical solution. The order conditions for RK methods up to order 4 are shown in Table 6.2 [2, 7].

Order k	Order conditions for weight vectors
1	$\sum_{j=1}^{s} b_{ij} = \theta_i$.
2	$\sum_{j=1}^{s} b_{ij} c_j = \frac{\theta_i^2}{2}$.
3	$\sum_{j=1}^{s} b_{ij} c_j^2 = \frac{\theta_i^3}{3}$, $\quad \sum_{j,k=1}^{s} b_{ik} a_{kj} c_j = \frac{\theta_i^3}{6}$.
4	$\sum_{j=1}^{s} b_{ij} c_j^3 = \frac{\theta_i^4}{4}$, $\quad \sum_{j,k=1}^{s} b_{ij} c_j a_{jk} c_k = \frac{\theta_i^4}{8}$, $\sum_{j,k=1}^{s} b_{ij} a_{jk} c_k^2 = \frac{\theta_i^4}{12}$, $\quad \sum_{j,k,l=1}^{s} b_{ij} a_{jk} a_{kl} c_l = \frac{\theta_i^4}{24}$.

Table 6.2: RK coefficients and k^{th} order weight vectors $\mathbf{b_i}$ are computed by solving a system of order conditions up to k. To attain order k at $t_n + \theta_i H$ all equations up to and including row k must be satisfied. Index i refers to the i^{th} weight vector. When $i = 1$, these conditions reduce to the standard case, as $\theta_1 = 1$.

The order conditions for the weight vectors in Table 6.2 can be modified to obtain internal stages of order k. The order conditions on the i^{th} internal stage to attain order k are given in Table 6.3.

Order k	Order conditions for internal stages
1	$\sum_{j=1}^{s} a_{ij} = c_i$.
2	$\sum_{j=1}^{s} a_{ij} c_j = \frac{c_i^2}{2}$.
3	$\sum_{j=1}^{s} a_{ij} c_j^2 = \frac{c_i^3}{3}$, $\quad \sum_{j,k=1}^{s} a_{ik} a_{kj} c_j = \frac{c_i^3}{6}$.
4	$\sum_{j=1}^{s} a_{ij} c_j^3 = \frac{c_i^4}{4}$, $\quad \sum_{j,k=1}^{s} a_{ij} c_j a_{jk} c_k = \frac{c_i^4}{8}$, $\sum_{j,k=1}^{s} a_{ij} a_{jk} c_k^2 = \frac{c_i^4}{12}$, $\quad \sum_{j,k,l=1}^{s} a_{ij} a_{jk} a_{kl} c_l = \frac{c_i^4}{24}$.

Table 6.3: k^{th} order RK coefficients are computed by solving a system of order conditions up to k. Order k is attained for the approximation at the internal point $t_n + c_i H$.

This family of RK starters has an embedded formula that enables error estimation. For an order k method, an approximate solution of order $k - 1$ is generated at $t_n + H$ as

$$\tilde{y}_{n+1} := y_n + H \sum_{j=1}^{s} a_{\ell j} K_j,$$

where the index ℓ refers to the internal stage of order $k - 1$ and $\sum_{j=1}^{s} a_{\ell j} = 1$.

The difference,

$$y_{n+1} - \tilde{y}_{n+1} = y_n + H\sum_{j=1}^{s} b_{1j}K_j - y_n - H\sum_{j=1}^{s} a_{\ell j}K_j \tag{6.28}$$
$$= H\sum_{j=1}^{s}(b_{1j} - a_{\ell j})K_j$$

is then an error estimate of order $k-1$ at $t_n + H$. The error, defined by $e_j = b_{1j} - a_{\ell j}$, is denoted by vector $\mathbf{e}$ and is included after the last row of the Butcher tableau.

A 2nd order Runge–Kutta starter

A 2nd order RK starter of this family is Heun's method. Its Butcher tableau is shown in Table 6.4.

0		
1	1	
1	$\frac{1}{2}$	$\frac{1}{2}$
$\mathbf{e}^{\mathrm{T}}$	$-\frac{1}{2}$	$\frac{1}{2}$

Table 6.4: Heun's method with error estimation. The error coefficients are given in vector $\mathbf{e}$.

The error estimation is computed by comparing 2nd and 1st order y_{n+1} and $\tilde{y}_{n+1} = Y_2$, respectively. This corresponds to the second and third rows in the tableau.

A 3rd order Runge–Kutta starter

In [7], Schwerin et al. introduced an RK starter for the 2-step Adams-Moulton method. This 3rd order RK starter has 6 internal stages, but here we present a 3rd order RK starter with only 5 internal stages. Approximated values Y_4 and Y_5 are of order 3 while y_{n+1} is 4th order accurate. The error can be estimated by comparing $\tilde{y}_{n+1} = Y_4$ with y_{n+1}, 4th and 6th rows of Figure 6.8.

0					
$\frac{1}{2}$	$\frac{1}{2}$				
$\frac{3}{4}$	0	$\frac{3}{4}$			
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$		
$\frac{1}{2}$	$\frac{17}{72}$	$\frac{1}{6}$	$\frac{2}{9}$	$-\frac{1}{8}$	
1	$\frac{1}{6}$	0	0	$\frac{1}{6}$	$\frac{2}{3}$
$\mathbf{e}^{\mathrm{T}}$	$-\frac{1}{18}$	$-\frac{1}{3}$	$-\frac{4}{9}$	$\frac{1}{6}$	$\frac{2}{3}$

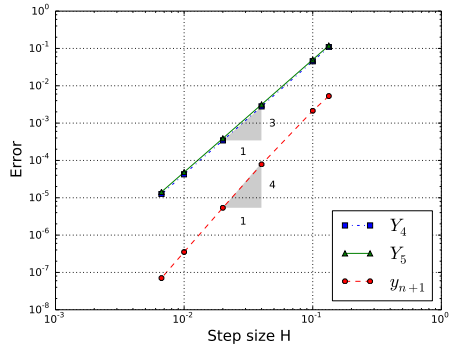

Figure 6.8: The order plot of the solution of the pendulum equation, $\ddot{\phi} + g\sin(\phi) = 0,\ \ \phi(0) = 1,\ \dot{\phi}(0) = 0$, by the 3rd order RK starter with internal stages Y_4 and Y_5 of order 3 and y_{n+1} of order 4. Error coefficients are given in $\mathbf{e}$.

4th order Runge–Kutta starter

The conditions that Schwerin et al. imposed did not allow for higher order RK starters. A premise of their construction is to have an internal stage of order k as soon as possible and then maintain this order for all subsequent stages. By dropping the last requirement, i.e., by not demanding that all subsequent stages be of order k, it is possible to construct RK starters of higher orders. The constructed 4th order RK starter based on modified premises (see Figure 6.9) has internal stages Y_5, Y_7 and the result y_{n+1} of order 4, and internal stage Y_6 of order 3.

0							
$\frac{1}{6}$	$\frac{1}{6}$						
$\frac{1}{6}$	0	$\frac{1}{6}$					
$\frac{1}{3}$	0	0	$\frac{1}{3}$				
$\frac{1}{3}$	$\frac{1}{18}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{18}$			
1	$\frac{25}{10}$	-3	-3	$\frac{225}{100}$	$\frac{225}{100}$		
$\frac{2}{3}$	$\frac{10}{45}$	$-\frac{8}{45}$	$-\frac{8}{45}$	$-\frac{4}{45}$	$\frac{13}{15}$	$\frac{1}{45}$	
1	$\frac{29}{1062}$	$\frac{83}{531}$	$\frac{83}{531}$	$\frac{83}{1062}$	$\frac{2}{531}$	$\frac{56}{531}$	$\frac{251}{531}$
$\mathbf{e}^{\mathrm{T}}$	$-\frac{1313}{531}$	$\frac{1676}{531}$	$\frac{1676}{531}$	$-\frac{4613}{2124}$	$-\frac{4771}{2124}$	$\frac{56}{531}$	$\frac{251}{531}$

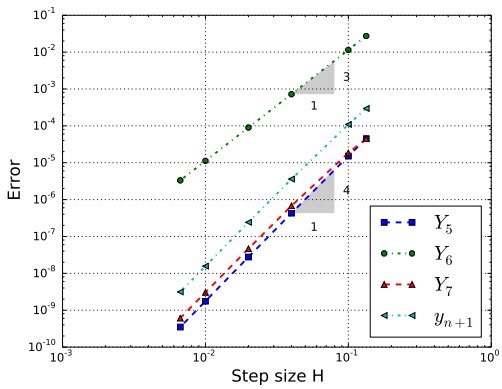

Figure 6.9: A 4th order RK starter with three equidistant stages of order 4 and its corresponding order plot of the solution of the pendulum equation, $\ddot{\phi} + g\sin(\phi) = 0,\ \ \phi(0) = 1,\ \dot{\phi}(0) = 0$. The 3rd order Y_6 is used for error estimation.

The error for the RK step can be estimated by comparing $\tilde{y}_{n+1} = Y_6$ and y_{n+1}, which are of order 4 and 3 respectively.

4.2 $\mathscr{R}_2$: Runge–Kutta starters with distinct weight vectors

We can construct RK starters with at least $k-1$ weight vectors of order k. These weight vectors allow for approximate solutions at distinct fractions of the first step, and together with the initial value they can be used to initialize the k-step multistep method. This family of RK starters also provides an error estimator for the RK step.

The 2nd order RK starter of this family is identical to the one in the $\mathscr{R}_1$ family (see Table 6.4).

A 3rd order Runge–Kutta starter

Three initial values are applied to start a 3-step method. Thus, we need to have two weight vectors of order 3, denoted by $\mathbf{b_1}$ and $\mathbf{b_2}$. Furthermore, a weight vector $\mathbf{b_3}$ is utilized for error estimation. Solving the nonlinear system of order conditions with three internal stages leads to two identical weight vectors. Thus we need at least four stages to derive a method which produces two additional starting values as well as an error estimation. This 3rd order RK starter has the following tableau,

$\mathbf{c}$	A
θ_1	$\mathbf{b_1}^{\mathrm{T}}$
θ_2	$\mathbf{b_2}^{\mathrm{T}}$
θ_3	$\mathbf{b_3}^{\mathrm{T}}$

with $\mathbf{b_1}$ and $\mathbf{b_2}$ of order 3 and $\mathbf{b_3}$ of order 2.

We substitute the 3rd order Ralston method [6], an RK method with minimal truncation error, in the first three stages of the Butcher tableau; $\mathbf{b_1}$ is the weight vector of this method and has order 3. The order conditions of Table 6.2 are solved to attain 3rd and 2nd order accurate weight vectors $\mathbf{b_2}$ and $\mathbf{b_3}$, respectively. We choose $\theta_2 = \frac{1}{2}$ to simply start with equidistant initial values and $\theta_3 = 1$ to estimate the error. The approximated solutions using $\mathbf{b_1}$ and $\mathbf{b_2}$ are the two additional values needed to start the 3-step method (see Figure 6.10)

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$-\frac{19}{16}$	$\frac{29}{16}$	$\frac{3}{8}$	
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	0
$\frac{1}{2}$	$\frac{1}{12}$	$\frac{13}{12}$	-1	$\frac{1}{3}$
1	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{6}$	$\frac{1}{4}$
$\mathbf{e}^{\mathrm{T}}$	$-\frac{1}{9}$	$\frac{1}{12}$	$\frac{5}{18}$	$-\frac{1}{4}$

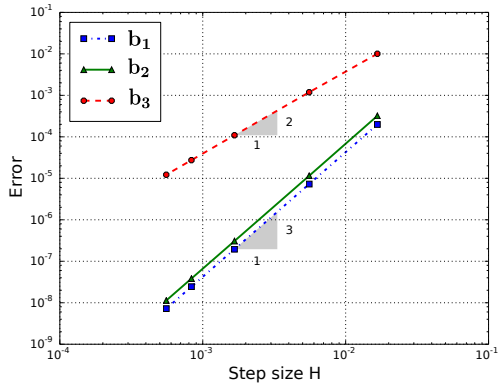

Figure 6.10: The order plot for the solution of the pendulum equation, $\ddot{\phi} + g\sin(\phi) = 0,\ \ \phi(0) = 1,\ \dot{\phi}(0) = 0$, with the 3rd order RK starter confirms the 3rd order accuracy for the results obtained from weight vectors $\mathbf{b_1}$ and $\mathbf{b_2}$. Note that $\mathbf{b_3}$ is of order 2.

4th order Runge–Kutta starter

Four state values are needed to start a 4-step method. Considering the initial value of the problem we need to generate three more values. Thus the RK method needs three 4th order weight vectors at distinct fractions of the RK step size, $c_i H$. For the purpose of error estimation, the 3rd order weight vector $\mathbf{b_4}$ is added. The Butcher tableau of a 4th order RK starter with weight vectors $\mathbf{b_1}$, $\mathbf{b_2}$ and $\mathbf{b_3}$ of order 4, and b_4 of order 3 has the form

$\mathbf{c}$	A
θ_1	$\mathbf{b_1^{\mathrm{T}}}$
θ_2	$\mathbf{b_2^{\mathrm{T}}}$
θ_3	$\mathbf{b_3^{\mathrm{T}}}$
$\mathbf{e}^{\mathrm{T}}$	$\mathbf{b_4^{\mathrm{T}}}$

To describe our methodology, we build up this method step by step. The 4th order Ralston method (see Table 6.5), that has minimal truncation error, is used for the first four stages of the RK method.

0					
$\frac{2}{5}$	$\frac{2}{5}$				
$\frac{3}{5}$	$-\frac{3}{20}$	$\frac{3}{4}$			
1	$\frac{19}{44}$	$-\frac{15}{44}$	$\frac{40}{44}$		
$\frac{1}{2}$	$-\frac{31}{64}$	$\frac{185}{192}$	$\frac{5}{64}$	$-\frac{11}{192}$	
1	$\frac{11}{72}$	$\frac{25}{72}$	$\frac{25}{72}$	$\frac{11}{72}$	0
$\frac{3}{5}$	$\frac{699}{5000}$	$\frac{81}{200}$	$-\frac{39}{200}$	$\frac{99}{5000}$	$\frac{144}{625}$

Table 6.6: The 4th order RK starter with two weight vectors, $\mathbf{b_1}$ and $\mathbf{b_2}$ of order 4.

0				
$\frac{2}{5}$	$\frac{2}{5}$			
$\frac{3}{5}$	$-\frac{3}{20}$	$\frac{3}{4}$		
1	$\frac{19}{44}$	$-\frac{15}{44}$	$\frac{40}{44}$	
1	$\frac{11}{72}$	$\frac{25}{72}$	$\frac{25}{72}$	$\frac{11}{72}$

Table 6.5: 4th order Ralston method

We may choose $\theta_2 = \frac{3}{5}$, and obtain an overdetermined system with four unknowns and seven equations. Thus we add an extra internal stage (see Table 6.6).

In the next step we are looking for a new weight vector of order 4, $\mathbf{b_3}$. It is not possible to find $\mathbf{b_3}$ with just these five stages, so the weight vector of the Ralston method is moved to the upper part of Table 6.6 and it becomes an internal stage of the method. Then, with one more internal stage and the choice of $\theta_3 = \frac{2}{5}$, the method in Table 6.7 is obtained. Now we have 3 weight vectors of order 4.

Finally, a 3rd order weight vector $\mathbf{b_4}$ with $\theta_4 = 1$ is derived to obtain an error estimation by comparing the solutions of 4th order $\mathbf{b_1}$ to 3rd order $\mathbf{b_4}$. The method is shown in Figure 6.11.

0						
$\frac{2}{5}$	$\frac{2}{5}$					
$\frac{3}{5}$	$-\frac{3}{20}$	$\frac{3}{4}$				
1	$\frac{19}{44}$	$-\frac{15}{44}$	$\frac{40}{44}$			
$\frac{1}{2}$	$-\frac{31}{64}$	$\frac{185}{192}$	$\frac{5}{64}$	$-\frac{11}{192}$		
1	$\frac{11}{72}$	$\frac{25}{72}$	$\frac{25}{72}$	$\frac{11}{72}$	0	
$\frac{3}{5}$	$\frac{699}{5000}$	$\frac{81}{200}$	$-\frac{39}{200}$	$\frac{99}{5000}$	$\frac{144}{625}$	0
$\frac{2}{5}$	$\frac{802}{5625}$	$\frac{68}{225}$	$-\frac{67}{225}$	$-\frac{143}{5625}$	$\frac{144}{625}$	$\frac{6}{125}$

Table 6.7: The 4th order RK starter with two weight vectors, $\mathbf{b_2}$, $\mathbf{b_3}$ and an internal stage Y_6 of order 4.

0						
$\frac{2}{5}$	$\frac{2}{5}$					
$\frac{3}{5}$	$-\frac{3}{20}$	$\frac{3}{4}$				
1	$\frac{19}{44}$	$-\frac{15}{44}$	$\frac{40}{44}$			
$\frac{1}{2}$	$-\frac{31}{64}$	$\frac{185}{192}$	$\frac{5}{64}$	$-\frac{11}{192}$		
1	$\frac{11}{72}$	$\frac{25}{72}$	$\frac{25}{72}$	$\frac{11}{72}$	0	
$\frac{3}{5}$	$\frac{699}{5000}$	$\frac{81}{200}$	$-\frac{39}{200}$	$\frac{99}{5000}$	$\frac{144}{625}$	0
$\frac{2}{5}$	$\frac{802}{5625}$	$\frac{68}{225}$	$-\frac{67}{225}$	$-\frac{143}{5625}$	$\frac{144}{625}$	$\frac{6}{125}$
1	$\frac{9929}{78075}$	$\frac{871}{2082}$	$\frac{418}{3123}$	$\frac{9581}{52050}$	$\frac{3554}{26025}$	0
$\mathbf{e}^{\mathrm{T}}$	$\frac{1777}{69400}$	$-\frac{1777}{24984}$	$\frac{1777}{8328}$	$-\frac{19547}{624600}$	$-\frac{3554}{26025}$	0

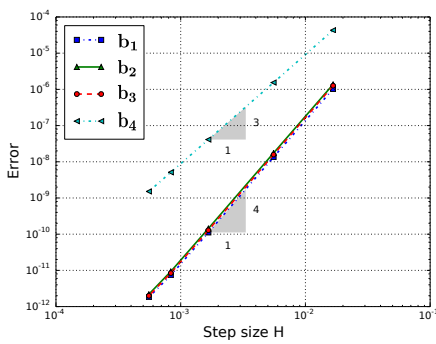

Figure 6.11: The order plot for the solution of the pendulum equation, $\ddot{\phi} + g\sin(\phi) = 0,\ \ \phi(0) = 1,\ \dot{\phi}(0) = 0$, shows 4th order accuracy for weight vectors $\mathbf{b_1}$, $\mathbf{b_2}$ and $\mathbf{b_3}$, and 3rd order for $\mathbf{b_4}$ from the given Butcher tableau.

Implementation

The RK starter families of Section 4 have been prototyped in Assimulo, a Python wrapper for ODE software, to solve the differential equations with solver packages like ODEPACK or SUNDIALS. Assimulo provides tools for discontinuity handling and event detection functions, or so-called switching functions, and allows access to a large variety of professional and experimental solvers [1]. We have chosen to implement our algorithms in Python, and to incorporate them in Assimulo in order to try them out with LSODAR.

The order of the RK starter is determined by the last successfully attempted order of the multistep method before the discontinuity is detected. If this value is greater than 4 then the starting order is set to 4 automatically. The initial step size is taken as the last successful step size before the discontinuity.

5 Numerical experiments

We demonstrate the effect of the two RK starter families on the numerical solution of two test models that contain frequent discontinuities in their solutions.

5.1 The bouncing ball

The first model is a bouncing ball with linear damping ($d = 0.1$),

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -d\, y_2 + 9.81 \end{aligned}$$

The bounces are modeled choosing a coefficient of restitution ($c = 0.88$) in order to give the system sufficiently many impacts to be able to make a statement about the effect of restarting (see Figure 6.12). The model includes two events, one to trigger bouncing and a second one which triggers the upper turning point. At the upper turning point the differential equation and its states remain unaltered, and only the switch to control the bouncing event becomes activated. At the bouncing event the velocity $\dot{y}(t^-)$ is altered to $\dot{y}(t^+) = -c\, \dot{y}(t^-)$.

In Figure 6.13 the step size and order history of the restarting techniques are compared. The winding-up starting procedure clearly shows a drop in order and step

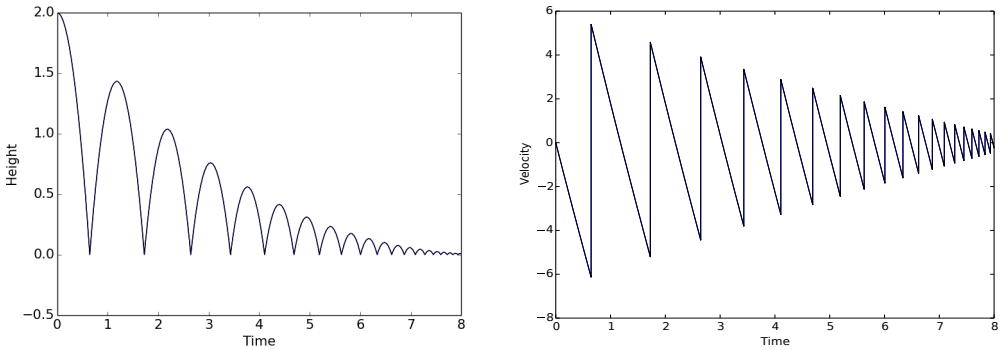

Figure 6.12: Simulation of a bouncing ball (damping: $d = 0.1$, coefficient of restitution: $c = 0.88$).

size. The method recovers quite quickly from the reduced step size as LSODAR allows exceptionally large step size changes during the initialization phase.

Table 6.8 shows the effect of the RK starters $\mathscr{R}_1$ and $\mathscr{R}_2$ in LSODAR. The reduction in the number of function evaluations for this example is more than 60%.

	Winding-up	RK starter $\mathscr{R}_1$	RK starter $\mathscr{R}_2$
# steps	642	159	140
# function evals	1477	449	376
# event function evals	1074	590	532

Table 6.8: Run time statistics for the bouncing ball example with absolute and relative tolerances set to 10^{-8}.

All starting schemes detected 38 events.

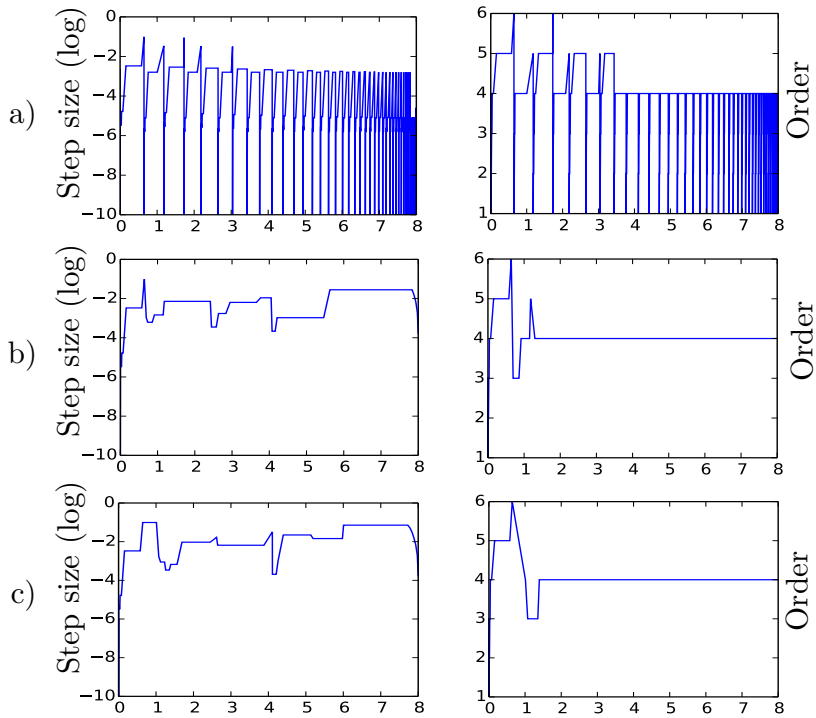

Figure 6.13: Comparison of the step size and order history for three restarting approaches, a) Winding-up, b) RK starter $\mathscr{R}_1$, and c) RK starter $\mathscr{R}_2$ in the simulation of the bouncing ball. A logarithmic scale is used for the step size variable.

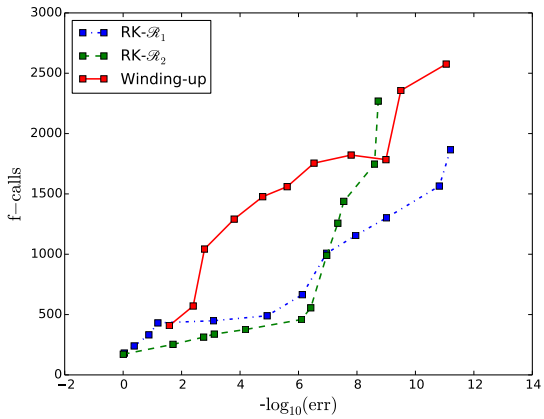

Figure 6.14: Work-precision diagram for the solution of the bouncing ball with winding-up, RK $\mathscr{R}_1$ and $\mathscr{R}_2$ starters.

In Figure 6.14 the precision, $\log_{10}(err)$, is compared to the number of function evaluations for tolerances ranging from 10^{-4} to 10^{-14} and it appears that both

single-step several-stage starters perform better than the winding-up scheme. The best method for this model for tighter tolerances is RK starter $\mathscr{R}_1$.

5.2 The pendulum

This model is described in the set of examples in Assimulo [1] and it demonstrates a free pendulum that bounces against an object situated at an angle of -45 degrees. The impact triggers the discontinuity in the velocity by changing its sign. The initial value problem that describes the motion of the pendulum is

$$\ddot{\phi} = -g\sin(\phi), \quad \phi(0) = \frac{\pi}{2}, \quad \text{and } \dot{\phi}(0) = 0. \tag{6.29}$$

At the impact event, the angular velocity of the pendulum $\dot{\phi}(t^-)$ is altered to $\dot{\phi}(t^+) = -c\,\dot{\phi}(t^-)$, where c is the coefficient of restitution (see Figure 6.15).

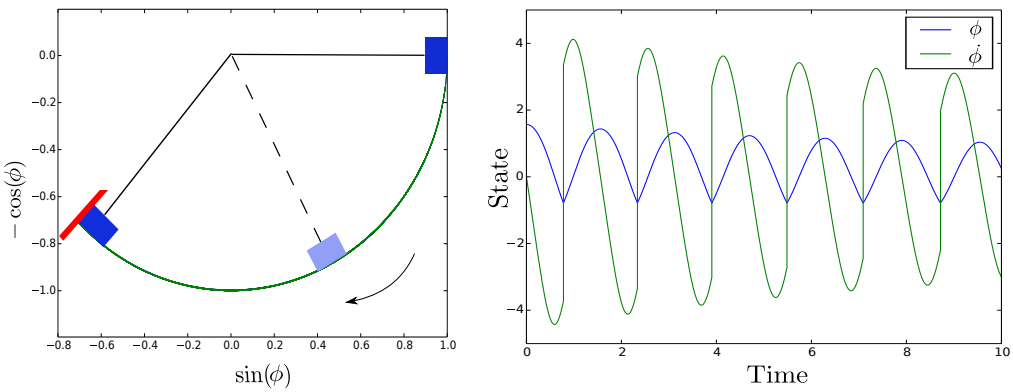

Figure 6.15: Simulation of the pendulum bounces against an object situated at an angle of -45 degrees for $c = 0.9$.

The performance of the RK starters is presented in Table 6.9 and shows little gain in comparison to the winding-up scheme.

Starter	Winding-up	RK $\mathscr{R}_1$	RK $\mathscr{R}_2$
# steps	510	457	448
# function evals	1073	1028	985
# event function evals	630	574	566

Table 6.9: Run time statistics for the pendulum with tolerance set to 10^{-7} and the simulation time is 10s.

The number of detected events for all the starters in Table 6.9 is 12.

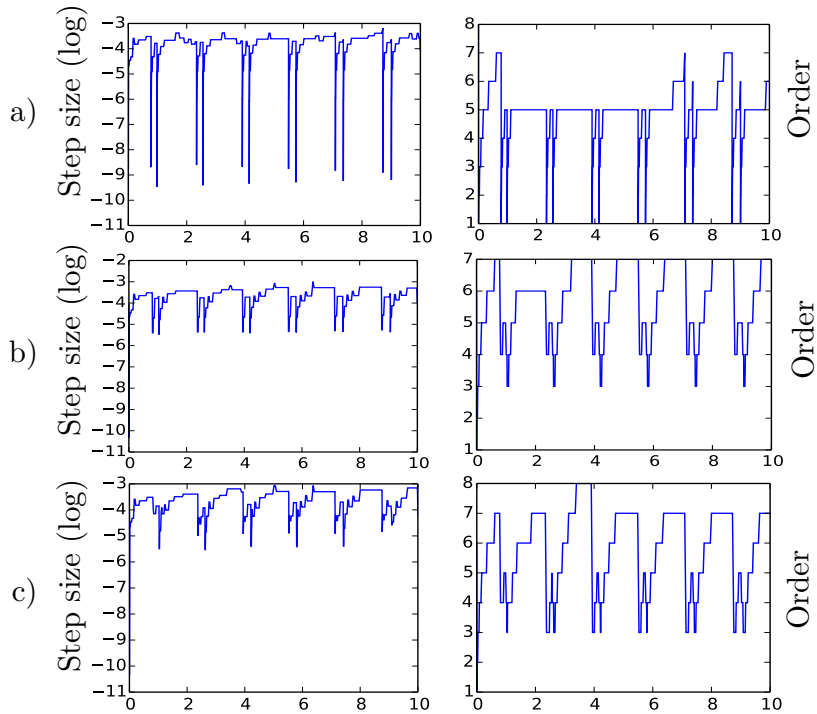

Figure 6.16: Comparison of the step size and order history for the restarting approaches, a) Winding-up, b) RK starter $\mathscr{R}_1$, and c) RK starter $\mathscr{R}_2$ in the simulation of the pendulum. A logarithmic scale is used for the step size variable.

In Figure 6.16, the orders and step sizes of winding-up method drop down after each discontinuity while the RK starters $\mathscr{R}_1$ and $\mathscr{R}_2$ maintain order 3 with larger step sizes.

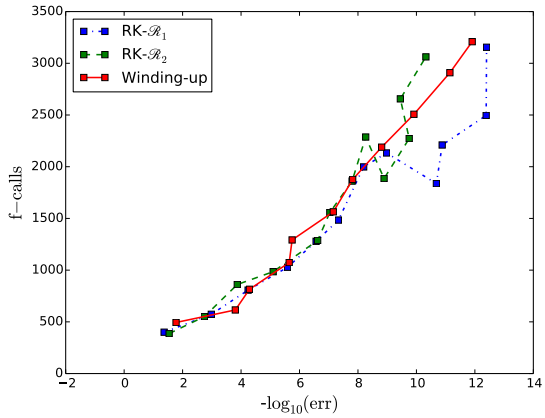

Figure 6.17: Function evaluations versus global error diagram for the solution of the pendulum with Winding-up, RK $\mathscr{R}_1$ and $\mathscr{R}_2$ starters. The tolerances range from 10^{-4} to 10^{-15}.

The RK starter $\mathscr{R}_1$ is clearly superior for tolerances lower than 10^{-7} (see Figure 6.17).

6 Conclusion

We have constructed two families of RK starters and studied the effect of these starters on the performance of the simulation of systems with discontinuities. The flexibility in selecting the order of the (re-)starter as well as being able to perform an error estimation are the main features of these starters. Furthermore, in presence of frequent discontinuities the performance gain in terms of function evaluations and accuracy is significant for tighter tolerances compare to the winding-up scheme, as evidenced by two examples.

Bibliography

[1] C. Andersson, C. Führer, and J. Åkesson. Assimulo: A unified framework for ODE solvers. *Mathematics and Computers in Simulation*, 116:26 – 43, 2015.

[2] J. C. Butcher. On fifth order Runge-Kutta methods. *BIT Numerical Mathematics*, 35(2):202–209, 1995.

[3] C. W. Gear. Runge–Kutta starters for multistep methods. *ACM Transactions on Mathematical Software (TOMS)*, 6(3):263–279, 1980.

[4] C William Gear. Algorithm 407: DIFSUB for solution of ordinary differential equations. *Communications of the ACM*, 14(3):185–190, 1971.

[5] L. R. Petzold and A. C. Hindmarsh. LSODAR. *Computing and Mathematics Research Division, I-316 Lawrence Livermore National Laboratory, Livermore, CA*, 94550, 1997.

[6] A. Ralston. Runge-Kutta methods with minimum error bounds. *Mathematics of computation*, 16(80):431–437, 1962.

[7] R. von Schwerin and H. G. Bock. A Runge–Kutta starter for a multistep method for differential-algebraic systems with discontinuous effects. *Applied Numerical Mathematics*, 18(1):337–350, 1995.

Paper III

SIAM Journal of Numerical Analysis, Reviewed and Re-submitted June 2018

A polynomial formulation of adaptive strong stability preserving multistep methods

Fatemeh Mohammadi Carmen Arévalo Claus Führer

Numerical Analysis, Center for Mathematical Sciences, Lund University,
Sölvegatan 18, SE-22100 Lund, Sweden

Abstract

A new formulation of explicit multistep methods allows variable step-sizes by construction. This formulation can be used to construct time-adaptive SSP multistep methods of any order for the solution of time dependent PDEs. The new formulation is implemented in a MATLAB package and some numerical examples are presented.

keywords: adaptive SSP multistep methods, variable step-sizes, strong stability preserving, TVD schemes

The authors are grateful to the anonymous colleagues who with great care produced critical and detailed reviews and useful suggestions that helped improve the quality of this work, and they also wish wish to thank G. Söderlind for his invaluable comments and advice.The second author wishes to thank D. Ketcheson for the invitation to King Abdullah University of Science and Technology and the fruitful discussions with his team at KAUST during this visit.

1 Introduction

Often the numerical solution of partial differential equations (PDEs) requires the use of ordinary differential equation (ODE) methods as part of the general technique, in order to solve systems of the form

$$u_t = Lu, \;\; u(t_0) = u_0, \tag{6.30}$$

where L is a difference operator that arises from the semi-discretization of PDEs.

In fact, some special methods for time dependent hyperbolic PDEs have been developed to ensure stability and avoid spurious oscillations of the numerical solutions during shocks. These are called *strong stability preserving* (SSP) methods, and are a class of ODE solvers that can be one of two types: one-step or multistep methods.

Several thorough studies on strong stability preserving methods and their discretizations have been published by Gottlieb, Shu, and Tadmor [7], Higueras [9], Gottlieb, Ketcheson, and Shu [6], and particularly on SSP multistep methods by Lenferink [12], and Ruuth and Hundsdorfer [15]. While SSP one-step methods have order barriers, SSP multistep methods can have arbitrarily high order. Nevertheless, we know of only one study on variable step-size SSP multistep methods, by Hadjimichael, Ketcheson, Lóczi, and Németh [8]. These authors developed adaptive explicit SSP multistep methods of orders two and three, pointing out that methods of order "at least 3 seem to have a complicated structure." They suggest looking for a new formulation with a simple structure that may be used for higher order explicit SSP multistep methods.

The aim of this paper is to develop a methodology that allows for given SSP multistep methods to be formulated as variable step-size methods. While we review existing theory for SSP methods in Section 2, we do not revisit the proofs for optimality made in [8], but rather use their results whenever possible, and present a general approach to making such methods adaptive. This approach is open to various step-size selection criteria, and can be combined with conventional error control using a tolerance parameter, or a greedy step-size selection scheme employing the maximal step-size that locally fulfills the SSP condition, as in [8].

The order p of explicit k-step SSP multistep methods with non-negative coefficients is less than the step number k. The most widely used variable step-size implementations of multistep methods are based on the Nordsieck representation [5], which does not have a straightforward extension to such lower order methods. Arévalo

and Söderlind [2] introduced a formulation that constructs adaptive k-step methods that are identified by a fixed set of parameters. These *parametric* multistep methods are described in Section 3. The computations are advanced by constructing and evaluating a polynomial at each step, where the step-size is chosen so as to control the local error. The main feature of this formulation is that the methods are intrinsically adaptive, and so require no extension to variable step-sizes. A complete technique was developed for explicit and implicit multistep methods of maximal order, i.e., for $p \geq k$. However, due to the specific order conditions associated with an SSP property for optimal methods with non-negative coefficients, i.e., that $p < k$ should be allowed, the technique is not directly applicable in the SSP context. In Section 4 we extend the original formulation to include explicit multistep methods with $p < k$. The new approach developed in this paper offers a straightforward construction of lower order explicit multistep methods, which are given a fully adaptive representation. Thus, each variable step-size k-step method of order p is completely identified by a set of $2k - p - 1$ fixed parameters, where k and p can be chosen freely. As the formulation of these type of methods is not unique, in Section 5 we design specific formulations that can describe adaptive versions of known fixed step-size SSP methods while maintaining the formulations as simple as possible.

In Section 6 we analyze formulations for optimal SSP methods. In general, these methods have several coefficients equal to zero. We propose a formulation that preserves each method's pattern of zero coefficients. These formulations are implemented in an adaptive ODE code, as explained in Section 7, and numerical examples are given in Section 8.

Because variable step-size methods have coefficients that vary from step to step as a function of the step-size ratios, it is important to study how the positivity of these coefficients is affected by each change of step-size. Although this topic is discussed for some particular examples, its general study is outside the scope of this paper. Instead, the focus here is on a methodology that permits an easy extension of explicit fixed step-size multistep formulas to variable step-size formulas. Such an extension has the property that it reverts to the original formula when the step-size is kept constant.

2 Linear multistep methods and strong stability preservation

We consider ODEs of the form

$$\dot{y} = F(y), \;\; y(t_0) = y_0, \;\; t \in [t_0, t_{\mathrm{f}}], \tag{6.31}$$

where $F : \mathcal{R}^m \to \mathcal{R}^m$. When an explicit linear multistep method with fixed step-size h is applied to this problem, a sequence of approximations $y_n \approx y(t_n)$ is computed from the difference equation

$$y_n = \sum_{i=1}^{k} (\alpha_i y_{n-i} + h\beta_i F(y_{n-i})), \tag{6.32}$$

where k is the step number, and α_i and β_i are the coefficients of the method. The method is chosen to suit the properties of the vector field F. In particular, strong stability preserving methods, also known as total variation diminishing (TVD) methods [16], or contractivity preserving methods [12], were developed for the time integration of semi-discretizations of hyperbolic conservation laws.

The setting for SSP methods is summarized as follows. Consider the autonomous system (6.31), where, for a given norm, the vector field F is assumed to have the property

$$\|y + hF(y)\| \leq \|y\| \;\text{ for all }\; y \;\text{ and }\; h \leq h^*. \tag{6.33}$$

This implies $F(0) = 0$, which obviously holds for linear vector fields, but also for many interesting nonlinear vector fields associated with PDEs. If F is linear we denote it by L, and (6.33) can be expressed in terms of the logarithmic norm, [4, 14, 17], as $\mu[L] \leq 0$, where

$$\mu[L] = \lim_{h \to 0^+} \frac{\|I + hL\| - 1}{h}. \tag{6.34}$$

The condition $\mu[L] \leq 0$ implies that $\|y(t)\|$ is a non-increasing function of t, and thus $y = 0$ is a stable solution. For nonlinear vector fields satisfying $F(0) = 0$, using the operator norm

$$\|F\| = \sup_{\|y\| \neq 0} \frac{\|F(y)\|}{\|y\|},$$

the corresponding logarithmic norm $\mu[F]$ can be readily defined as in (6.34); this is a straightforward special application of the general extension of the logarithmic norm to nonlinear maps, [17]. Thus (6.33) implies $\mu[F] \leq 0$, and once again $\|y(t)\|$ is a non-increasing function of t. Morover, if $\mu[F] < 0$, then $\|y + hF(y)\| \leq \|y\|$ in a nonempty interval, $0 < h < h^*$, since $\|y + hF(y)\|$ is a convex function of h.

The objective is to find methods that reproduce this behavior. A numerical method whose solution is non-increasing for all vector fields F satisfying (6.33) for step-sizes $0 < h \leq c\,h^*$ is said to be *strong stability preserving* (SSP). Obviously, the explicit Euler method is SSP with $c = 1$.

Whether the vector field F satisfies $\mu[F] < 0$ depends on the choice of norm, but the choice is not restricted to specific norms, such as inner product norms. A standard objective is to construct semi-discretizations of hyperbolic conservation laws so that the total variation of discrete solutions does not increase in time [16], where the *total variation* in space is defined as

$$\|u\|_{TV} = \sum_{j=1}^{m} |u_{j+1} - u_j|, \;\; u \in \mathcal{R}^m.$$

This is easily seen to be a semi-norm (referred to as the TV norm). If a semi-discretization satisfies $\mu[F] < 0$ with respect to the TV norm, and the time stepping method is SSP, the resulting method will overcome deficiencies that may occur in other methods, such as numerically induced oscillations in space, which are not present in the exact solutions of conservation laws. Thus the SSP method will produce a total variation diminishing (TVD) scheme for time steps $h \leq c\,h^*$.

To investigate linear multistep methods reducing total variation, further definitions are needed.

Definition 3 *[6] Let problem* (6.31) *satisfy condition* (6.33). *A k-step method given by formula* (6.32) *is an SSP method if there is a constant c such that the method applied to problem* (6.31) *with $0 < h \leq c\ h^*$ produces a sequence $\{y_i\}$ satisfying*

$$\|y_n\| \leq \max\{\|y_{n-1}\|, \|y_{n-2}\|, \ldots, \|y_{n-k}\|\}. \tag{6.35}$$

The maximal value of c is called the SSP constant of the method and is denoted by C.

Remark 2 *[6] Consider an explicit method defined by formula* (6.32) *with $\alpha_i \geq 0,\ \beta_i \geq 0$ for all i, and let $\gamma = \min_i \left\{ \dfrac{\alpha_i}{\beta_i} \;\middle|\; \beta_i \neq 0 \right\}$. If $\gamma > 0$, the method is SSP with $C = \gamma$.*

Although there are implicit multistep methods with larger SSP constants, the added cost of solving the implicit system of equations usually makes explicit methods computationally more efficient [6]. For example, the trapezoidal method, of order 2, has SSP constant $C = 2$, while the optimal explicit 3-step SSP method of order 2 has $C = 0.5$. Thus, this last method may require four times as many steps per unit time. However, the cost of solving the implicit system is often far beyond the cost of the extra steps.

It is known that for $k \geq 2$ there is no explicit k-step SSP method of order $p = k$ with all $\beta_i \geq 0$ [6]. Therefore, we will explore methods of order $p < k$. For brevity, we will refer to a k-step, order p method as a (k, p) method.

Remark 3 *[12] The SSP constant of an explicit* (k, p) *method with* $p > 1$, $\alpha_i \geq 0, \beta_i \geq 0$, *satisfies*

$$C \leq \frac{k-p}{k-1}.$$

Thus, increasing the number of steps allows for larger upper bounds on the SSP constant.

3 Parametric multistep methods

A new formulation for linear multistep methods where each k-step method of maximal order is represented by a fixed set of parameters was introduced by Arévalo and Söderlind [2]. This formulation supports variable step-sizes by construction, and has a simple structure, but it is valid only for maximal order, i.e., when $p \geq k$. Thus an extension to SSP methods is needed.

The parametric formula of a $k-$step method is defined as a linear combination of state values, y_{n-i}, where $i = 1, \ldots, k$, and their corresponding vector field values, $y'_{n-i} = F(y_{n-i})$. The vectors y_{n-i} represent the approximation of the solution $y(t)$ to the initial value problem (6.31) at a sample of times $t_{n-k}, \ldots, t_{n-1}$ with $h_{n-i} = t_{n+1-i} - t_{n-i}$. Let Π_p denote the space of polynomials of degree p. A k-step method in parametric form approximates the solution $y(t)$ for $t > t_{n-1}$ by constructing a polynomial $P_n \in \Pi_p$ and defining

$$y_n = P_n(t_n). \tag{6.36}$$

The formulation of parametric multistep methods makes use of the state and derivative *slacks*, defined as follows.

Definition 4 *[2] Let the sequences* $\{y_{n-i}\}_{i=0}^{k}$ *and* $\{y'_{n-i}\}_{i=0}^{k}$ *be given for a fixed* n*. Further, let* $P_n \in \Pi_p$ *with* $p \leq k+1$*. The state slack* s_{n-i} *and the derivative slack* s'_{n-i} *at* t_{n-i} *are defined by*

$$s_{n-i} = P_n(t_{n-i}) - y_{n-i}, \qquad s'_{n-i} = \dot{P}_n(t_{n-i}) - y'_{n-i}, \qquad i = 0, \ldots, k. \tag{6.37}$$

Further, in [2] it was demonstrated that every explicit k-step method of order p can be defined by $y_n = P_n(t_n)$, with the polynomial $P_n \in \Pi_k$ satisfying the conditions

$$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-i}\cos\theta_{i-1} + h_{n-i}s'_{n-i}\sin\theta_{i-1} = 0; \quad i = 2, \ldots, k, \end{cases} \tag{6.38}$$

where $\theta_i \in (-\frac{\pi}{2}, \frac{\pi}{2}]$ are the *method parameters*. The first two conditions are called *structural conditions* and specify the explicitness of the method. The additional linear combinations of state and derivative slacks are called *slack balance conditions*, and specify the particular method. Arévalo et al. [3] showed that the following parametric equivalence holds between the coefficients of a classical, constant step-size, multistep formula of maximal order and the method parameters:

$$\tan\theta_{i-1} = \frac{\beta_i}{\alpha_i} \text{ for } i = 2, \ldots, k. \tag{6.39}$$

Note that for a variable step-size method the parameters θ_{i-1} are constants, even though the coefficients α_i, β_i vary from step to step. Thus, with (6.38) a variable step-size k-step method is defined in terms of constants $\theta_1, \ldots, \theta_{k-1}$.

We need to consider a modified framework for SSP multistep methods. Our goal is to construct a general parametric formulation for explicit multistep methods that, while similar to (6.38), also covers SSP methods for which $p < k$. For this general formulation of explicit SSP multistep methods we will also prove an equivalence similar to (6.39), which will be seen to be essential to SSP methods.

4 Parametric formulations of explicit SSP multistep methods

Our approach in finding an adaptive formulation for SSP methods is based on obtaining a general formulation for all explicit k-step methods of a fixed order $p \leq k-1$.

As a method of order p is defined by polynomials of degree p, the $p+1$ coefficients of $P_n \in \Pi_p$ must be uniquely determined by $p+1$ interpolation conditions. For an explicit (k,p) SSP method we can construct these conditions as $p+1$ slack conditions. Consider each pair of coefficients (α_i, β_i) in (6.32) where either $\alpha_i = \beta_i = 0$ or $\alpha_i \neq 0$. Note that for SSP methods, if $\beta_i = 0$, the value of γ in Remark 2 is not influenced by the value of α_i. We call a pair (α_i, β_i) with $\alpha_i \neq 0$ a *non-zero pair*. It is these pairs that determine the value of γ.

For a pair of coefficients satisfying $(\alpha_i, \beta_i) = (0,0)$, the interpolation conditions that define P_n can not include s_{n-i} or s'_{n-i}, because these slacks contain y_{n-i} and y'_{n-i}. On the other hand, if $\alpha_i \neq 0$, the slack s_{n-i}, the only one that contains the term y_{n-i}, must be present in the interpolation conditions. The derivative slack s'_{n-i} may or may not be present, depending on the value of β_i. Thus, to construct the polynomial of a (k,p) SSP method we can select one of the following alternatives acting at each $t = t_{n-i}$, for each $i \in \{1,\dots,k\}$ corresponding to a non-zero pair:

$$\begin{aligned} &1. \quad && s_{n-i} + h_{n-i}\tau_i s'_{n-i} = 0 \\ &2. \quad && \begin{cases} s_{n-i} = 0 \\ s'_{n-i} = 0. \end{cases} \end{aligned} \tag{6.40}$$

The way in which these conditions are chosen for particular methods will be discussed in Sections 5 and 6.

Much of the literature on SSP methods has focused on finding the (k,p) method with largest SSP constant. Using standard optimization techniques, Lenferink [12] obtained all optimal explicit SSP multistep methods up to $k = 20$ steps and order $p = 7$. Ketcheson [11] constructed an algorithm for computing the coefficients of optimal explicit SSP methods given the number of steps and the order required, and presented a table of optimal SSP constants up to $k = 50$ steps and order $p = 15$.

Table 6.10 shows the SSP constants of optimal explicit SSP multistep methods up to $k = 7$ steps and order $p = 5$. Note that the optimal SSP constant for a fixed order

k \ p	1	2	3	4	5
1	1.0				
2	-				
3	-	0.5			
4	-	0.667	0.333		
5	-	0.75	0.5	0.021	
6	-	0.8	0.583	0.165	
7	-	0.833	-	0.282	0.038

Table 6.10: SSP constants of optimal explicit multistep methods up to $k = 7$ steps and order $p = 5$. No optimal methods of order 1 can be found for $k > 1$, nor of order $p = 3$ for $k \geq 7$.

increases as the number of steps increases, so it can pay off to construct methods of a larger number of steps without increasing the order of accuracy, as long as the number of non-zero coefficients of the method remains low. This is of particular interest if the methods are adaptive and the pattern of zero coefficients of the fixed step-size formula is to be preserved in its variable step-size extension.

Remark 4 *[12] There is no optimal order 1 SSP method for $k > 1$. The supremum of the SSP constants for explicit $(k, 1)$ SSP methods with $k > 1$ is obtained when $\alpha_i = \beta_i = 0$ for $i = 2, \ldots, k$, but in that case the method reduces to the 1-step method. Also, there is no optimal $(k, 3)$ SSP method for $k > 6$, and in fact, the same phenomenon occurs after some value of k for all odd values of p.*

Optimal SSP multistep methods were originally defined for constant step-sizes. Here we adopt the definition introduced by Hadjimichael et al. [8] to cover adaptive multistep methods with formula

$$y_n = \sum_{i=1}^{k} (\alpha_{i,n} y_{n-i} + h_{n-1} \beta_{i,n} F(y_{n-i})). \tag{6.41}$$

The step ratios are defined as $\Omega_j = \frac{1}{h_{n-1}} \sum_{i=0}^{j-1} h_{n-k+i}$ and the SSP coefficient at step n is $C_n = \gamma_n$, if $\gamma_n = \min_i \left\{ \frac{\alpha_{i,n}}{\beta_{i,n}} \;\middle|\; \beta_{i,n} \neq 0 \right\} > 0$. The SSP condition then reads

$$0 \leq h_{n-1} \leq C_n \delta_n \tag{6.42}$$

where

$$\delta_n := \min_{0 \leq j \leq k-1} h^*(y_{n-k+j}). \tag{6.43}$$

Structure	Order: $p = 2$
$\mathcal{A}$	$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-k} + h_{n-k}\ \tau_k\ s'_{n-k} = 0 \end{cases}$
$\mathcal{B}$	$\begin{cases} s_{n-1} + h_{n-1}\ \tau_1\ s'_{n-1} = 0 \\ s_{n-k} = 0 \\ s'_{n-k} = 0 \end{cases}$
$\mathcal{C}$	$\begin{cases} s_{n-1} + h_{n-1}\ \tau_1\ s'_{n-1} = 0 \\ s_{n-i} + h_{n-i}\ \tau_i\ s'_{n-i} = 0 \\ s_{n-k} + h_{n-k}\ \tau_k\ s'_{n-k} = 0 \end{cases}$ $i \in \{2, 3, \ldots, k-1\}$

Table 6.11: Possible sets of slack conditions to construct optimal k-step SSP methods of order 2.

When the step-size remains constant, it is clear that $C_n = C$, and in fact, if the step-sizes are slowly varying, the variable coefficients are slowly varying too and C_n will remain close to C as long as the patterns of non-zero coefficients of the variable step-size formula and the constant step-size formula are the same.

The following is an extension of Remark 3 for the variable step-size case.

Remark 5 *[8] The SSP coefficient of an explicit* (k, p) *method with* $p > 1$, $\alpha_i \geq 0, \beta_i \geq 0$, *requires that* $\Omega_k > p$ *and satisfies*

$$C_n \leq \frac{\Omega_k - p}{\Omega_k - 1}.$$

There are many different possible sets of slack conditions that may define SSP methods of various orders. In the example below we look for a parametric formulation of optimal explicit $k-$step SSP methods of order 2 with $k \geq 3$. We restrict our choices by only considering formulations that include s_{n-1} and s'_{n-1}, that is, the last computed values.

Example 1. The parametric formulation of an explicit $(k, 2)$ method in the interval $[t_{n-1}, t_n]$ is given by a second degree polynomial. In order to find the corresponding polynomial coefficients three equations in terms of slack conditions are needed. To render a k-step method, slack conditions for $t = t_{n-k}$ must be part of the formulation. There are many different possible sets of slack conditions that may define k-step methods of order 2.

Table 6.11 shows some possible different formulations for an optimal $(k, 2)$ SSP method categorized in one of three different structures. The first two only include slack conditions for the points at $t = t_{n-k}$ and $t = t_{n-1}$, and the third one also includes one of the intermediate points. Here we will analyze what type of method is described by Structure $\mathcal{B}$. The polynomial of the optimal method must be of the form

$$P(t) = A\frac{(t-t_n)^2}{h_{n-1}^2} + y'_{n-1}\frac{t-t_n}{h_{n-1}} + y_{n-1} \tag{6.44}$$

and satisfy the conditions given in Table 6.11. Using this together with (6.36), we get that the non-zero coefficients of the multistep formula are

$$\alpha_{1,n} = \frac{\Omega_k^2}{\Omega_k^2 + 2(\tau_1-1)\Omega_k - 2\tau_1 + 1} \tag{6.45}$$

$$\alpha_{k,n} = \frac{2(\tau_1-1)\Omega_k - 2\tau_1 + 1}{\Omega_k^2 + 2(\tau_1-1)\Omega_k - 2\tau_1 + 1} \tag{6.46}$$

$$\beta_{1,n} = \frac{\tau_1\Omega_k^2}{\Omega_k^2 + 2(\tau_1-1)\Omega_k - 2\tau_1 + 1} \tag{6.47}$$

$$\beta_{k,n} = \frac{\Omega_k((\tau_1-1)\Omega_k - 2\tau_1 + 1)}{\Omega_k^2 + 2(\tau_1-1)\Omega_k - 2\tau_1 + 1}. \tag{6.48}$$

The SSP coefficient is then given by

$$\gamma_n(k) = \min\left\{\frac{1}{\tau_1}, \frac{2(\tau_1-1)\Omega_k - 2\tau_1 + 1}{\Omega_k^2 + 2(\tau_1-1)\Omega_k - 2\tau_1 + 1}\right\} \tag{6.49}$$

as long as the coefficients remain non-negative. As the parameter τ_1 is independent of the step number n, we can consider the constant step-size case, i.e., when $\Omega_j = j$. Then

$$\gamma(k) = \min\left\{\frac{1}{\tau_1}, \frac{2(\tau_1-1)k - 2\tau_1 + 1}{k((\tau_1-1)k - 2\tau_1 + 1)}\right\}, \tag{6.50}$$

and bearing in mind the positivity of the coefficients, we get that for each fixed value of k the value of τ_1 that maximizes $\gamma > 0$ is $\tau_1 = (k-1)/(k-2)$ and the constant of the optimal SSP k-step method of order 2 is

$$C(k) = \frac{k-2}{k-1}. \tag{6.51}$$

Nevertheless, we observe that the coefficient $\beta_{k,n}$, which agrees with the constant step-size coefficient $\beta_k = 0$ when the step-size is kept constant, does not remain

zero in the variable step-size extension, and even becomes negative if $\Omega_k < k$. The implication is that the method would not be SSP if, after $k-1$ constant step-sizes, the step-size is allowed to increase. Thus, this formulation is ill-fitted for SSP methods. □

In the next section we derive a formulation for explicit (k,p) methods with $p<k$ where the pattern of zero coefficients of the fixed step-size formula is retained in the variable step-size extension.

5 A formulation for explicit multistep methods of lower orders

We have suggested possible parametric formulations for explicit $(k,2)$ SSP methods using different combinations of slack conditions. In order to formulate a general explicit (k,p) method with $p<k$, we need $p+1$ interpolation conditions. There must be at least one slack present for each non-zero pair (α_i,β_i), but having one slack condition for each point at $t_{n-i}, i=1,\dots,k$, might result in an overdetermined system. Therefore, we consider using linear combinations of the slack conditions. In the classical formulation of explicit multistep methods, a $k-$step, order p method is defined by its coefficients $\alpha_1,\dots,\alpha_k$ (α_0 is normalized to 1), and $\beta_1,\dots,\beta_k$, and $p+1$ order conditions. We are looking for a parametric formulation of (k,p) methods that includes methods that can be obtained by fixing $2k-p-1$ parameters, so that there is a one-to-one correspondence between the classical coefficients of a method and its parameters in the polynomial formulation. Here we do not assume that the methods are SSP.

We add one more alternative to the set of conditions given in (6.40), combining one or more balance slacks with the condition at the farthest away point, $t=t_{n-k}$. The slack s_{n-k} must be present in the formulation to guarantee the prescribed number of steps. Thus, to p interpolation conditions for $i\in\{1,\dots,k-1\}$ chosen from (6.40), add one condition,

$$\sum\left(\lambda_i s_{n-i}+h_{n-i}\tau_i s'_{n-i}\right)+s_{n-k}+h_{n-k}\tau_k s'_{n-k}=0, \tag{6.52}$$

so that each slack s_{n-i}, s'_{n-i}, appears at most once in the formulation. The constants λ_i and τ_i in (6.40) and (6.52) are the *method parameters* of this formulation, where the constants τ_i correspond to $\tan\theta_i$ in (6.38). As these parameters do not

depend on the step-sizes, we can use a fixed step-size formulation of the method to calculate their values. The following theorem gives the formulas that allow us to calculate these parameters for a set of given method coefficients.

To extend a fixed step-size method with given coefficients $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k$, in a way to preserve the pattern of zero coefficients, we observe the following rules:

Procedure 1: Formulation of explicit methods of order $p < k$

To define the method polynomial $P_n \in \Pi_p$, use the following rules to set up the interpolation conditions:

- If $(\alpha_i, \beta_i) = (0, 0)$, do not include either s_{n-i} or s'_{n-i}.
- If $\alpha_i = 0, \beta_i \neq 0$, include only $s'_{n-i} = 0$.
- For each $\alpha_i \neq 0$ and $\beta_i \neq 0$, include one of the following:

$$\begin{cases} s_{n-i} = 0 \\ s'_{n-i} = 0, \end{cases} \tag{6.53}$$

$$\text{or } ; s_{n-i} + h_{n-i}\tau_i s'_{n-i} = 0, \tag{6.54}$$

$$\text{or } ; \sum \left(\lambda_i s_{n-i} + h_{n-i}\tau_i s'_{n-i}\right) + s_{n-k} + h_{n-k}\tau_k s'_{n-k} = 0 \tag{6.55}$$

so that the total number of conditions adds up to $p + 1$.

There is some freedom in choosing the interpolation conditions, but the following theorem gives formulas for the method coefficients once the choice has been made.

Theorem 3 *For a (k, p) method defined by $p + 1$ slack conditions chosen according to Procedure 1, the method parameters can be defined as follows:*

$$\begin{aligned} \tau_i &= \frac{\beta_i}{\alpha_i}, && \textit{when (6.54) is used} \\ \tau_i &= \frac{\beta_i}{\alpha_k}, \quad \lambda_i = \frac{\alpha_i}{\alpha_k}, && \textit{when (6.55) is used} \end{aligned} \tag{6.56}$$

Proof 1 *Consider the constant step-size formulation of a method defined as prescribed in Procedure 1.*

As the method is of order p, and P_n is of degree p, formula (6.32) *is satisfied exactly when y_{n-i} and y'_{n-i} are replaced by $P_n(t_{n-i})$ and $\dot{P}_n(t_{n-i})$ respectively. Inserting the polynomial into the formula we obtain*

$$P_n(t_n) = \sum_{i=1}^{k} (\alpha_i P_n(t_{n-i}) + h\beta_i \dot{P}_n(t_{n-i})), \tag{6.57}$$

and subtracting (6.57) *from* (6.32) *we get*

$$P_n(t_n) - y_n = \sum \left(\alpha_i s_{n-i} + h\beta_i s'_{n-i}\right) + \sum \left(\alpha_i s_{n-i} + h\beta_i s'_{n-i}\right) + \alpha_k s_{n-k} + h\beta_k s'_{n-k}, \tag{6.58}$$

where the first sum corresponds to the indexes involved in (6.54) and the second sum to those involved in (6.55). From (6.54) we get that each term in the first sum satisfies

$$s_{n-i} + h\frac{\beta_i}{\alpha_i} s'_{n-i} = 0 \tag{6.59}$$

and from (6.55) we have that

$$\sum \left(\frac{\alpha_i}{\alpha_k} s_{n-i} + h\frac{\beta_i}{\alpha_k} s'_{n-i} \right) + s_{n-k} + h\frac{\beta_k}{\alpha_k} s'_{n-k} = 0. \tag{6.60}$$

Thus we conclude that $y_n = P_n(t_n)$.

Theorem 3 states the relation between the set of method coefficients in the classical formulation, α_i and β_i, and the parameters in the polynomial formulation, τ_i and λ_j. Thus, given the coefficients in formula (6.32), one can obtain the method parameters from Theorem 3. It is important to note that the method parameters do not vary with the step-sizes, so that this relation, obtained for the constant coefficients of a fixed step-size method, gives the method parameters for the equivalent variable step-size method in the new formulation.

Example 2. To construct the variable step-size extension of the 6-step, order 3 formula with coefficients $\alpha_1 = 1/4, \alpha_2 = 0, \alpha_3 = 1/2, \alpha_4 = 1/8, \alpha_5 = 1/8, \beta_1 = 1/16, \beta_2 = 565/96, \beta_3 = -253/48, \beta_4 = 199/96, \beta_5 = 1/8$, we set up a system of four equations to calculate the coefficients of the method polynomial $P_n \in \Pi_3$ according to Procedure 1.

$$\begin{aligned}
&s_{n-1} + \tau_1 h s'_{n-1} = 0 \\
&s'_{n-2} = 0 \\
&s_{n-3} + \tau_3 h(\Omega_3 - \Omega_2) s'_{n-3} = 0 \\
&\lambda_4 s_{n-4} + \tau_4 h(\Omega_2 - \Omega_1) s'_{n-4} + s_{n-5} + \tau_5 h \Omega_1 s'_{n-5} = 0
\end{aligned}$$

From Theorem 3 we have that $\tau_1 = 1/4, \tau_3 = -253/24, \tau_4 = 199/12, \tau_5 = 1, \lambda_4 = 1$. The variable step-size method will advance the solution by setting $y_n = P_n(t_n)$.

In particular, for SSP methods, for which $\alpha_i = 0 \Rightarrow \beta_i = 0$, we have a similar procedure for constructing the interpolation conditions that define the method polynomial. Note that by construction the pattern of zero coefficients of the fixed step-size formula is preserved by the variable step-size extension. It is also clear that if the constant coefficients are positive, the variable step-size coefficients will remain positive for some change in the step-sizes, which in some cases may have to be quite small. This would require limiting the allowed step-size ratios in the method implementation, but as will be seen in the next section, precise bounds of the step-size ratios may be difficult to calculate.

Procedure 2: Formulation of explicit formulas for SSP methods

- If $(\alpha_i, \beta_i) = (0, 0)$, do not include either s_{n-i} or s'_{n-i}.
- If $\alpha_i \neq 0, \beta_i = 0$, include only $s_{n-i} = 0$.
- For each $\alpha_i \neq 0$ and $\beta_i \neq 0$, include one of the following:

$$\begin{cases} s_{n-i} = 0 \\ s'_{n-i} = 0, \end{cases} \tag{6.61}$$

$$\text{or } ; s_{n-i} + h_{n-i}\tau_i s'_{n-i} = 0, \tag{6.62}$$

$$\text{or } ; \sum \left(\lambda_i s_{n-i} + h_{n-i}\tau_i s'_{n-i}\right) + s_{n-k} + h_{n-k}\tau_k s'_{n-k} = 0 \tag{6.63}$$

so that the total number of conditions adds up to $p + 1$.

Corollary 1 *The method parameters of a method constructed using Procedure 2 are*

$$\hat{\tau}_i = \frac{\beta_i}{\alpha_i}, \quad \textit{when (6.62) is used}$$

$$\tau_i = \frac{\beta_i}{\alpha_k}, \quad \lambda_i = \frac{\alpha_i}{\alpha_k}, \quad \textit{when (6.63) is used.}$$

Let $\gamma = \min_i \left\{ \frac{1}{\hat{\tau}_i}, \frac{\lambda_i}{\tau_i} \;\middle|\; \hat{\tau}_i \neq 0, \tau_i \neq 0 \right\}$. *If* $\gamma > 0$, *the SSP constant is* $C = \gamma$.

Proof 2 *The proof follows from Remark 2. Note that* $\alpha_k \neq 0$ *because for* $\alpha_k = 0$ *either the method is not a* $k-$*step method (if* $\beta_k = 0$*), or the value of* γ *is zero (if* $\beta_k \neq 0$*).*

6 Alternative formulations of optimal SSP methods

The method parameters of an optimal $k-$step SSP formula of order p, represented here by SSPkp, can be calculated using Theorem 3, thus effectively converting a fixed step-size method to variable step-size. Its coefficients, $\alpha_i, \beta_i,\ i = 1, \ldots, k$, can be obtained using Ketcheson's algorithm [11]. Alternatively, given the method parameters λ_j and τ_j obtained with Procedure 2, the coefficients of an SSPkp method, α_i and β_i, can be determined by solving the system consisting of the parametric relations in Corollary 1 together with the $p+1$ order conditions. Note that λ_j and τ_j depend only on the fixed step-size method coefficients α_i, β_i. It is clear that the variable step-size formulas are not SSP methods without the appropriate restriction to the step-size ratios. In this paper the emphasis is in the construction of the adaptive formulas, but further attention must be given to the step-size ratio restrictions and the SSP coefficients.

Example 3. Consider the optimal SSP32 method with constant coefficients $\alpha_1 = \frac{3}{4}, \alpha_2 = 0, \alpha_3 = \frac{1}{4}, \beta_1 = \frac{3}{2}$ and $\beta_2 = \beta_3 = 0$. This method has $\tau_1 = 2,\ \tau_2 = 0,\ \tau_3 = 0$, and in this case there is no parameter λ, as $k = p + 1$.

We can express the adaptive method by a variable step-size formula

$$y_n = \sum_{i=1}^{3} \left(\alpha_{i,n}(\Omega_1, \Omega_2) y_{n-i} + h_{n-1} \beta_{i,n}(\Omega_1, \Omega_2) y'_{n-i} \right). \tag{6.64}$$

If we use Procedure 2 to construct the optimal SSP32 method, the equations that must be solved to advance the solution are

$$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-3} = 0, \end{cases} \tag{6.65}$$

and solving this system and evaluating the method polynomial at $t = t_n$ yields

$$\begin{aligned}
\alpha_{1,n}(\Omega_1, \Omega_2) &= \frac{\Omega_2^2 - 1}{\Omega_2^2}, \\
\alpha_{2,n}(\Omega_1, \Omega_2) &= 0, \\
\alpha_{3,n}(\Omega_1, \Omega_2) &= \frac{1}{\Omega_2^2}, \\
\beta_{1,n}(\Omega_1, \Omega_2) &= \frac{\Omega_2 + 1}{\Omega_2^2}, \\
\beta_{2,n}(\Omega_1, \Omega_2) &= 0, \\
\beta_{3,n}(\Omega_1, \Omega_2) &= 0.
\end{aligned}$$

When all step-sizes are set equal, then $\Omega_1 = 1$ and $\Omega_2 = 2$, and the coefficients coincide with the fixed step-size SSP32 method. Note that the pattern of zero coefficients is preserved. Also, as long as $\Omega_2 > 1$, the variable coefficients remain positive. This result coincides with Hadjimichael et al. [8]. □

Consider the pairs of constant step-size coefficients (α_i, β_i) of an explicit SSP method given by (6.32). As was discussed previously, each non-zero pair must have $\alpha_i \neq 0$. As we could determine from Table 3 in [11], optimal SSP methods with $(k, p) \neq (6, 3), k \leq 50$, satisfy the following conditions:

1. $\alpha_1 \neq 0$, $\beta_1 \neq 0$, $\alpha_k \neq 0$.
2. If p is even, $\beta_k = 0$ and there are p pairs of non-zero coefficients.
3. If p is odd, $\beta_k \neq 0$ and there are $p - 1$ pairs of non-zero coefficients.

The single exception to this rule, the optimal $(6, 3)$ method, has p pairs of non-zero coefficients. An optimal method can be constructed by choosing $p + 1$ slack conditions that depend only on the points corresponding to non-zero coefficients, to preserve the pattern of zeros of the constant step-size method. This simplified strategy is described in Procedure 3.

Example 4. The optimal explicit $(k, 3)$ SSP methods with $k = 4, 5$ are constructed as

$$\begin{cases}
s_{n-1} = 0, \\
s'_{n-1} = 0, \\
s_{n-k} = 0 \\
s'_{n-k} = 0.
\end{cases} \tag{6.66}$$

Procedure 3: Formulation of optimal SSP methods

- *Take structural conditions at the point* $t = t_{n-1}$.
- *Take slack balance conditions at the intermediate points* $t = t_{n-j}$ *with* $\alpha_j \neq 0$, $1 < j < k$. *The method parameters are* $\tau_j = \beta_j/\alpha_j$.
- *Add the state slack* $s_{n-k} = 0$. *If* p *is odd, also add the derivative slack* $s'_{n-k} = 0$. *(For the* $(6, 3)$ *method, take a slack balance condition at* t_{n-6}.*)*

The non-zero variable coefficients of the multistep formula can be derived from (6.66) as

$$\alpha_{1,n} = \frac{(\Omega_k - 3)\Omega_k^2}{(\Omega_k - 1)^3} \tag{6.67}$$

$$\alpha_{k,n} = \frac{3\Omega_k - 1}{(\Omega_k - 1)^3} \tag{6.68}$$

$$\beta_{1,n} = \frac{\Omega_k^2}{(\Omega_k - 1)^2} \tag{6.69}$$

$$\beta_{k,n} = \frac{\Omega_k}{(\Omega_k - 1)^2}. \tag{6.70}$$

The SSP coefficient is given by

$$\gamma_k = \min\left\{\frac{\Omega_k - 3}{\Omega_k - 1}, \frac{3\Omega_k - 1}{(\Omega_k - 1)\Omega_k}\right\} \tag{6.71}$$

when the coefficients are positive and $\gamma_k > 0$, that is, when $\Omega_k > 3$. It can be easily shown that $\gamma_k = \frac{\Omega_k - 3}{\Omega_k - 1}$ if $3 < \Omega_k \leq 5.828$ while $\gamma_k = \frac{3\Omega_k - 1}{(\Omega_k - 1)\Omega_k}$ if $\Omega_k \geq 5.828$, but from Remark 5 we know that $C_n \leq \dfrac{\Omega_k - 3}{\Omega_k - 1}$, so the method is not optimal if $\Omega_k \geq 5.828$. These results agree with those of Hadjimichael et al. [8]. □

Methods with higher order and a larger number of steps depend, in general, on several step-size ratios, their variable step-size coefficients are high degree rational functions of these step-size ratios, and are therefore difficult to study in terms of their SSP coefficients.

Example 5. The optimal explicit $(8, 5)$ SSP method has nonzero coefficients $\alpha_1 = 1360/4363$, $\alpha_4 = 233/2112$, $\alpha_5 = 2323/10831$, $\alpha_8 = 896/2465$, $\beta_1 = 275/128$,

$\beta_4 = 1044/1373$, $\beta_5 = 6661/4506$ and $\beta_8 = 1781/5144$. Thus, we construct the method as

$$\begin{cases} s_{n-1} = 0, \\ s'_{n-1} = 0, \\ s_{n-4} + h_{n-4}\tau_4 s'_{n-4} = 0, \\ s_{n-5} + h_{n-5}\tau_5 s'_{n-5} = 0, \\ s_{n-8} = 0, \\ s'_{n-8} = 0. \end{cases} \tag{6.72}$$

For this method we calculate $\tau_4 = \beta_4/\alpha_4 = 2433/353$, $\tau_5 = \beta_5/\alpha_5 = 2433/353$. The variable-step-size formula preserves the pattern of zeros, and the positivity of the coefficients is also preserved as long as the step-sizes vary smoothly. Exact calculations of the bounds on the step-size ratios to secure positivity are difficult to perform. Equally difficult is the calculation of the SSP coefficient, but as it varies continuously with the step-size ratios, a slowly varying step-size sequence will produce an SSP coefficient that is close to the constant step-size SSP constant for the method, $C = 353/2433$. Although sharp bounds are hard to obtain, we found that positivity of the variable step-size coefficients for this method can be ensured for the extreme case when step-sizes are assumed to increase at a constant rate of 3.5%, or decrease at the constant rate of 5.5%. □

In the implementation of these methods, the variable coefficients of the adaptive method are not explicitly calculated. To advance the solution at each step, (6.72) is solved for the method polynomial and the new solution is obtained by evaluating the polynomial at $t = t_n$. A new step-size h_{n-1} is selected at each step. This can be done in different ways. To use the greedy step-size selection of Hadjimichael et al. [8] a complicated stability analysis of the acceptable step-sizes is needed. As that approach means that each particular method must be analyzed and that these calculations can be difficult, and because the error cannot be monitored, we take an alternative approach. Given an error tolerance, we use controllers based on digital filter theory [1], to adapt the step-size to the error tolerance. By a continuity argument, the method coefficients remain positive when the step-size varies slowly.

7 Implementation

We implemented our formulation in the adaptive multistep solver MODES [3]. This ODE Matlab toolbox, including the SSP module, is publicly available for download [1]. The original package contains the explicit $p = k$ methods and the implicit $p = k$ and $p = k + 1$ methods. The user can choose fixed or variable step-sizes, and has a choice of several step-size controllers designed both for general and for more specific needs. For each class of methods with k steps and order p there is a function that computes the coefficients of the method polynomial of degree p by solving the derived parametric formulation for that class. For instance, a $2-$step explicit multistep method of order 2 is characterized by a second degree polynomial P_n,

$$P_n(t) = c_2(t - t_{n-1})^2 + c_1(t - t_{n-1}) + c_0,$$

satisfying the slack conditions in (6.38),

$$\begin{cases} s_{n-1} = 0 \\ s'_{n-1} = 0 \\ s_{n-2}\cos\theta_1 + h_{n-2}s'_{n-2}\sin\theta_1 = 0 \end{cases} \tag{6.73}$$

where these three conditions, together with the parameter value θ_1, uniquely determine the polynomial coefficients. Then the solution at time t_n is obtained as $P_n(t_n) = c_2h_{n-1}^2 + c_1h_{n-1} + c_0$.

The implementation in MODES was made by adding a function `polElow` which, given the method parameters λ and τ, computes the solution at t_n by solving system for the coefficients of P_n, and then evaluates $y_n = P_n(t_n)$. We also have the option of calling some optimal (k, p) SSP methods for $p \leq 5$ by name, without giving their parameters explicitly. The step-size controllers in MODES provide an estimate of the local error at each step. By monitoring the error estimation, the controllers increase or decrease the step-size when the error estimate is below or above the specified tolerance, and in particular they reduce the step-size when a numerical instability is detected. This is particularly important for explicit SSP integrators that cannot operate with step-sizes above their stability limit. Using an adaptive implementation of these methods eliminates the need of calculating the SSP constant for each particular method, although care must be taken to set appropriate bounds for step-size changes. These bounds will be conservative and the method will be costlier than when the greedy step-size selection is used, but on the other hand, the error will be monitored, keeping it below a given tolerance.

Our implementation retains MODES's step-size controllers and its mechanism for calculating the starting values of the multistep methods, which are provided by Runge-Kutta methods. As MODES allows the user to set upper and lower limits on the step-size ratios, it is a particularly suited platform for maintaining these ratios bounded, as required by SSP methods.

8 Numerical results

In this section we investigate the performance of the parametric SSP methods as implemented in MODES.

We consider the inviscid Burgers' equation with periodic boundary conditions

$$\begin{aligned} &u_t + uu_x = 0, \\ &u(x,0) = g(x), \quad x \in [0,1], \end{aligned} \tag{6.74}$$

using the smooth initial function

$$g(x) = \frac{1}{2} + \sin(2\pi x). \tag{6.75}$$

Furthermore, the model is semi-discretized with the fifth order Weighted Essentially Non Oscillatory (WENO) scheme [13, 10]. WENO is one of the spatial discretizations that are often combined with an SSP time integrator to preserve contractivity properties.

The step-size controllers in MODES are designed to obtain a smooth step-size sequence. The small change in step-sizes is crucial to guarantee the positivity of time-dependent SSP multistep coefficients. However, the implementation of an efficient controller that keeps the step-sizes nearby the SSP coefficient is out of the scope of this paper.

The problem was solved using the optimal (8,5) SSP formula described in Section 6. Figure (6.18) shows (a) the ratio $\frac{h}{\Delta x}$ and (b) the step-size ratio, $\frac{h_{n-1}}{h_{n-2}}$, versus the simulation time. The implemented error controllers in MODES keep the step-sizes below the SSP bound (dashed line) during the shock formation and so we observed no oscillations in the solution. Furthermore, the step-size ratios, $\frac{h_{n-1}}{h_{n-2}}$, remain close to the vicinity of 1. This indicates that step-size change is small at every step. However a noticeable change in the step-size ratio in Figure (6.18(b)) occurs

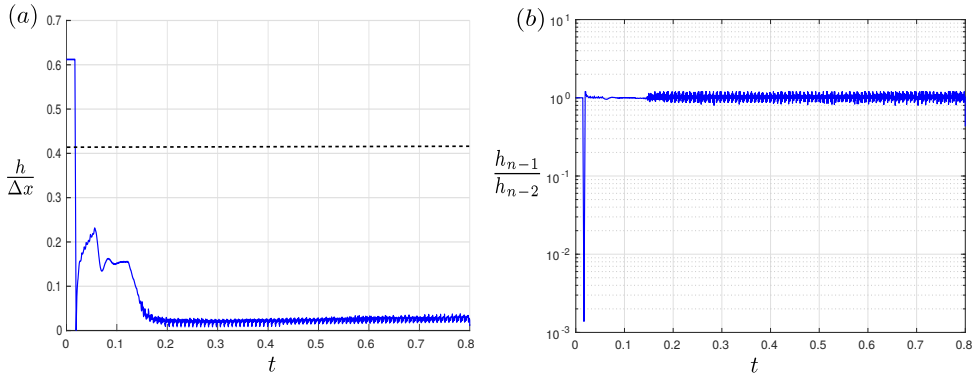

Figure 6.18: The adaptive optimal (8,5) SSP method as described in Example 5 was used to solve Burgers' equation with an error tolerance of 10^{-6}, $0.8h_{n-2} < h_{n-1} < 1.2h_{n-2}$ and 256 spatial discretization points. The plot shows the ratio $\frac{h}{\Delta x}$ for the optimal $(8, 5)$ SSP method, and (b) the step-size ratios $\frac{h_{n-1}}{h_{n-2}}$.

when the step-size is controlled immediately after initialization. This suggests that the choice of initial step-size can be refined.

9 Conclusion

The parametric formulation presented in this paper gives a simple structure to explicit SSP multistep methods of higher orders. With an addendum to the multistep ODE solver MODES, we have implemented adaptive explicit multistep methods of any order and any number of steps. For SSP methods, the available step-size controllers in MODES keep the step-sizes under the stability limit, in particular during the shock formation. We have proved an equivalence relation between the coefficients of the classical method formulas and the parameters that define an explicit k-step, order p method, with $p < k$.

Although we have observed that with the available controllers SSP methods take steps within the stability bound, it is best to keep the ratio $\frac{h_n}{\Delta x}$ near or at the step-size limit $C_n h^*$, in order to take larger step-sizes. It would be useful to construct a specific controller that keeps the step-sizes as large as possible while requiring the method to satisfy the non-increasing condition.

The methodology employed in this paper is also suited to the development of adaptive implicit SSP methods. It is possible that adaptivity makes up for some of the additional computations required to solve the nonlinear systems.

Bibliography

[1] Carmen Arevalo, Erik Jonsson-Glans, Josefine Olander, Monica Selva-Soto, and Gustaf Söderlind. MODES, 2016.

[2] Carmen Arévalo and Gustaf Söderlind. Grid-independent construction of multistep methods. *Journal of Computational Mathematics*, 35(5):670–690, 2017.

[3] Carmen Arévalo, Gustaf Söderlind, Erik Jonsson-Glans, Josefine Olander, and Monica Selva-Soto. MODES: A software platform for adaptive high order multistep methods. *Lund University, Preprints in Mathematical Sciences*, 2017:1.

[4] Germund Dahlquist. *Stability and error bounds in the numerical integration of ordinary differential equations*. PhD thesis, Almqvist & Wiksell, 1958.

[5] Peter Deuflhard and Folkmar Bornemann. Scientific computing with ordinary differential equations. *Springer Texts in Applied Mathematics*, 2000.

[6] Sigal Gottlieb, David I Ketcheson, and Chi-Wang Shu. *Strong stability preserving Runge-Kutta and multistep time discretizations*. World Scientific, 2011.

[7] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43(1):89–112, 2001.

[8] Yiannis Hadjimichael, David I Ketcheson, Lajos Lóczi, and Adrián Németh. Strong stability preserving explicit linear multistep methods with variable step size. *SIAM Journal on Numerical Analysis*, 54(5):2799–2832, 2016.

[9] Inmaculada Higueras. On strong stability preserving time discretization methods. *Journal of Scientific Computing*, 21(2):193–223, 2004.

[10] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996.

[11] David Ketcheson. Computation of optimal monotonicity preserving general linear methods. *Mathematics of Computation*, 78(267):1497–1513, 2009.

[12] Hermanus WJ Lenferink. Contractivity preserving explicit linear multistep methods. *Numerische Mathematik*, 55(2):213–223, 1989.

[13] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(1):200–212, 1994.

[14] Sergei Mikhailovich Lozinskii. Error estimate for numerical integration of ordinary differential equations. i. *Izvestiya Vysshikh Uchebnykh Zavedenii. Matematika*, (5):52–90, 1958.

[15] Steven J Ruuth and Willem Hundsdorfer. High-order linear multistep methods with general monotonicity and boundedness properties. *Journal of Computational Physics*, 209(1):226–248, 2005.

[16] Chi-Wang Shu and Stanley Osher. Efficient implementation os essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77:439–471, 1988.

[17] Gustaf Söderlind. The logarithmic norm. history and modern theory. *BIT Numerical Mathematics*, 46(3):631–652, 2006.