Efficient Time Stepping Methods and Sensitivity Analysis for Large Scale Systems of Differential Equations

Hong Zhang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science Application

Adrian Sandu, Chair
Yang Cao
Traian Iliescu
Tao Lin
Calvin J. Ribbens
Raymond Spiteri

August 11, 2014
Blacksburg, Virginia

Keywords: Time Stepping, General Linear Methods, Implicit-explicit, Sensitivity Analysis

Efficient Time Stepping Methods and Sensitivity Analysis for Large Scale Systems of Differential Equations

Hong Zhang

(ABSTRACT)

Many fields in science and engineering require large-scale numerical simulations of complex systems described by differential equations. These systems are typically multi-physics (they are driven by multiple interacting physical processes) and multiscale (the dynamics takes place on vastly different spatial and temporal scales). Numerical solution of such systems is highly challenging due to the dimension of the resulting discrete problem, and to the complexity that comes from incorporating multiple interacting components with different characteristics. The main contributions of this dissertation are the creation of new families of time integration methods for multiscale and multiphysics simulations, and the development of industrial-strengh tools for sensitivity analysis.

This work develops novel implicit-explicit (IMEX) general linear time integration methods for multiphysics and multiscale simulations typically involving both stiff and non-stiff components. In an IMEX approach, one uses an implicit scheme for the stiff components and an explicit scheme for the non-stiff components such that the combined method has the desired stability and accuracy properties. Practical schemes with favorable properties, such as maximized stability, high efficiency, and no order reduction, are constructed and applied in extensive numerical experiments to validate the theoretical findings and to demonstrate their advantages. Approximate matrix factorization (AMF) technique exploits the structure of the Jacobian of the implicit parts, which may lead to further efficiency improvement of IMEX schemes. We have explored the application of AMF within some high order IMEX Runge-Kutta schemes in order to achieve high efficiency.

Sensitivity analysis gives quantitative information about the changes in a dynamical model outputs caused by caused by small changes in the model inputs. This information is crucial for data assimilation, model-constrained optimization, inverse problems, and uncertainty quantification. We develop a high performance software package for sensitivity analysis in the context of stiff and nonstiff ordinary differential equations. Efficiency is demonstrated by direct comparisons against existing state-of-art software on a variety of test problems.

This work was supported in part by the National Science Foundation through the awards NSF DMS–0915047, NSF CCF–0916493, NSF OCI–0904397, NSF CMMI–1130667, NSF CCF–1218454, by the Air Force Office of Scientific Research AFOSR FA9550–12–1–0293–DEF, AFOSR 12–2640–060 and by the Computational Science Laboratory at Virginia Tech.

Dedication

To my wife, my son and my parents.

Acknowledgments

I would like to thank my advisor Prof. Adrian Sandu for the countless weekly meetings, guidance, encouragement, support, advice, and patience over the past five years. Without his supervision, this dissertation would have not been possible.

I would also like to thank Profs. Yang Cao, Traian Iliescu, Tao Lin, Calvin J. Ribbens, and Raymond Spiteri for serving on my committee.

I acknowledge my collaborators, Dr. Sebastien Blaise, Dr. Angelamaria Cardone and Prof. Zdzislaw Jackiewicz for their time and talents. I am also grateful to Jose Garcia and Dr. Emil Constantinescu for offering me great summer research experiences at National Center for Atmospheric Research and Argonne National Laboratory respectively.

Looking back at the courses I have taken at Virginia Tech, the numerical analysis courses taught by Prof. Adjerid Slimane stand out to me. I thank him for technically bringing me into the wonderful numerical world, and for never giving perfect scores for homework so that I always worked hard to get the best.

I owe gratitude to my best friend Prof. Kunpeng Zhang. His continuous help drives me to pursue an academic path.

I would like to give special thanks to my dear wife Yang. I will always appreciate her love, understanding and sacrifices. Her support has made this dissertation a reality.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem formulation

Numerical solution of systems of differential equations is essential for simulating complex physical and engineered systems characterized by multiple interacting physical processes. Solving such multiphysics and multiscale problems poses considerable challenges due to their large scale, and to complexities resulting from the computation of interacting components with different dynamical properties [1]. Examples of relevant applications include fission reactor fuel performance, reactor core modeling, crack propagation, DNA sequencing, fusion, subsurface science, hydrology, climate, geodynamics, and accelerator design [2].

This work focuses on *time discretization algorithms* which can be applied to solve ordinary differential equations (ODEs) or time-dependent partial differential equations (PDEs). To fix ideas consider the following system of ODEs

$$y' = F\left(t, y; p\right) \ , \quad t_0 \leq t \leq t_F \ , \quad y(t_0) = y_0(p) \, . \tag{1.1}$$

Here $y(t;p) \in \mathbb{R}^d$ is the solution vector, y_0 the initial condition, and $p \in \mathbb{R}^m$ a vector of model parameters. This category of problems are also known as initial value problems (IVPs). In the method of lines approach time-dependent PDEs are represented in the form (1.1) after the spatial discretization.

Multiphysics and multiscale problems may result in problems in which the right-hand side F has components with widely different dynamics, often classified as stiff and non-stiff. Such problems can be expressed concisely as

$$y' = F\left(t, y; p\right) := f(t, y; p) + g(t, y; p) \quad t_0 \leq t \leq t_F \, , \quad y(t_0) = y_0(p) \, , \tag{1.2}$$

where f corresponds to the nonstiff term, and g corresponds to the stiff term.

The time discretization of multiple components in multiphysics and multiscale systems may impose more severe constraints on stability, accuracy, or robustness than the limitations

associated with each individual component. For example, in advection-diffusion-reaction systems, the advection is usually a slow process and can be solved efficiently with an explicit time integration scheme; however, the diffusion and chemistry processes can be very fast, restricting the time steps of explicit methods to impractically small values due to stability constraints [3, 4, 5].

Research question 1. *What are the most effective time stepping methods for multiphysics and multiscale systems?* Our point of view is that numerical approaches that allow the use of different strategies for different components are essential for providing an efficient and time-accurate solution to these problems.

Partitioned time stepping schemes apply different numerical discretizations to different components; for example, implicit-explicit (IMEX) schemes treat the nonstiff term explicitly and the stiff term implicitly, therefore combing the low cost of explicit methods with the favorable stability properties of implicit methods. Partitioned methods have been designed to tackle the temporal coupling of multiphysics components [4, 6, 7, 8, 9, 10, 11, 12, 13], to improve efficiency and stability for multiscale problems such as solid mechanics and heat transfer applications [14, 15], and to overcome geometry-induced stiffness in fluid-flow problems [16].

Objective 1. One objective of this research is to develop novel partitioned time stepping methods for the efficient simulation of large scale ODE systems and time-dependent PDE systems. Specifically, we study new IMEX methods based on general linear methods for multiscale and multiphysics time discretization.

Systems such as diffusion-reaction PDEs often lead to an ODE system in a semi-linear form

$$y' = F(y) = \mathbf{L}\, y + f(y)\,, \quad y \in \mathbb{R}^N \tag{1.3}$$

where $\mathbf{L}$ is a constant matrix (Jacobian of the stiff component). For example, $\mathbf{L}$ may correspond to the diffusion term that is approximated by a linear discrete Laplace operator in finite difference. In each time step, IMEX methods need to solve a linear system involving $\mathbf{L}$

$$(\mathbf{I} - h\,\gamma\,\mathbf{L})\; x = b, \quad x, b \in \mathbb{R}^N, \tag{1.4}$$

where h is the step size and γ is a parameter of the implicit integration scheme.

The matrix $\mathbf{L}$ is usually sparse but have a large bandwidth, especially when high order spatial discretization schemes are applied. The complexity of the structure of the matrix also increases with the dimensionality of the problems. So direct solve of the system (1.4) can be very costly. Assuming that $\mathbf{L} = \sum_{k=1}^{M} \mathbf{L}_k$, an *Approximate Matrix Factorization* (AMF) scheme uses the easier to factorize matrix

$$\mathbf{I} - h\,\gamma\,\mathbf{L} \approx \mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}} = \prod_{k=1}^{M} \left(\mathbf{I} - h\,\gamma\,\mathbf{L}_k\right), \tag{1.5}$$

where the subsystems $(\mathbf{I} - h\,\gamma\,\mathbf{L}_k)$ can usually be solved much faster and expose more parallelism.

Objective 2. Another objective coming along with the first one is to investigate the (AMF) technique that is known to be useful for efficiently solving systems of diffusion-reaction type. We want to explore better ways to apply it to high order IMEX schemes and evaluate its potential benefits and value for large scale stiff systems.

Sensitivity analysis quantifies the relationship between changes in model parameters and changes in model outputs. It has become a crucial ingredient for complex applications such as data assimilation, model-constrained optimization, inverse problems and uncertainty quantification. For example, sensitivity analysis helps identify the most influential parameters and help the modeler better understand the dynamics of a system. The sensitivity of a solution to changes in the data may provide insight that leads to improvements in the model.

Research question 2. *What tools are needed to efficiently perform sensitivity analysis in large scale simulations?*

In the context of the ODE system (1.1), sensitivity analysis yields derivatives of the solution, or a function of the solution $\Psi\big(y(t;p)\big)$, with respect to the initial conditions or system parameters, as follows

$$S_\ell(t) = \frac{\partial \Psi\big(y(t;p)\big)}{\partial p_\ell}, \quad 1 \le \ell \le m. \tag{1.6}$$

Here we assume that the system parameters p are independent of time.

Objective 3. A third objective of this research is to develop industrial-strength software for ODE simulations, endowed with sensitivity analysis capabilities, and which can be easily incorporated into other simulation codes.

1.2 Current state-of-the-art methodologies for integration and sensitivity analysis

This section summarizes relevant work from the literature related to our research objectives.

ODE solvers with sensitivity analysis capabilities. Only a few available software packages for the solution of ODEs have the capability to compute sensitivities. One of the earlier packages is Odessa [17], which performs direct sensitivity analysis. A modern package is CVODES within SUNDIALS [18] from Lawrence Livermore National Laboratory. CVODES is able to compute direct and continuous adjoint sensitivities. Both Odessa and CVODES are

based on backward differentiation formulas (BDF). A software based on explicit Runge-Kutta discretizations is DENSERKS [19] which implements continuous adjoint sensitivity analysis for nonstiff ODEs. The Kinetic PreProcessor KPP [20, 21] is a widely used tool for the simulation of chemical kinetics, and incorporates Runge-Kutta and Rosenbrock solvers that are endowed with tangent linear and discrete adjoint sensitivity analysis capabilities.

General linear methods. The general linear method (GLM) family proposed by J.C. Butcher [22] generalizes both Runge-Kutta (RK) and linear multistep methods. The added complexity improves the flexibility to develop methods with better stability and accuracy properties. While Runge-Kutta and linear multistep methods are special cases of GLMs, the framework allows for the construction of many other methods as well. the diagonally implicit multistage integration methods (DIMSIM) [23] and two step Runge Kutta (TSRK) methods are two subclasses of general linear methods, which are both efficient and accurate, and great potentials for practical use. GLM can overcome the limitations of both linear multistep methods (lack of A-stability at high orders) and of Runge-Kutta methods (low stage order leading to order reduction). A complete treatment of GLMs can be found in the book of Jackiewicz [24]. However, there is no available general purpose software or large-scale applications for GLMs.

Implicit-explicit methods. The idea of IMEX makes an optimal trade-off between efficiency and stability possible [25, 26, 27, 28]. IMEX linear multistep (LM) methods have been proposed in [29, 30, 31] and IMEX RK schemes in [32, 33, 34, 35, 36]. The orders of consistency of these methods is typically lower than five. High order IMEX RK methods are difficult to construct due to the large number of order conditions. IMEX LM methods have decreasing stability properties with increasing order. A special category of IMEX schemes based on Extrapolation methods, which can theoretically achieve very high order, are proposed in [27].

AMF technique. Calvo and Gerisch [37] studied the use of LIRK methods with AMF on reaction-diffusion systems. They proposed an approach to apply AMF to a transformed formula of LIRK methods which can avoid costly matrix vector multiplications. But only first order of convergence is obtained. To recover second order, they came up with a correction procedure on the solution at each time step. Applications of AMF with other time integration methods such as Rosenbrock methods and peer methods are investigated in [38, 39].

1.3 Research accomplishments

This section provides a summary of the main contributions of this dissertation.

1.3.1 ODE solvers with sensitivity analysis capabilities

We developed FATODE [40, 41], a Fortran library for the integration of ordinary differential equations with direct and adjoint sensitivity analysis capabilities. FATODE implements four families of methods – explicit Runge-Kutta for nonstiff problems and fully implicit Runge-Kutta, singly diagonally implicit Runge-Kutta, and Rosenbrock for stiff problems. Each family contains several methods with different orders of accuracy; users can add new methods by simply providing their coefficients. For each family the forward, adjoint, and tangent linear models are implemented. General purpose solvers for dense and sparse linear algebra are used; users can easily incorporate problem-tailored linear algebra routines. The performance of the package is demonstrated on several test problems. To the best of our knowledge FATODE is the first publicly available general purpose package that offers forward and adjoint sensitivity analysis capabilities in the context of Runge-Kutta methods. A wide range of applications are expected to benefit from its use; examples include parameter estimation, data assimilation, optimal control, and uncertainty quantification.

1.3.2 Practical IMEX two-step Runge-Kutta schemes

We have developed new implicit-explicit schemes in the family of two-step Runge-Kutta methods [42]. The class of schemes of interest is characterized by stage consistency (same abscissae) and linear invariant preservation (same weights). The study of order conditions for partitioned TSRK methods reveals that, in case of high stage orders, no additional coupling conditions need to be satisfied. Therefore this framework offers extreme flexibility in pairing implicit and explicit methods. The new framework allows to obtain IMEX schemes of order p and stage order $q = p - 1$ using only $s = p - 1$ stages. For $p \geq 4$ this is another advantage over existing IMEX Runge-Kutta schemes.

We construct two practical IMEX TSRK methods, of orders three and four, respectively. Their implicit parts are stiffly-accurate. The corresponding explicit parts have been constructed such as to maximize their stability properties; their coefficients were found via a numerical optimization approach. A convergence analysis for the Prothero-Robinson problem shows that the effective order of IMEX schemes in case of stiffness equals the stage order of the explicit part; a slight order reduction (from p to $q = p - 1$) is expected in the general case, but can be avoided in principle by using explicit methods with $q = p$.

Numerical examples include an advection-reaction system, the Van der Pol equation, and a semi-implicit integration of shallow water equations. Representative existing IMEX schemes of Runge-Kutta and linear multistep types are included for comparison. The results show that IMEX TSRK methods have favorable properties for stiff problems where IMEX Runge-Kutta methods suffer from severe order reduction. The larger shallow water test shows that the IMEX-TSRK methods are competitive in terms of accuracy and efficiency with the currently available families of methods.

A framework for partitioned and IMEX general linear methods

In [43, 44] we introduced a new family of partitioned time integration methods based on high stage order general linear methods. We proved that the general linear framework is well suited for the construction of multi-methods. Specifically, owing to the high stage orders, no coupling conditions are needed to ensure the order of accuracy of the partitioned GLM. We applied the partitioned general linear framework to construct new implicit-explicit GLM pairs, together with appropriate starting and ending procedures.

The linear stability analysis proposes the use of constrained stability functions to quantify the joint stability of the IMEX pair. A Prothero-Robinson convergence analysis reveals that the order of an IMEX GLM scheme on very stiff problems is dictated by the stage order of its non-stiff component; in particular, no order reduction appears if the explicit method has a full stage order. This result indicates that IMEX GLMs are particularly attractive for solving stiff problems, where other multistage methods may suffer from order reduction.

We constructed several practical IMEX GLM pairs starting from known implicit schemes and adding an appropriate explicit counterpart. This strategy is applied to build second and third order IMEX schemes based on DIMSIMs, which are a subfamily of general linear methods promising for practical applications. Numerical experiments with the van der Pol equation confirm the fact that IMEX GLMs converge at full order while IMEX RK methods suffer from order reduction. We implemented our algorithms in a discontinuous Galerkin finite element model developed by Blaise et al. [45], which has efficient parallel scalability. The results show that the new IMEX DIMSIM schemes perform slightly better than the IMEX RK methods of the same order.

1.3.3 Practical high-order IMEX general linear schemes

IMEX general linear methods can have similar stability properties as IMEX Runge-Kutta methods, but are superior in terms of accuracy and efficiency, especially at high orders. In the numerical solution of partial differential equations using a method-of-lines approach, the availability of high-order spatial discretization schemes motivates the development of sophisticated high-order time integration methods.

We construct two IMEX general linear methods of orders four and five, respectively. The schemes are based on DIMSIMs with $p = q = r = s$, where p and q are method order and stage order, r and s are number of external stages and number of internal stages. The fourth-order IMEX DIMSIM comes with a new L-stable implicit DIMSIM. The fifth-order IMEX DIMSIM consists of an existing L-stable implicit DIMSIM with refined accuracy in the coefficients. The stability regions of the new schemes are optimized numerically to cover as large a portion of the negative plane as possible. Compared to high-order classical IMEX Runge-Kutta methods, our methods are more computationally efficient. The performance gain comes from the fact that our methods require fewer number of stages to achieve same

order, and more importantly, from the property of high-stage order that avoids the order reduction IMEX Runge-Kutta methods may suffer from when applied to stiff problems.

Numerical tests on the Allen-Cahn equation and the viscous Burgers' equation with time-varying Dirichlet boundary conditions using finite difference schemes for spatial discretization have been performed to show the advantages of our methods over classic high-order IMEX Runge-Kutta methods. The new methods have also been successfully applied to large-scale atmospheric simulations using high-order discontinuous Galerkin schemes for spatial discretization. Two-dimensional and three-dimensional simulations of rising thermal bubbles are used for illustration. Results show that IMEX GLMs perform better than high-order IMEX Runge-Kutta methods in terms of accuracy and efficiency.

1.3.4 Application of approximate matrix factorization to high order linearly implicit integration methods.

Existing AMF approaches limit the application only to low order schemes due to the possible loss of order. We use a refinement approach which is based on inexact Newton iteration to apply the AMF technique to high order methods, at the same time maintaining the high order of accuracy and high efficiency. For convergence study, we analyze the linear system solution error and then show how this error propagates to the solution. We also look into the stability issues associated with AMF approach. Numerical results on a reaction-diffusion problem validate the effectiveness and evaluate the efficiency of our method applied to third-order and fourth-order linearly implicit Runge-Kutta methods.

1.4 Dissertation layout

The dissertation is organized as follows.

- Chapter 2 presents the general purpose library FATODE for the integration of ordinary differential equations and for direct and adjoint sensitivity analysis. The implementation, code structure and usage are discussed.
- Chapter 3 reviews the theory of general linear methods (GLMs). A novel framework of partitioned time stepping methods based on GLMs is constructed. The stability and accuracy theory for the new methods is established. The partitioned framework is applied to construct new IMEX GLMs, and to explain their advantages over classic IMEX schemes. New second order and third order IMEX methods are constructed and tested.
- Chapter 4 develops practical high order IMEX GLMs based on diagonally implicit multistage integration methods. Similar to the low order methods described in the

previous chapter, the new high order methods have inherent Runge-Kutta stability property. The constrained stability regions are maximized using numerical optimization. Numerical results with the Allen-Cahn and Burgers' equations show that the new methods do not suffer from order reduction. The new methods are implemented in a high performance geophysical simulation software based upon the mesh database of the GMSH mesh generator code [46], and applied to 2D and 3D simulations of atmospheric flows governed by Euler equations. The high order IMEX DIMSIMs are more efficient than classic IMEX Runge-Kutta methods.

- Chapter 5 studies another family of IMEX methods – IMEX two-step Runge-Kutta (TSRK) methods in the partitioned GLMs framework. We construct the third and fourth order IMEX TSRK methods that have a stiffly-accurate implicit component and an explicit component with numerically maximized stability regions. The theoretical findings are validated by numerical experiments on an advection-reaction system, van der Pol equations, and shallow water equations.

- Chapter 6 explores the application of AMF technique to high order linearly implicit Runge-Kutta methods, which are a special case of IMEX Runge-Kutta methods. By using a cheap correction procedure, similar to the inexact Newton's iteration, on each stage computation, high order and high accuracy is achieved with high order LIRK methods and AMF. Numerical experiments on reaction-diffusion type problems with different degrees of stiffness and scales illustrate the effectiveness and efficiency of our approach.

- Chapter 7 gives the conclusions of this work and future research directions.

Chapter 2

FATODE: A library for Forward, Adjoint and Tangent linear integration of ODEs

2.1 Introduction

Many dynamical systems in science and engineering are modeled by ordinary differential equations (ODEs)

$$y' = f\left(t, y; p\right) , \quad t_0 \leq t \leq t_F , \quad y(t_0) = y_0(p) . \tag{2.1}$$

Here $y(t) \in \mathbb{R}^d$ is the solution vector, y_0 the initial condition, and $p \in \mathbb{R}^m$ a vector of model parameters. Stiffness results from the existence of multiple dynamical scales, with the fastest characteristic times being much smaller than the time scales of interest in the simulation. It is well known that the numerical solution of stiff systems requires unconditionally stable discretizations which allow time steps that are not bounded by the fastest time scales in the system [47]. Here we assume that the system parameters p are independent of time. In the context of the ODE system (2.1), sensitivity analysis yields derivatives of the solution with respect to the initial conditions or system parameters, as follows

$$S_\ell(t) = \frac{\partial y(t)}{\partial p_\ell}, \quad \ell = 1, \ldots, m. \tag{2.2}$$

Two main approaches are available for computing the sensitivities (2.2). The direct (or tangent linear) method is efficient when the number of parameters is smaller than the dimension of the system ($m \ll d$), while the adjoint method is efficient when the number of parameters is larger than the dimension of the system ($m \gg d$). Furthermore, two distinct approaches can be taken for defining adjoint sensitivities; the continuous adjoint (differentiate then discretize) and the discrete adjoint (discretize then differentiate) approaches typically lead to different computational results.

Sensitivity analysis is an essential ingredient for uncertainty quantification, parameter estimation, optimization, optimal control, and construction of reduced order models. Only a few available software packages for the solution of ODEs have the capability to compute sensitivities. One of the earlier packages is Odessa [17], which performs direct sensitivity analysis. A modern package is CVODES within SUNDIALS [18] from Lawrence Livermore National Laboratory. CVODES is able to compute direct and continuous adjoint sensitivities. Both Odessa and CVODES are based on backward differentiation formulas (BDF). A software based on explicit Runge-Kutta discretizations is DENSERKS [19] which implements continuous adjoint sensitivity analysis for nonstiff ODEs. The Kinetic PreProcessor KPP [20, 21] is a widely used tool for the simulation of chemical kinetics, and incorporates Runge-Kutta and Rosenbrock solvers that are endowed with tangent linear and discrete adjoint sensitivity analysis capabilities.

In this chapter we present a library of explicit/implicit Runge-Kutta and Rosenbrock solvers for the simulation of nonstiff and stiff ODEs. The library, called FATODE [48, 40, 49, 50], performs forward simulations, and sensitivity analysis via the discrete adjoint and the tangent linear methods. FATODE brings new capabilities to the constellation of available sensitivity analysis software, as follows.

- FATODE is based on the KPP library of solvers. While the KPP implementation is specifically aimed at chemical kinetic systems, the FATODE implementation is general and suitable for a wide range of applications.
- FATODE is the first available general purpose software that implements a *discrete adjoint sensitivity analysis* approach. This gives gradients that are exact, within roundoff error, and therefore are highly suitable for numerical optimization problems. In contradistinction, both DENSERKS and CVODES implement a continuous adjoint approach, i.e., they solve the adjoint ODEs by applying the same numerical methods as for the forward ODEs.
- FATODE is the first general purpose package that implements sensitivity analysis for stiff systems using implicit Runge-Kutta and Rosenbrock methods. In contrast, DENSERKS uses explicit Runge-Kutta methods for non-stiff problems, while CVODES is based on linear multistep methods.
- Numerical tests show that FATODE is competitive with CVODES, from a computational efficiency perspective, both as an ODE integration package, as well as a forward and adjoint sensitivity solver. In addition, the FATODE library implements not one, but multiple methods, and the user has the possibility to select the best one for the application at hand.

The chapter is organized as follows. The background Section 2.2 reviews numerical integration algorithms, with emphasis on those implemented in FATODE, and summarizes the direct and adjoint approaches to sensitivity analysis. Section 2.3 discusses the array of tunable

parameters offered by the package, and provides general guidelines to select the best options for the problem at hand. Section 2.4 presents the code structure and implementation aspects of FATODE. Numerical tests with a nonstiff system are shown in Section 2.6, and with a stiff system in Section 2.8. Section 2.9 discusses discusses several important choices of methods and parameters. Section 2.10 summarizes the conclusions of this chapter.

2.2 Background

2.2.1 Numerical solution of ODEs

A considerable body of work has been devoted to the numerical solution of ODE systems (2.1), as presented in the monograph [51, 47]. Many classes of numerical discretizations are available and rigorous mathematical theories have been developed to analyze their accuracy and stability properties. Numerical discretizations can be broadly classified into explicit, which compute the new solution by repeated evaluations of the right hand side of (2.1), and implicit, which compute the new solution by solving (non-)linear systems of equations at each step. When solving nonstiff systems of ODEs the step sizes are decided solely based on accuracy requirements, and explicit methods [51] are preferred due to their lower cost per step. Implicit methods [47] are employed when solving stiff systems due to their better numerical stability properties.

FATODE implements four families of methods: explicit Runge-Kutta for nonstiff problems, and Rosenbrock, fully implicit Runge-Kutta, and singly diagonally implicit Runge-Kutta for stiff problems. Forward and adjoint sensitivity solvers are also included. The implicit methods have been previously implemented in KPP [20, 21] and have proved to be very efficient for solving many stiff chemical problems including CBM-IV [52], SAPRC [53] and NASA HSRP/AESA. The implementation of the forward implicit methods is inspired by the codes associated with the monograph [47].

All methods in FATODE are one-step integration formulas that advance the numerical solution $y_n \approx y(t_n)$ to $y_{n+1} \approx y(t_{n+1})$, $t_{n+1} = t_n + h$, using

$$y_{n+1} = \Phi^n(y_n, p), \quad n = 0, \ldots, N-1\,. \tag{2.3}$$

The specific families of methods are introduced below.

Runge-Kutta methods A general s-stage Runge-Kutta method reads [51]

$$T_i \;=\; t_n + c_j\, h\,, \quad Y_i = y_n + h \sum_{j=1}^{s} a_{i,j}\, f\left(T_j\,,\, Y_j\right)\,, \quad i = 1, \ldots, s\,, \tag{2.4a}$$

$$y_{n+1} \;=\; y_n + h \sum_{j=1}^{s} b_j \, f\left(T_j \,,\, Y_j\right) \,. \tag{2.4b}$$

where the coefficients

$$\boldsymbol{A} = [a_{i,j}]_{1\le i,j\le s} \;, \quad \boldsymbol{b} = [b_i]_{1\le i\le s} \;, \quad \boldsymbol{c} = [c_i]_{1\le i\le s} = \boldsymbol{A}\cdot \mathbf{1}_{(s,1)} \,, \tag{2.5}$$

define the method and determine its accuracy and stability properties. Explicit Runge-Kutta methods are characterized by the coefficients $a_{i,j} = 0$ for all i and $j \ge i$. Singly diagonal implicit Runge-Kutta methods are defined by (2.4) with $a_{i,i} = \gamma$ and $a_{i,j} = 0$ for all i and $j > i$. Fully implicit methods don't enjoy any special structure of the coefficient matrix and require coupled solutions for all the stage vectors, i.e., solutions of nonlinear systems of dimension $sd \times sd$.

Rosenbrock methods An s-stage Rosenbrock method [51] is given by the formulas

$$T_i = t_n + \alpha_i h \,, \quad Y_i = y_n + \sum_{j=1}^{i-1} \alpha_{i,j}\, k_j \,, \quad i = 1, \ldots, s \,, \tag{2.6a}$$

$$k_i = h\, f\left(T_i, Y_i\right) + h \boldsymbol{f_y}(t_n, y_n) \cdot \sum_{j=1}^{i} \gamma_{i,j} k_j + \gamma_i h^2 f_t(t_n, y_n), \tag{2.6b}$$

$$y_{n+1} = y_n + \sum_{j=1}^{s} b_j k_j \,, \tag{2.6c}$$

where particular methods are defined by their coefficients

$$\begin{aligned} &\boldsymbol{\alpha} = [\alpha_{i,j}]_{1\le i,j\le s} \;, \quad \boldsymbol{b} = [b_i]_{1\le i\le s} \;, \quad \boldsymbol{\gamma} = [\gamma_{i,j}]_{1\le i,j\le s} \;, \\ &\alpha_i = \sum_{j=1}^{i-1} \alpha_{i,j} \,, \quad \gamma_i = \sum_{j=1}^{i} \gamma_{i,j} \,; \quad \alpha_{i,j} = 0 \,,\ \forall\, i \ge j \,; \quad \gamma_{i,j} = 0 \,,\ \forall\, i > j \,. \end{aligned} \tag{2.7}$$

We have $\gamma_{i,i} = \gamma$ for all i for computational efficiency. Here $\boldsymbol{f_y} = \partial f / \partial y \in \mathbb{R}^{d\times d}$ represents the Jacobian of the ODE function, and $f_t = \partial f / \partial t$, as discussed in 2.2.2. We will denote matrices and tensors by bold symbols, and vectors and scalar by regular symbols. Rosenbrock methods are attractive because of their excellent stability properties and the conservation of linear invariants of the system. They typically outperform backward differentiation formulas such as those implemented in SMVGEAR [54] for medium accuracy solutions.

Methods implemented in FATODE The particular numerical schemes available in FATODE are summarized in Table 2.1.

The explicit Runge-Kutta (ERK) methods are suitable for non-stiff systems. The orders of accuracy range between two (RK2) and eight (DOPRI-853). The cost per step is (roughly) proportional to the number of stages.

All fully implicit Runge-Kutta (FIRK) methods in FATODE have three stages and require a coupled solution of all of them. This is implemented via two LU factorizations per step, one real and one complex. Thus the FIRK cost per step is the largest among all methods in FATODE. A-stability and L-stability are linear stability properties that guarantee that truncation errors do not accumulate quickly regardless of the stepsize [47]. L-stable methods are well suited for stiff problems as they guarantee a strong damping of the fast solution transients. Stiff accuracy is a property that allows for correct solutions of nonlinear systems in the limit of infinite stiffness (when the ordinary differential equation becomes a differential algebraic equation [47]).

Singly diagonally implicit Runge-Kutta (SDIRK) schemes perform a single LU factorization per time step, which is used to solve the nonlinear equations in each stage by simplified Newton equations. The number of forward-backward substitutions equals the total number of simplified Newton iteration for all stages. The methods implemented have orders two and four, and are all stiffly accurate, meaning that they are suitable for stiff problems.

Rosenbrock methods have the lowest cost per time step among all implicit schemes. Only linear systems need to be solved. There is one LU factorization per time step and as many forward-backward substitutions as there are stages. All methods implemented are L-stable. The stiff accuracy property is defined differently for Rosenbrock schemes than for Runge-Kutta schemes [47], but the meaning is similar: stiffly accurate methods provide correct results in the asymptotic limit of infinite stiffness.

2.2.2 Preliminaries on derivatives

Consider the following smooth function

$$\phi(y,p) \,:\, \mathbb{R}^{d+m} \to \mathbb{R}^n$$

and its first and second order derivatives

$$\begin{aligned}
\boldsymbol{\phi_y} &= \left(\frac{\partial \phi_i}{\partial y_j}\right)_{1\le i\le n,\ 1\le j\le d}, \quad \boldsymbol{\phi_p} = \left(\frac{\partial \phi_i}{\partial p_j}\right)_{1\le i\le n,\ 1\le j\le m}, \\
\boldsymbol{\phi_{y,y}} &= \left(\frac{\partial^2 \phi_i}{\partial y_j\,\partial y_k}\right)_{1\le i\le n,\ 1\le j,k\le d}, \quad \boldsymbol{\phi_{y,p}} = \left(\frac{\partial^2 \phi_i}{\partial y_j\,\partial p_k}\right)_{1\le i\le n,\ 1\le j\le d,\ 1\le k\le m}, \\
\boldsymbol{\phi_{p,p}} &= \left(\frac{\partial^2 \phi_i}{\partial p_j\,\partial p_k}\right)_{1\le i\le n,\ 1\le j,k\le m}, \quad \boldsymbol{\phi_{p,y}} = \left(\frac{\partial^2 \phi_i}{\partial p_j\,\partial y_k}\right)_{1\le i\le n,\ 1\le j\le m,\ 1\le k\le d}.
\end{aligned}$$

Table 2.1: Time stepping methods implemented in FATODE. (ERK, FIRK, SDIRK and ROS stand for Explicit Runge-Kutta, Fully Implicit Runge-Kutta, Singly Diagonally Implicit Runge-Kutta and Rosenbrock, respectively.)

Family	Method	Stages	Order	Stability properties
ERK	RK2(3) [51]	3	2	Conditionally stable
	RK3(2) [51]	4	3	Conditionally stable
	RK4(3) [51]	5	4	Conditionally stable
	DOPRI5 [51]	7	5	Conditionally stable
	Verner [51]	8	6	Conditionally stable
	DOPRI853 [51]	12	8	Conditionally stable
FIRK	Radau1A [47]	3	5	L-stable
	Radau2A [47]	3	5	L-stable, stiffly-accurate
	Lobatto3C [47]	3	4	L-stable, stiffly-accurate
	Gauss [47]	3	6	Weakly A-stable
SDIRK	Sdirk2a	2	2	L-stable, stiffly-accurate
	Sdirk2b	2	2	L-stable, stiffly-accurate
	Sdirk3a	3	2	L-stable, stiffly-accurate
	Sdirk4a [47]	5	4	L-stable, stiffly-accurate
	Sdirk4b [47]	5	4	L-stable, stiffly-accurate
ROS	Ros2 [55]	2	2	L-stable
	Ros3 [56]	3	3	L-stable
	Rodas3 [56]	4	3	L-stable, stiffly-accurate
	Ros4 [47]	4	4	L-stable
	Rodas4 [47]	6	4	L-stable, stiffly-accurate

Let $u, v \in \mathbb{R}^d$, $\overline{u}, \overline{v} \in \mathbb{R}^m$, and $z \in \mathbb{R}^n$. Hessian-vector products are matrices of the form

$$\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}} \cdot u = \left(\sum_{k=1}^{d} \frac{\partial^2 \phi_i}{\partial y_j \, \partial y_k} \, u_k \right)_{1 \le i \le n,\ 1 \le j \le d}, \quad z \cdot \boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}} = \left(\sum_{i=1}^{n} \frac{\partial^2 \phi_i}{\partial y_j \, \partial y_k} \, z_i \right)_{1 \le j,k \le d},$$

and similar for all other derivatives. The derivatives of Jacobian-vector products are:

$$\begin{aligned}
\left(\frac{d}{dy} \left(\boldsymbol{\phi}_{\boldsymbol{y}} \cdot u \right) \right) \cdot v &= \left(\sum_{k=1}^{d} \frac{d}{dy_k} \left(\sum_{i=1}^{d} \frac{\partial \phi_j}{\partial y_i} \, u_i \right) v_k \right)_{1 \le j \le n} \\
&= \left(\sum_{i,k=1}^{d} \frac{\partial^2 \phi_j}{\partial y_i \partial y_k} \, u_i \, v_k \right)_{1 \le j \le n} \\
&= \left(\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}} \cdot u \right) \cdot v \in \mathbb{R}^n \\
&= \left(\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}} \cdot v \right) \cdot u \in \mathbb{R}^n .
\end{aligned}$$

$$\left(\frac{d}{dy} \left(\boldsymbol{\phi}_{\boldsymbol{y}}^T \cdot z \right) \right) \cdot u = \left(\sum_{k=1}^{d} \frac{d}{dy_k} \left(\sum_{i=1}^{n} \frac{\partial \phi_i}{\partial y_j} \, z_i \right) u_k \right)_{1 \le j \le d}$$

$$
\begin{aligned}
&= \left(\sum_{k=1}^{d} \sum_{i=1}^{n} \frac{\partial^2 \phi_i}{\partial y_j \, \partial y_k} \, z_i \, u_k \right)_{1 \le j \le d} \\
&= (z \cdot \boldsymbol{\phi_{y,y}}) \cdot u \in \mathbb{R}^d \\
&= (\boldsymbol{\phi_{y,y}} \cdot u)^T \cdot z \in \mathbb{R}^d \, .
\end{aligned}
$$

The derivatives of the Jacobian with respect to model parameters are

$$
\begin{aligned}
\left(\frac{d}{dy} \left(\boldsymbol{\phi_p} \cdot \overline{u} \right) \right) \cdot u &= \left(\sum_{k=1}^{d} \frac{d}{dy_k} \left(\sum_{i=1}^{m} \frac{\partial \phi_j}{\partial p_i} \, \overline{u}_i \right) \, u_k \right)_{1 \le j \le n} \\
&= \left(\sum_{i=1}^{m} \sum_{k=1}^{d} \frac{\partial^2 \phi_j}{\partial p_i \partial y_k} \, u_k \, \overline{u}_i \right)_{1 \le j \le n} \\
&= (\boldsymbol{\phi_{p,y}} \cdot u) \cdot \overline{u} \in \mathbb{R}^n \\
&= (\boldsymbol{\phi_{y,p}} \cdot \overline{u}) \cdot u \in \mathbb{R}^n \, .
\end{aligned}
$$

$$
\begin{aligned}
\left(\frac{d}{dy} \left(\boldsymbol{\phi}_{\boldsymbol{p}}^T \cdot z \right) \right) \cdot u &= \left(\sum_{k=1}^{d} \frac{d}{dy_k} \left(\sum_{i=1}^{n} \frac{\partial \phi_i}{\partial p_j} \, z_i \right) \, u_k \right)_{1 \le j \le m} \\
&= \left(\sum_{i=1}^{n} \sum_{k=1}^{d} \frac{\partial^2 \phi_i}{\partial p_j \, \partial y_k} \, u_k \, z_i \right)_{1 \le j \le m} \\
&= (z \cdot \boldsymbol{\phi_{p,y}}) \cdot u \in \mathbb{R}^m \\
&= (\boldsymbol{\phi_{p,y}} \cdot u)^T \cdot z \in \mathbb{R}^m \, .
\end{aligned}
$$

$$
\begin{aligned}
\left(\frac{d}{dp} \left(\boldsymbol{\phi_y} \cdot u \right) \right) \cdot \overline{u} &= \left(\sum_{k=1}^{m} \frac{d}{dp_k} \left(\sum_{i=1}^{d} \frac{\partial \phi_j}{\partial y_i} \, u_i \right) \, \overline{u}_k \right)_{1 \le j \le n} \\
&= \left(\sum_{i=1}^{d} \sum_{k=1}^{m} \frac{\partial^2 \phi_j}{\partial p_k \partial y_i} \, u_i \, \overline{u}_k \right)_{1 \le j \le n} \\
&= (\boldsymbol{\phi_{p,y}} \cdot u) \cdot \overline{u} \in \mathbb{R}^n \\
&= (\boldsymbol{\phi_{y,p}} \cdot \overline{u}) \cdot u \in \mathbb{R}^n \, .
\end{aligned}
$$

$$
\begin{aligned}
\left(\frac{d}{dp} \left(\boldsymbol{\phi}_{\boldsymbol{y}}^T \cdot z \right) \right) \cdot \overline{u} &= \left(\sum_{k=1}^{m} \frac{d}{dp_k} \left(\sum_{i=1}^{n} \frac{\partial \phi_i}{\partial y_j} \, z_i \right) \, \overline{u}_k \right)_{1 \le j \le d} \\
&= \left(\sum_{i=1}^{n} \sum_{k=1}^{m} \frac{\partial^2 \phi_i}{\partial p_k \, \partial y_j} \, \overline{u}_k \, z_i \right)_{1 \le j \le d}
\end{aligned}
$$

$$
\begin{aligned}
&= (z \cdot \boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{p}}) \cdot \overline{u} \in \mathbb{R}^d \\
&= (z \cdot \boldsymbol{\phi}_{\boldsymbol{p},\boldsymbol{y}})^T \cdot \overline{u} \in \mathbb{R}^d \\
&= (\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{p}} \cdot \overline{u})^T \cdot z \in \mathbb{R}^d .
\end{aligned}
$$

They are used to express derivatives of Jacobian-vector products as follows:

$$
\begin{aligned}
\left(\frac{d}{dy}\left(\boldsymbol{\phi}_{\boldsymbol{y}} \cdot u\right)\right) \cdot v &= (\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}} \cdot u) \cdot v = (\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}} \cdot v) \cdot u \in \mathbb{R}^n , \\
\left(\frac{d}{dy}\left(\boldsymbol{\phi}_{\boldsymbol{y}}^T \cdot z\right)\right) \cdot u &= (z \cdot \boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}}) \cdot u = (\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{y}} \cdot u)^T \cdot z \in \mathbb{R}^d , \\
\left(\frac{d}{dp}\left(\boldsymbol{\phi}_{\boldsymbol{p}} \cdot \overline{u}\right)\right) \cdot \overline{v} &= (\boldsymbol{\phi}_{\boldsymbol{p},\boldsymbol{p}} \cdot \overline{u}) \cdot \overline{v} = (\boldsymbol{\phi}_{\boldsymbol{p},\boldsymbol{p}} \cdot \overline{v}) \cdot \overline{u} \in \mathbb{R}^n , \\
\left(\frac{d}{dp}\left(\boldsymbol{\phi}_{\boldsymbol{p}}^T \cdot z\right)\right) \cdot \overline{u} &= (z \cdot \boldsymbol{\phi}_{\boldsymbol{p},\boldsymbol{p}}) \cdot \overline{u} = (\boldsymbol{\phi}_{\boldsymbol{p},\boldsymbol{p}} \cdot \overline{u})^T \cdot z \in \mathbb{R}^m , \\
\left(\frac{d}{dy}\left(\boldsymbol{\phi}_{\boldsymbol{p}} \cdot \overline{u}\right)\right) \cdot u &= (\boldsymbol{\phi}_{\boldsymbol{p},\boldsymbol{y}} \cdot u) \cdot \overline{u} = (\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{p}} \cdot \overline{u}) \cdot u \in \mathbb{R}^n , \\
\left(\frac{d}{dy}\left(\boldsymbol{\phi}_{\boldsymbol{p}}^T \cdot z\right)\right) \cdot u &= (\boldsymbol{\phi}_{\boldsymbol{p},\boldsymbol{y}} \cdot u)^T \cdot z \in \mathbb{R}^m , \\
\left(\frac{d}{dp}\left(\boldsymbol{\phi}_{\boldsymbol{y}} \cdot u\right)\right) \cdot \overline{u} &= (\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{p}} \cdot \overline{u}) \cdot u \in \mathbb{R}^n , \\
\left(\frac{d}{dp}\left(\boldsymbol{\phi}_{\boldsymbol{y}}^T \cdot z\right)\right) \cdot \overline{u} &= (\boldsymbol{\phi}_{\boldsymbol{y},\boldsymbol{p}} \cdot \overline{u})^T \cdot z \in \mathbb{R}^d .
\end{aligned}
$$

Consider now the extended ODE (2.11). The right hand side function has the following extended Jacobian

$$
\frac{d\ [f, 0, r]}{d\ [y, p, q]} = \boldsymbol{J}(t, y, p) = \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{y}}(t, y, p) & \boldsymbol{f}_{\boldsymbol{p}}(t, y, p) & \mathbf{0}_{(d,1)} \\ \mathbf{0}_{(m,d)} & \mathbf{0}_{(m,m)} & \mathbf{0}_{(m,1)} \\ \boldsymbol{r}_{\boldsymbol{y}}(t, y, p) & \boldsymbol{r}_{\boldsymbol{p}}(t, y, p) & 0_{(1,1)} \end{bmatrix} , \tag{2.8}
$$

$$
\boldsymbol{J}^T(t, y, p) = \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{y}}^T(t, y, p) & \mathbf{0}_{(d,m)} & \boldsymbol{r}_{\boldsymbol{y}}^T(t, y, p) \\ \boldsymbol{f}_{\boldsymbol{p}}^T(t, y, p) & \mathbf{0}_{(m,m)} & \boldsymbol{r}_{\boldsymbol{p}}^T(t, y, p) \\ \mathbf{0}_{(1,d)} & \mathbf{0}_{(1,m)} & 0_{(1,1)} \end{bmatrix} .
$$

The extended Hessian times vector terms reads:

$$
\begin{aligned}
\left(\begin{bmatrix} u \\ \overline{u} \\ \widehat{u} \end{bmatrix} \cdot \widetilde{H}\right) \cdot \begin{bmatrix} v \\ \overline{v} \\ \widehat{v} \end{bmatrix} &= \frac{d}{d\ [y, p, q]} \left(\begin{bmatrix} \boldsymbol{f}_{\boldsymbol{y}}^T & \mathbf{0} & \boldsymbol{r}_{\boldsymbol{y}}^T \\ \boldsymbol{f}_{\boldsymbol{p}}^T & \mathbf{0} & \boldsymbol{r}_{\boldsymbol{p}}^T \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ \overline{u} \\ \widehat{u} \end{bmatrix}\right) \cdot \begin{bmatrix} v \\ \overline{v} \\ \widehat{v} \end{bmatrix} \\
&= \begin{bmatrix} (\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{y}} \cdot v)^T \cdot u + (\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{p}} \cdot \overline{v})^T \cdot u + (\boldsymbol{r}_{\boldsymbol{y},\boldsymbol{y}} \cdot v)^T \cdot \widehat{u} + (\boldsymbol{r}_{\boldsymbol{y},\boldsymbol{p}} \cdot \overline{v})^T \cdot \widehat{u} \\ (\boldsymbol{f}_{\boldsymbol{p},\boldsymbol{y}} \cdot v)^T \cdot u + (\boldsymbol{f}_{\boldsymbol{p},\boldsymbol{p}} \cdot \overline{v})^T \cdot u + (\boldsymbol{r}_{\boldsymbol{p},\boldsymbol{y}} \cdot v)^T \cdot \widehat{u} + (\boldsymbol{r}_{\boldsymbol{p},\boldsymbol{p}} \cdot \overline{v})^T \cdot \widehat{u} \\ \mathbf{0} \end{bmatrix} .
\end{aligned} \tag{2.9}
$$

2.2.3 Sensitivity analysis

Consider an output vector $\Psi \in \mathbb{R}^o$ that depends on the solution of the system (2.1)

$$\Psi = g\left(y(t_F), p\right) + \int_{t_0}^{t_F} r\left(t, y(t), p\right)\, dt\,, \tag{2.10}$$

where $g : \mathbb{R}^{d+m} \to \mathbb{R}^o$ and $r : \mathbb{R}^{1+d+m} \to \mathbb{R}^o$. For example, the first terms can penalize the discrepancy between the model state and a target state at the final time, $g(t_F) = \|y - y_{\text{target}}(t_F)\|$. The integral terms can measure the discrepancy between quantities $\mathcal{O}(y)$ predicted by the model and measurements of the same quantities in the physical world, $r = \|\mathcal{O}(y(t)) - z_{\text{observed}}(t)\|$. In the solution of inverse problems the entries of Ψ are referred to as *objective functions*, and in the context of uncertainty quantification as *quantities of interest*.

The output functions (2.10) can be formulated solely in terms of the final state by extending the ODE system. To account for the evolution of the parameters we add the formal equations for the parameter evolution $p' = 0$. To compute the integral terms in (2.10) we add the quadrature variables $q \in \mathbb{R}^o$ whose evolution is defined by $q(t_0) = 0$ and $q' = r\left(t, y, p\right)$. The equation (2.1) and the outputs (2.10) become

$$\begin{bmatrix} y \\ p \\ q \end{bmatrix}' = \begin{bmatrix} f\left(t, y, p\right) \\ 0 \\ r\left(t, y, p\right) \end{bmatrix}, \quad t_0 \le t \le t_F\,; \quad \begin{bmatrix} y(t_0) \\ p(t_0) \\ q(t_0) \end{bmatrix} = \begin{bmatrix} y_0 \\ p \\ 0 \end{bmatrix}, \tag{2.11}$$

$$\Psi = g\left(y(t_F), p\right) + q(t_F)\,. \tag{2.12}$$

Since (2.10) is equivalent to (2.12) in the remainder of this section we will consider $r = 0$ in order to simplify the presentation, and without loss of generality. However, as explained later, the FATODE implementation treats the case $r \neq 0$ separately to enhance computational efficiency.

Sensitivity analysis yields the derivatives of the model outputs $\Psi_1, \ldots, \Psi_o$, with respect to the model inputs, i.e., the parameters $p_1, \ldots, p_m$. The two main approaches to compute the sensitivity matrix $d\Psi/dp \in \mathbb{R}^{o\times m}$ are discussed next.

2.2.4 Direct sensitivity analysis

Small changes δp in the parameters result in small perturbations $\delta y(t)$ of the solution of ODE system (2.1), which in turn lead to small changes $\delta\Psi$ in the model outputs. Let $\dot{p} = \delta p / \|\delta p\|$ be the scaled perturbation and $\dot{y} = \delta y / \|\delta p\|$ be the directions of solution change. These directions propagate forward in time according to the *tangent linear ODE*:

$$\dot{y}' = \boldsymbol{f_y}(t, y; p) \cdot \dot{y} + \boldsymbol{f_p}(t, y; p) \cdot \dot{p}\,, \quad \dot{y}(t_0) = \frac{dy_0}{dp} \cdot \dot{p}\,, \quad \dot{y}(t) \in \mathbb{R}^d\,. \tag{2.13}$$

The sensitivity matrix $d\Psi/dp$ is computed column by column, as follows. Solve the tangent linear model (2.13) with $\dot{p} = e_\ell$ to obtain $\dot{y}_\ell = S_\ell$ (2.2). Here $e_\ell \in \mathbb{R}^m$ is a vector with the ℓ-th entry equal to one, and all other entries equal to zero. The ℓ-th column of the sensitivity matrix is

$$\frac{d\Psi}{dp_\ell} = \mathbf{g_y}\left(y(t_F), p\right) \cdot \dot{y}_\ell(t_F) + \mathbf{g_{p_\ell}}\left(y(t_F), p\right) \ , \quad \ell = 1, \ldots, m \,, \tag{2.14}$$

(recall that we now assume $r = 0$ without loss of generality). The computational cost of the forward sensitivity analysis is dominated by the m integrations of the tangent linear model (2.13). Therefore the forward approach works best when the number of parameters m is small. On the other hand, the number of outputs defines the dimension of (2.14) but has a small influence on (2.13), therefore it has only a small impact on the total computational cost.

In principle two approaches are possible to obtain the sensitivities in an application. In the *continuous forward sensitivity* approach one first differentiates the forward ODE system (2.1) to obtain the continuous tangent linear ODE (2.13). The forward and the tangent linear ODE systems are solved *together* forward in time. For example, an application of the implicit Euler method leads to the numerical solution

$$y_{n+1} = y_n + h\, f(t_{n+1}, y_{n+1}, p) \,, \tag{2.15a}$$

$$\dot{y}_{n+1} = \dot{y}_n + h\, \boldsymbol{f_y}(t_{n+1}, y_{n+1}, p) \cdot \dot{y}_{n+1} + h\, \boldsymbol{f_p}(t_{n+1}, y_{n+1}, p) \cdot \dot{p} \,. \tag{2.15b}$$

In this case $\dot{y}_n$ represents a numerical approximation of the continuous forward sensitivities $\dot{y}(t_n)$.

In the *discrete forward sensitivity* approach one starts with the numerical approximation of the forward system (2.3) and differentiates it in the direction $\dot{p}$

$$\dot{y}_{n+1} = \mathbf{\Phi^n_y}(y_n, p) \cdot \dot{y}_n + \mathbf{\Phi^n_p}(y_n, p) \cdot \dot{p}, \quad n = 0, \ldots, N-1 \,. \tag{2.16}$$

In this case $\dot{y}_n$ represents the sensitivity of the forward numerical solution $dy_n/dp \cdot \dot{p}$.

It is easy to see that the differentiation of the implicit Euler solution (2.15a) with respect to parameters leads to discrete sensitivity equations (2.16) that are precisely (2.15b). Like with implicit Euler, for all methods implemented in FATODE the continuous and the discrete forward sensitivity approaches lead to exactly the same results. FAOTODE solves the combined equations (2.15) in an efficient manner; for example, in case of implicit schemes, the LU factorization used to solve the sensitivity equation (2.15b) is re-used in the next step to solve the forward system (2.15a).

2.2.5 Adjoint sensitivity analysis

In adjoint sensitivity analysis the matrix $d\Psi/dp$ is computed row by row, as follows. For each output function Ψ_i solve the following *adjoint ODE* for $\lambda_i(t) \in \mathbb{R}^d$ and $\mu_i(t) \in \mathbb{R}^m$

$$\begin{aligned} \lambda_i' &= -\boldsymbol{f_y}^T(t,y;p)\cdot\lambda_i\,, \quad \lambda_i(t_F) = (\boldsymbol{g_i})_{\boldsymbol{y}}^T\,(y(t_F),p)\ , \\ \mu_i' &= -\boldsymbol{f_p}^T(t,y;p)\cdot\lambda_i\,, \quad \mu_i(t_F) = (\boldsymbol{g_i})_{\boldsymbol{p}}^T\,(y(t_F),p)\ ; \quad t_F \geq t \geq t_0\,. \end{aligned} \tag{2.17}$$

The adjoint equation (2.17) is derived using variational calculus [57, 58]. We observe that the adjoint equation is solved backwards in time, and that its formulation depends on the forward model state $y(t)$. Therefore in order to solve (2.17) one needs to first solve (2.1) forward in time and save the entire solution $y(t)$. This solution is used to form the Jacobians during the backward in time adjoint integration.

It can be shown that the the sensitivity of the i-th output with respect to all parameters can be obtained from the adjoint variables as [57, 58]

$$\frac{d\Psi_i}{dp} = \mu_i^T(t_0) + \lambda_i^T(t_0)\cdot\left(\frac{dy_0}{dp}\right)\,, \quad i = 1,\ldots,o\,.$$

The cost of computing the entire sensitivity matrix $d\Psi/dp$ is dominated by the repeated solution of the adjoint systems (2.17) for $i = 1,\ldots,o$. Therefore the adjoint method is most effective when the number of outputs o is small. The number m of model inputs (parameters) defines the size of μ_i. Since these variables are obtained via relatively inexpensive Jacobian-transposed vector products ${\mathbf{f_p}}^T\lambda_i$, the number of parameters m does not impact considerably the overall computational cost.

Two approaches are possible to evaluate numerically the sensitivities $d\Psi/dp$. The *continuous adjoint approach* applies a numerical discretization to solve the adjoint ODE (2.17). Since the adjoint ODE is formed first, this strategy is also called differentiate-then-discretize approach. For example, if the implicit Euler method (2.15) is used to solve (2.17) one obtains

$$\begin{aligned} (\overline{\lambda}_i)_N &= (\boldsymbol{g_i})_{\boldsymbol{y}}^T\,(y(t_F),p)\ ; \quad (\overline{\lambda}_i)_n = (\overline{\lambda}_i)_{n+1} + h\,\boldsymbol{f_y}^T(t_n,y(t_n);p)\cdot(\overline{\lambda_i})_n\,, \\ (\overline{\mu}_i)_N &= (\boldsymbol{g_i})_{\boldsymbol{p}}^T\,(y(t_F),p)\ ; \quad (\overline{\mu}_i)_n = (\overline{\mu}_i)_{n+1} + h\,\boldsymbol{f_p}^T(t_n,y(t_n);p)\cdot(\overline{\lambda}_i)_n\,, \end{aligned}$$

for $N-1 \geq n \geq 0$. In practice the Jacobian arguments are replaced by numerical approximations of the forward trajectory. In general the forward steps and the continuous adjoint steps need not coincide. The Jacobian arguments are computed by interpolating the stored forward solution, such as to obtain intermediate state variables at the times required by the backward integration. The continuous adjoint variables are approximation of the solutions of the adjoint ODE (2.17), $(\overline{\lambda}_i)_n \approx \lambda_i(t_n)$ and $(\overline{\mu}_i)_n \approx \mu_i(t_n)$.

In the *discrete adjoint sensitivity* approach one starts with the numerical approximation of the forward system (2.3) and builds the adjoint (linearized transpose) of the discrete system

$$\begin{aligned} \lambda_N &= \boldsymbol{g_y}^T(y_N)\,; & \mu_N &= \boldsymbol{g_p}^T(y_N)\,; \\ \lambda_n &= \boldsymbol{\Phi_y}^n(y_n,p)^T\cdot\lambda_{n+1}; & \mu_n &= \boldsymbol{\Phi_p}^n(y_n,p)^T\cdot\lambda_{n+1}, \quad n = N-1,\ldots,0\,. \end{aligned} \tag{2.18}$$

For example, the discrete adjoint of the implicit Euler method (2.15) reads

$$(\lambda_i)_N = (\boldsymbol{g_i})_{\boldsymbol{y}}^T\,(y_N, p)\ ;\quad (\lambda_i)_n = (\lambda_i)_{n+1} + h\,\boldsymbol{f}_{\boldsymbol{y}}^T(t_{n+1}, y_{n+1}; p)\cdot(\lambda_i)_n\,,$$
$$(\mu_i)_N = (\boldsymbol{g_i})_{\boldsymbol{p}}^T\,(y_N, p)\ ;\quad (\mu_i)_n = (\mu_i)_{n+1} + h\,\boldsymbol{f}_{\boldsymbol{p}}^T(t_{n+1}, y_{n+1}; p)\cdot(\lambda_i)_n\,,$$

for $N-1 \geq n \geq 0$. The arguments of $\boldsymbol{f}_{\boldsymbol{y}}^T$, $\boldsymbol{f}_{\boldsymbol{p}}^T$ are the numerical approximations of the implicit Euler method applied to the forward problem (2.15). The discrete adjoint approach (2.19) follows exactly the same sequence of time steps as the forward integration (2.15), but in reverse order. The discrete adjoint variables represent derivatives of the numerical solution, e.g., $(\lambda_i)_n = d\Psi_i(y_N)/dy_n$.

The continuous (differentiate-then-discretize) and the discrete (discretize-then-differentiate) adjoint approaches lead, in general, to different computational results [59]. This can be easily seen for our implicit Euler example by comparing equations (2.18) and (2.19). The arguments of $\boldsymbol{f}_{\boldsymbol{y}}^T$, $\boldsymbol{f}_{\boldsymbol{p}}^T$ differ; even if the same sequence of forward steps is used in both cases, the Jacobians are evaluated at y_n in the continuous adjoint, and at y_{n+1} in the discrete adjoint systems.

All sensitivity packages currently available (Odessa [17], CVODES [18], and DENSERKS [19]) implement a continuous adjoint sensitivity approach. Important characteristics of the continuous adjoint approach are as follows. Continuous adjoints are numerical solutions of the adjoint ODE, therefore they always approximate the continuous derivatives. However they are not the (exact) gradients of any function. The continuous approach offers implementation flexibility; the numerical method and the sequence of step sizes used to solve the adjoint ODE may differ from those used to solve the forward ODE, and may be tuned separately to satisfy the accuracy needs of the reverse integration. However, this comes at a cost; the forward numerical solution needs to be interpolated to the time points required in the adjoint integration.

The discrete adjoint approach has several important characteristics that distinguishes it from the continuous approach. Discrete adjoint variables are derivatives of the forward numerical solution. However, there is no guarantee that they are approximations of the continuous derivatives [57, 21]. The discrete approach provides exact gradients (within roundoff) of the numerical cost functions, which is advantageous in the solution of numerical optimization problems. The computational process and the sequence of step sizes are determined by the forward numerical method and by the forward steps; there is no flexibility to separately tune the adjoint integration. In the same time, all the variables needed to form the discrete adjoint are computed during the forward solution, and no additional interpolations are necessary. The human effort required to implement the adjoint of a complex model is considerable; discrete adjoints can be constructed automatically by algorithmic differentiation [60] to reduce this effort. In addition, one can compute sensitivities of ODEs described by legacy code, and for which the analytical formulation may be unavailable.

FATODE is the first package to implement discrete adjoints for all the methods considered. This allows the users to benefit from all the advantages of this approach described above.

In addition, the discrete adjoints implemented have good theoretical properties, in the sense that they are consistent discretizations of the adjoint ODE. We recall the following theorem from [57, 58].

Theorem 2.2.1 (Consistency of discrete Runge-Kutta adjoints)**.** *If a Runge-Kutta method* (2.4) *has order of consistency p, then its discrete adjoint* (2.18) *is an order p discretization of the adjoint ODE* (2.17)*.*

The proof of this theorem accounts for both the adjoint truncation error and for the approximation of the linearization point (the use of a forward discrete solution y_n instead of $y(t_n)$ in the Jacobian arguments). Similar arguments apply to the discrete adjoints of Rosenbrock methods (2.6).

2.3 Selection of different options in FATODE

FATODE offers a wide array of tunable parameters for the solution of ODEs and the calculation of sensitivities. We provide some general guidelines that could help a user select the options that best suit the problem at hand. A complete discussion of the available options is given in FATODE User's guide [49].

2.3.1 Selecting a family of methods

ERK are the methods of choice for non-stiff problems. SDIRK and ROS are preferable for stiff problems where a moderate accuracy is required, and FIRK for very stiff systems or when a high accuracy is needed. Among the implicit schemes the FIRK cost per step is the highest, while the ROS cost per step is the lowest.

2.3.2 Selecting a particular method within a family

This choice reflects the tradeoff between compute time and accuracy. For maximum efficiency one would choose schemes of order two to four for faster, low to medium accuracy calculations, and progressively higher order schemes for more accurate results. Methods of order five to eight are most efficient when high accuracies are sought.

As explained in Section 2.2.1 all L-stable methods are suitable for the integration of stiff systems as they provide strong damping of the fast components of the error. Methods which, in addition, are stiffly accurate retain the solution accuracy in the limit of infinite stiffness. Therefore stiffly accurate methods like Radau2A, Lobatto3C, Sdirk4{a,b}, and Rodas{3,4} are well suited for the integration of very stiff systems (2.1). On the other hand the Gauss method is only weakly A-stable, meaning that it only weakly attenuates fast transients, and

is therefore not suited for very stiff systems. Gauss method is useful when the energy of the system needs to be preserved, for example in the case of Hamiltonian dynamics.

2.3.3 Selecting a linear algebra solver

The most computationally intensive part in solving large-scale ODE systems by implicit methods is the solution of linear systems at each step. FATODE provides several direct linear algebra solvers that work well for small and medium size problems. The selection of a specific package depends on the problem at hand: LAPACK should be used for problems with full Jacobian matrices, and UMFPACK or SuperLU for problems with a sparse structure. For very large problems a better solution is to link the integrators to third party Krylov space iterative solvers and utilize problem specific preconditioners.

2.3.4 Choosing the direct or the adjoint approach to sensitivity calculations

The two main approaches to compute the sensitivity matrix $d\Psi/dp \in \mathbb{R}^{o\times m}$ have different computational complexities as explained in Section 2.2. As a rule of thumb the direct (or tangent linear) method is chosen when the number of parameters is smaller than the number of outputs ($m \ll o$), while the adjoint method is chosen when the number of parameters is larger than the number of outputs ($m \gg o$). When $m \approx o$ the direct method is preferable due to its lower implementation complexity.

For the adjoint method method the user has the option to compute sensitivities with respect to only the initial conditions, e.g., for a data assimilation application, or to compute sensitivities with respect to both model parameters and initial conditions, e.g., in a parameter estimation application.

2.3.5 Selecting the solution approach for the sensitivity systems

Both the tangent and the adjoint ODEs are linear, and during each step the corresponding sensitivity variables are are the solutions of a linear system of equations. FATODE offers the choice to build this system and to solve it directly, or to employ simplified Newton iterations that reuse the LU decompositions performed during forward calculation. If the LU factors can be stored then the simplified Newton iterations saves considerable CPU time in forward sensitivity mode by reusing the LU decomposition. In adjoint sensitivity calculations the computational savings need to be judged against the additional read/write overhead for checkpointing large, multiple LU factorized matrices, and against the physical dimensions of the tape. The direct approach should be selected if an iterative linear algebra solver is employed, and no explicit LU decomposition is available. When simplified Newton iterations

are used the computed adjoints are equal to the discrete gradients within the truncation error margin (and not to roundoff error).

2.3.6 Choosing tolerances

Absolute and relative tolerances reflect the desired degree of solution accuracy. It is a good idea to select slightly tighter tolerances for sensitivities than required by the forward application alone. For best performance absolute tolerances need to be chosen such as to reflect the magnitude of each solution component. This is typically very difficult with sensitivity calculations, as sensitivity coefficients can vary over many many orders of magnitude, and little information about their values is available apriori. Repeated calculations may be needed to properly calibrate absolute tolerances.

2.3.7 Computing derivatives

The stiff solvers, as well as the tangent linear and adjoint methods, require the computation of various derivatives such as the Jacobians of the ODE function or of the output functions with respect to the state and parameters. The derivatives supplied to FATODE can be obtained analytically, by finite differences, or by automatic differentiation. Details about the required derivatives and their implementation can be found in [61, part 2.5.7]. Analytical or automatic differentiation generated Jacobians and Hessians are to be preferred due to their accuracy and, most often, their computational efficiency. Finite differences are the least recommended as their accuracy is difficult to control, and their errors impact directly the computed sensitivities. ROS methods require most additional derivatives, including Hessians for sensitivity calculations. If derivatives are difficult to obtain, or expensive to run, then SDIRK or FIRK methods are to be preferred.

2.4 Code organization

FATODE implements four types of methods: explicit, fully implicit, and singly diagonally implicit Runge-Kutta methods, and Rosenbrock methods. For each family of methods, a module is given for the main integrator, a module for linear system solver interface, and a set of modules for generic linear system solvers. They form the basic structure shown in Figure 2.1.

The main integration module provides the basic time stepping framework, and is independent of the linear system solver.

- The forward integrator calls the user-supplied right-hand side function and Jacobian and accesses the linear system solver, in order to compute the ODE solution. Users

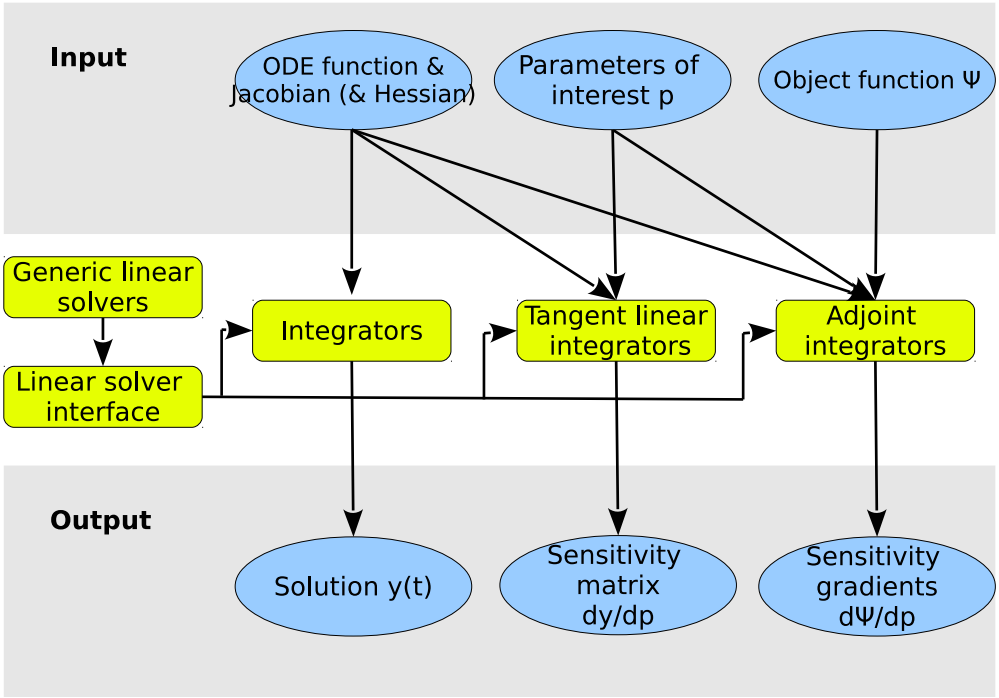


Figure 2.1: Overall structure of FATODE.

can employ FATODE as a high quality ODE solution library even when its sensitivity analysis capabilities are not needed. Details of the implementation of different Runge-Kutta and Rosenbrock methods are given in [61, part 2.5.1].

- The tangent linear integrator and the adjoint integrator require users to also specify the parameters of interest as additional inputs. The tangent linear integrator, by default, considers the initial conditions of the ODE system as the parameters of interest and computes the sensitivity of the ODE solution with respect to them. Highly efficient tangent linear implementations are obtained by re-using the LU decompositions from the forward solution on the sensitivity equations [62]. Details about each family of tangent linear methods are given in [61, part 2.5.2].

- For efficiency reasons FATODE provides two implementations of the adjoint integrator, one for sensitivities with respect to initial conditions, detailed in [61, part 2.5.3], and and one sensitivities with respect to parameters, discussed in [61, part 2.5.5]. By default scalar cost functions (2.10) are considered. The cost function and its derivatives are supplied by the user; cost function derivatives are used to define the adjoint initial conditions, see (2.18) and [61, Eqn. (2.47)], and to provide the forcing of the adjoint equations.

The linear system solver modules define the data structures for the Jacobians (e.g., dense, sparse), interfaces to routines that compute Jacobian (or its transpose) times vector products,

as well as interfaces to linear solvers. Linear algebra is implemented transparently to the ODE solvers via four generic routines: *LS_Init* (allocates memory and initializes the specific linear solver), *LS_Decomp* (LU decomposition), *LS_Solve* (solves the triangular systems by substitution), and *LS_Free* (frees the memory allocated during the initialization stage). They provide interfaces to dense (LAPACK [63]) and sparse (UMFPACK [64] and SuperLU [65]) linear algebra packages. Separate modules are provided for each of the implicit time stepping families in FATODE, as they perform different Jacobian operations and solve different types of systems (e.g., real-valued linear systems of dimension $d \times d$ for Rosenbrock and real and complex linear systems of dimension $d \times d$ for Runge-Kutta). All the required code modifications for a new application, or for adding a new solver, are done within the linear algebra modules. Details on the linear algebra part of FATODE are given in [61, part 2.5.8].

The adjoint model requires two runs. First the forward code performs a regular integration of the ODE system. At each step the time t_n, the step size, the forward solution vector, and the intermediate stage vectors are all saved in checkpoints. Next, the discrete adjoint code is run backward in time and traces the same sequence of steps, but in reverse order. At each step, the data in the checkpoint storage is retrieved and used to construct the adjoint system. The results of LU factorizations can, in principle, be checkpointed and reused in the adjoint calculations. This is not done in our implementation, due to the following reasons. The memory and input/output costs for storing LU factors can be extremely large when the system is large or when many steps are taken. Next, FATODE is designed such that the details of the linear solver are transparent to the main integrator. This transparency cannot be preserved when one builds and manages a stack for data structures that are specific to particular linear solvers.

Variable step size is employed by FATODE to control numerical errors and maximize efficiency. The forward integrators estimate the truncation errors using Runge-Kutta and Rosenbrock embedded solutions. Local error tests are performed based on the relative and absolute error tolerances specified by the user, and they are used to decide acceptance/rejection of the current solution as well as the size of the next step. For the forward sensitivity analysis FATODE allows to control the truncation errors for both the forward solution and the tangent linear model solution. There is no step size control during the discrete adjoint integration as it traces the same sequence of steps as the forward integration. However, user specified tolerances are employed to control the convergence of the iterative solutions of sensitivity equations. Implementation details for error estimates and step size control are given in [61, part 2.5.6].

To increase efficiency the FIRK and SDIRK implementations have the possibility to reuse previous LU factorizations in lieu of recomputing a new factorization at the current step. The reuse is allowed when the following three conditions are simultaneously met: the same LU factorization has not been in use for more than $N_{\max}$ consecutive steps; the ratio of the predicted step size to the current step size satisfies $q_{\min} \leq h_{\text{predicted}}/h \leq q_{\max}$; and the convergence rate of the simplified Newton iterations in previous step is smaller than $\theta_{\max}$. The default values are $N_{\max} = 20$, $q_{\min} = 1$, $q_{\max} = 1.2$, $\theta_{\max} = 0.001$, and they can be

changed by the user.

2.5 Implementation

2.5.1 FATODE implementation of forward model integration

FATODE provides a high quality solvers for the forward ODE problem (2.1). Even without the sensitivity analysis capabilities it can be used as a generic ODE solution library. In this section we discuss the efficient implementation of different Runge-Kutta and Rosenbrock methods for solving (2.1). The implementation of forward solvers is inspired by [51, 47].

Explicit Runge-Kutta methods

The implementation of explicit methods is based on (2.4). The stage vectors are computed in succession using

$$Y_1 = y_n\,; \quad Y_i = y_n + h \sum_{j=1}^{i-1} a_{i,j}\, f(T_j, Y_j)\,, \quad i = 2, \ldots, s\,. \tag{2.19}$$

Matrices for solving implicit methods

The implementations of implicit methods use the following matrices

$$\mathbf{R}\,(\gamma, t, y) = \boldsymbol{I}_{(d,d)} - h\,\gamma\,\boldsymbol{f_y}(t, y)\,, \tag{2.20a}$$

$$\widetilde{\mathbf{R}}\,(\gamma, t, y) = \frac{1}{h\,\gamma}\boldsymbol{I}_{(d,d)} - \boldsymbol{f_y}(t, y)\,, \tag{2.20b}$$

$$\widehat{\mathbf{R}}_n = \begin{bmatrix} 1 - h\,a_{1,1}\,\boldsymbol{f_y}(T_1, Y_1) & \cdots & -h\,a_{1,s}\,\boldsymbol{f_y}(T_s, Y_s) \\ \vdots & \ddots & \vdots \\ -h\,a_{s,1}\,\boldsymbol{f_y}(T_1, Y_1) & \cdots & 1 - h\,a_{s,s}\,\boldsymbol{f_y}(T_s, Y_s) \end{bmatrix} \in \mathbb{R}^{ds \times ds}\,. \tag{2.20c}$$

Replacing each $\boldsymbol{f_y}(T_i, Y_i)$ in (2.20c) by $\boldsymbol{f_y}(t_n, t_n)$ leads to the approximation:

$$\overline{\mathbf{R}}_n = \boldsymbol{I}_{(ds,ds)} - h\,\boldsymbol{A} \otimes \boldsymbol{f_y}(t_n\,,\, y_n) \approx \widehat{\mathbf{R}}_n\,, \tag{2.20d}$$

where $\otimes$ is the matrix Kronecker product [47].

Implicit Runge-Kutta methods

To reduce the influence of round-off errors, we apply the transformation $z_i = Y_i - y_n$ [47] in the formulas (2.4) to obtain

$$T_i = t_n + c_i\, h\,, \quad z_i = h \sum_{j=1}^{s} a_{i,j}\, f(T_j\,,\, y_n + z_j)\,, \quad i = 1, \ldots, s\,, \tag{2.21a}$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} d_i\, z_i\,. \tag{2.21b}$$

The new coefficients are

$$\boldsymbol{d} = [d_i]_{1 \le i \le s}\,, \quad \boldsymbol{d}^T = \boldsymbol{b}^T \cdot \boldsymbol{A}^{-1}\,. \tag{2.22}$$

Singly diagonally implicit Runge-Kutta methods

The stage equations (2.21a) read

$$z_i = h \sum_{j=1}^{i-1} a_{i,j}\, f(T_j\,,\, y_n + z_j) + h\,\gamma\, f(T_i\,,\, y_n + z_i)\,. \tag{2.23}$$

The nonlinear systems of equations (2.23) are solved in succession for each stage $i = 1, \ldots, s$ by simplified Newton iterations of the form

$$\begin{aligned} \mathbf{R}\,(\gamma, t_n, y_n) \cdot \Delta z_i^{[k]} &= z_i^{[k]} - h \sum_{j=1}^{i-1} a_{i,j}\, f(T_j\,,\, y_n + z_j) \\ z_i^{[k+1]} &= z_i^{[k]} - \Delta z_i^{[k]}\,, \quad k = 0, 1, \cdots\,. \end{aligned} \tag{2.24}$$

The same matrix is shared for all iterations and all stages, so that only one LU decomposition of $\mathbf{R}$ is performed in each time step.

Fully implicit Runge-Kutta methods

Fully implicit Runge-Kutta methods require the solution of the $ds \times ds$ nonlinear system (2.21a) [66]. With the compact notation

$$Z = \left[z_1^T \cdots z_s^T\right]^T\,, \quad F(Z) = \left[f^T(T_1, y_n + z_1) \cdots f^T(T_s, y_n + z_s)\right]^T\,, \tag{2.25}$$

where $Z, F(Z) \in \mathbb{R}^{ds}$, the nonlinear system (2.21a) can be written as

$$Z = \left(\boldsymbol{A} \otimes \boldsymbol{I}_{(d,d)}\right) \cdot F(Z)\,. \tag{2.26}$$

The system (2.26) is solved by simplified Newton iterations [47],

$$\begin{aligned} \overline{\mathbf{R}}_n \cdot \Delta Z^{[k]} &= Z^{[k]} - (h\,\boldsymbol{A} \otimes \boldsymbol{I}_{(d,d)}) \cdot F(Z^{[k]}) \\ Z^{[k+1]} &= Z^{[k]} - \Delta Z^{[k]}\,, \quad k = 0, 1, \cdots . \end{aligned} \tag{2.27}$$

Note that only the Jacobian at the beginning of the time step is used in Newton's iterations. Following [47], our implementation of the fully implicit s-stage Runge-Kutta method uses a transformation of the system (2.27) to a complex form such that the costly ds-dimensional real LU decomposition is replaced by d-dimensional LU decompositions of matrices of the form $\mathbf{R}\,(\lambda_i, t_n, y_n)$, where λ_i are the eigenvalues of $\mathbf{A}$. For the 3-stage methods implemented in FATODE the coefficient matrices $\mathbf{A}$ have one real and two complex conjugate eigenvalues, which leads to solving one real and one complex d-dimensional systems.

Rosenbrock methods

For implementation purpose, we use the alternative formulation [51] of the formula (2.6)

$$T_i = t_n + \alpha_i h\,, \quad Y_i = y_n + \sum_{j=1}^{i-1} \alpha_{i,j}\, k_j\,, \tag{2.28a}$$

$$\widetilde{\mathbf{R}}\,(\gamma, t_n, y_n) \cdot k_i = f\,(T_i, Y_i) + \sum_{j=1}^{i-1} \frac{c_{i,j}}{h}\, k_j + h\,\gamma_i\, f_t\,(t_n, y_n)\,, \tag{2.28b}$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} m_i\, k_i\,, \tag{2.28c}$$

where $\boldsymbol{a} = [a_{i,j}]_{1\le i,j\le s}$, $\boldsymbol{c} = [c_{i,j}]_{1\le i,j\le s}$, $\boldsymbol{m} = [m_i]_{1\le i\le s}$, are defined by

$$\boldsymbol{a} = \boldsymbol{\alpha} \cdot \boldsymbol{\gamma}^{-1}\,, \quad \boldsymbol{c} = diag(\gamma^{-1}) - \boldsymbol{\gamma}^{-1}\,, \quad \boldsymbol{m} = \boldsymbol{\gamma}^{-T} \cdot \boldsymbol{b}\,. \tag{2.29}$$

At each stage (2.28b) the solution of a linear system of dimension $d \times d$ is required. The same matrix $\widetilde{\mathbf{R}}$ is shared by all the stages and one LU decomposition per step is required.

2.5.2 FATODE implementation of tangent linear model integration

Tangent linear models are derived for direct sensitivity analysis with each of the families of methods in FATODE. Highly efficient implementations are obtained by re-using the LU decompositions from the forward solution on the sensitivity equations [62].

Tangent linear Runge-Kutta methods

A tangent linear Runge-Kutta (2.4) method reads

$$Y_i = y_n + h \sum_{j=1}^{s} a_{i,j} f(T_j, Y_j), \qquad \dot{Y}_i = \dot{y}_n + h \sum_{j=1}^{s} a_{i,j}\, \boldsymbol{f_y}(T_i\,,\, Y_i) \cdot \dot{Y}_i, \tag{2.30a}$$

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i f(T_i, Y_i), \qquad \dot{y}_{n+1} = \dot{y}_n + h \sum_{i=1}^{s} b_i\, \boldsymbol{f_y}(T_i\,,\, Y_i) \cdot \dot{Y}_i\,. \tag{2.30b}$$

Similar to the implementation of implicit forward integrators, we introduce the sensitivity stage variables $\dot{z}_i = \dot{Y}_i - \dot{y}_n$ and the sensitivity part becomes

$$\dot{z}_i - h \sum_{j=1}^{s} a_{i,j}\, \boldsymbol{f_y}(T_j\,,\, Y_j) \cdot \dot{z}_j = h \sum_{j=1}^{s} a_{i,j}\, \boldsymbol{f_y}(T_j\,,\, Y_j) \cdot \dot{y}_n\,, \quad i = 1, \cdots, s\,, \tag{2.31a}$$

$$\dot{y}_{n+1} = \dot{y}_n + \sum_{i=1}^{s} d_i\, \dot{z}_i\,. \tag{2.31b}$$

Using the compact notation (2.25) and the matrix (2.20c) the stage equations (2.31a) can be written as

$$\widehat{\mathbf{R}}_n \cdot \dot{Z} = \left(\boldsymbol{I}_{(sd,sd)} - \widehat{\mathbf{R}}_n \right) \cdot (\mathbf{1}_s \otimes \dot{y}_n)\ . \tag{2.32}$$

Explicit RK methods

For ERK methods the equations (2.30a) are solved successively for each stage $i = 1, \cdots, s$, using

$$\dot{Y}_1 = \dot{y}_n\,; \quad \dot{Y}_i = \dot{y}_n + h \sum_{j=1}^{i-1} a_{i,j}\, \boldsymbol{f_y}(T_j\,,\, Y_j) \cdot \dot{Y}_j\,, \quad i = 2, \cdots, s\,.$$

Singly diagonally implicit Runge-Kutta methods

For SDIRK methods the system (2.31a) reduces to s independent d-dimensional linear systems that are solved successively for each stage $i = 1, \cdots, s$

$$\mathbf{R}\,(\gamma, T_i, Y_i) \cdot \dot{z}_i \;\; = \;\; h \sum_{j=1}^{i-1} a_{i,j}\, \boldsymbol{f_y}(T_j\,,\, Y_j) \cdot (\dot{y}_n + \dot{z}_j) + h\,\gamma\, \boldsymbol{f_y}(T_i\,,\, Y_i) \cdot \dot{y}_n\,.$$

FATODE allows users to choose to solve the linear system (2.33) directly at the expense of an additional LU decomposition of the matrix $\mathbf{R}\,(\gamma, T_i, Y_i)$ per stage, or to apply simplified

Newton iterations of the form

$$\begin{aligned} \mathbf{R}\left(\gamma, t_n, y_n\right) \cdot \Delta \dot{z}_i^{[m]} &= \dot{z}_i^{[m]} - h \sum_{j=1}^{i} a_{i,j}\, \boldsymbol{f_y}(T_j\,,\, Y_j) \cdot (\dot{y}_n + \dot{z}_j^{[m]}) \\ \dot{z}_i^{[m+1]} &= \dot{z}_i^{[m]} - \Delta \dot{z}_i^{[m]}\,, \quad m = 0, 1, \cdots . \end{aligned} \tag{2.33}$$

The LU decomposition of the matrix $\mathbf{R}\left(\gamma, t_n, y_n\right)$ is also necessary in forward integration. Equations (2.33) re-use the LU decomposition which is available after the equations (2.24) are calculated in each step. FATODE controls the iteration number, and possibly the step size, such that the iteration error in (2.33) is smaller than the local truncation error at the current step. When (2.33) is used the accuracy of the sensitivity coefficients is of the same order as the local truncation error. The reuse of the forward LU factorization can save considerable CPU time.

Fully implicit Runge-Kutta methods

For fully implicit Runge-Kutta methods two options are available for solving the system (2.32). One is to construct the $ds \times ds$ linear system (2.32) explicitly and solve it directly by factorizing the matrix $\widehat{\mathbf{R}}_n$.

The other is to apply simplified Newton iterations of the form

$$\begin{aligned} \overline{\mathbf{R}}_n \cdot \Delta \dot{Z}^{[m]} &= \widehat{\mathbf{R}}_n \cdot \left(\mathbf{1}_s \otimes \dot{y}_n + \dot{Z}^{[m]}\right) - \mathbf{1}_s \otimes \dot{y}_n \\ \dot{Z}^{[m+1]} &= \dot{Z}^{[m]} - \Delta \dot{Z}^{[m]}\,, \quad m = 0, 1, \cdots . \end{aligned} \tag{2.34}$$

The matrix $\overline{\mathbf{R}}_n$ of the resulting $ds \times ds$ linear system is available from the forward solution process, i.e., the calculations of the equations (2.27). The real and complex LU decompositions can be reused. According to our experience, the second option is usually more efficient than the first one for large systems.

Rosenbrock methods

The tangent linear Rosenbrock method consists of the formula formula (2.28) plus the sensitivity part, which is obtained by differentiating the formula (2.28). In each step we solve the combined set of equations

$$\widetilde{\mathbf{R}}\left(h\gamma, t_n, y_n\right) \cdot k_i = f\left(T_i, Y_i\right) + \sum_{j=1}^{i-1} \frac{c_{i,j}}{h}\, k_j + h\gamma_i k_i f_t(t_n, y_n)\,, \tag{2.35a}$$

$$\widetilde{\mathbf{R}}\left(h\gamma, t_n, y_n\right) \cdot \dot{k}_i = \boldsymbol{f_y}\left(T_i, Y_i\right) \cdot \left(\dot{y}_n + \sum_{j=1}^{i-1} a_{i,j} \dot{k}_j\right) + \sum_{j=1}^{i-1} \frac{c_{i,j}}{h}\, \dot{k}_j \tag{2.35b}$$

$$+ (\dot{y}_n \cdot \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{y}}(t_n,\, y_n)) \cdot k_i + h\,\gamma_i\, f_{y,t}(t_n,\, y_n) \cdot \dot{y}_n\,,$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} m_i\, k_i, \tag{2.35c}$$

$$\dot{y}_{n+1} = \dot{y}_n + \sum_{i=1}^{s} m_i\, \dot{k}_i\,. \tag{2.35d}$$

The stage vectors $\dot{k}_i$ are obtained in succession by solving a sequence of linear systems, all of which re-use the LU decomposition of $\widetilde{\mathbf{R}}\,(h\gamma, t_n, y_n)$ performed in (2.28). Formula (2.35b) involves the Hessian tensor $\boldsymbol{f}_{\boldsymbol{y},\boldsymbol{y}}(t_n,\, y_n)$. In practice, an analytical Hessian tensor is difficult to obtain, and its evaluation is costly in both CPU time and memory storage. Note that the above equation only needs the product $(\dot{y}_n \cdot \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{y}}(t_n,\, y_n)) \cdot k_i$. Such terms can be obtained efficiently using automatic differentiation [60, 67] twice, in forward over reverse mode.

2.5.3 FATODE implementation of discrete adjoint model integration: sensitivities with respect to initial conditions

FATODE implements discrete adjoints of all the methods. Such discrete adjoints have good theoretical properties, in the sense that they are consistent discretizations of the adjoint ODE [57, 58]. This section discusses the adjoint sensitivities with respect to initial conditions.

The discrete adjoint Runge-Kutta method [68] solving the discrete adjoint equations (2.18) reads

$$u_i = h\, \boldsymbol{f}_{\boldsymbol{y}}^T\,(T_i, Y_i) \cdot \left(b_i \lambda_{n+1} + \sum_{j=1}^{s} a_{j,i} u_j \right)\,, \quad i = s, \ldots, 1\,, \tag{2.36a}$$

$$\lambda_n = \lambda_{n+1} + \sum_{j=1}^{s} u_j\,. \tag{2.36b}$$

The stage equations (2.36a) form a $ds \times ds$ linear system involving the transpose of matrix (2.20c):

$$\begin{aligned} U &= \left[u_1^T \cdots u_s^T\right]^T\,, \\ \widehat{\mathbf{R}}_n^T \cdot U &= h\, \left[b_1 \lambda_{n+1}^T\, \boldsymbol{f}_{\boldsymbol{y}}\,(T_1, Y_1) \cdots b_s \lambda_{n+1}^T\, \boldsymbol{f}_{\boldsymbol{y}}\,(T_s, Y_s)\right]^T\,. \end{aligned} \tag{2.37}$$

For $b_i \neq 0$ one can rewrite (2.36) as another Runge-Kutta method [68] applied to the adjoint

ODE

$$
\begin{aligned}
\ell_i &= \boldsymbol{f}_{\boldsymbol{y}}^T\Big(t_{n+1} - \bar{c}_i h,\, Y_{s+1-i}\Big) \cdot \left(\lambda^{n+1} + h \sum_{j=1}^{s} \bar{a}_{i,j} \ell_j\right), \quad i = s, \ldots, 1\,, \\
\lambda^n &= \lambda^{n+1} + h \sum_{i=1}^{s} \bar{b}_i\, \ell_i\,, \\
\text{where} \qquad & \bar{b}_i = b_{s+1-i}\,, \quad \bar{c}_i = 1 - c_{s+1-i}\,, \quad \bar{a}_{i,j} = \frac{a_{s+1-j,s+1-i} \cdot b_{s+1-j}}{b_{s+1-i}}\,.
\end{aligned}
\tag{2.38}
$$

The method $(\bar{\boldsymbol{A}}, \bar{b}, \bar{c})$ is called the *formal adjoint* of the Runge-Kutta method $(\boldsymbol{A}, b, c)$ [57]. The formal adjoint of Radau-2A is Radau-1A, vice versa. Lobatto-3C, Gauss and SDIRK-3a are formally self-adjoint.

Explicit Runge-Kutta methods

The stage equations (2.36a) are solved in succession for stages s down to 1:

$$
\begin{aligned}
u_s &= h\, b_s\, \boldsymbol{f}_{\boldsymbol{y}}^T(T_s,\, Y_s)\, \lambda^{n+1}\,, \\
u_i &= h\, \boldsymbol{f}_{\boldsymbol{y}}^T(T_i,\, Y_i) \cdot \left(b_i\, \lambda^{n+1} + \sum_{j=i+1}^{s} a_{j,i}\, u_j\right), \quad i = s-1, \ldots, 1\,.
\end{aligned}
\tag{2.39}
$$

Each stage i requires the computation of the Jacobian $\boldsymbol{f}_{\boldsymbol{y}}(T_i, Y_i)$, forming the vector $b_i\, \lambda^{n+1} + \sum_{j=i+1}^{s} a_{j,i} u_j$ from previously computed stages $u_s \ldots u_{i+1}$, and performing one Jacobian vector product.

Singly diagonally implicit Runge-Kutta methods

For SDIRK methods the s stages of the system (2.36a) are solved successively from the last stage to the first. Each stage requires the solution of a different linear system:

$$
\mathbf{R}(\gamma, T_i, Y_i) \cdot u_i = h\, \boldsymbol{f}_{\boldsymbol{y}}^T(T_i, Y_i) \cdot \left(b_i\, \lambda^{n+1} + \sum_{j=i+1}^{s} a_{j,i} u_j\right), \quad i = s, \ldots, 1\,. \tag{2.40}
$$

FATODE offers two options: to form and solve one linear system (2.40) per stage, or to employ simplified Newton iterations of the form (2.33) and re-use the LU decomposition of $\mathbf{R}(\gamma, t_n, y_n)$ for all stages. When the iterative approach is used the accuracy of the gradients is of the same order as the local truncation error. Considerable CPU time can be saved if checkpointing the forward LU factorization is feasible from a storage perspective.

Fully implicit Runge-Kutta methods

For the fully implicit Runge-Kutta methods the $ds \times ds$ system (2.37) is fully coupled. FATODE offers two approaches to solve it. The first is to build and solve directly (2.37) via a $ds \times ds$ LU decomposition of $\widehat{\mathbf{R}}_n$. The second approach uses simplified Newton iterations of the form (2.32), where $\widehat{\mathbf{R}}_n$ is replaced by $\overline{\mathbf{R}}_n$ in (2.37), and the transformation to real and complex systems is performed. The real and complex LU factorizations associated with the matrix $\overline{\mathbf{R}}_n$ are re-used in all iterations. These factorizations are computed during the forward solution, and in principle they can be checkpointed. The tradeoff between the size of the LU factorizations to store and the time needed to recompute them will determine the best strategy.

Rosenbrock methods

The discrete Rosenbrock adjoint [20] reads

$$\widetilde{\mathbf{R}}^T\left(h\gamma, t_n, y_n\right) \cdot u_i = m_i\, \lambda_{n+1} + \sum_{j=i+1}^{s} \left(a_{j,i}\, v_j + \frac{c_{j,i}}{h}\, u_j\right) , \tag{2.41a}$$

$$v_i = \boldsymbol{f}_{\boldsymbol{y}}^T(T_i\,,\, Y_i) \cdot u_i\,, \quad i = s, \ldots, 1\,, \tag{2.41b}$$

$$\lambda_n = \lambda_{n+1} + \sum_{i=1}^{s} \left(u_i \cdot \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{y}}(t_n, y_n)\right) \cdot k_i + h \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{t}}^T(t_n, y_n) \cdot \sum_{i=1}^{s} \gamma_i u_i + \sum_{i=1}^{s} v_i\,. \tag{2.41c}$$

The linear system (2.41a) can be solved directly at each stage. Users have to supply a routine for calculating the term $(u_i \cdot \boldsymbol{f}_{\boldsymbol{y},\boldsymbol{y}}(t_n,\, y_n)) \cdot k_i$, whose meaning is explained in Appendix 2.2.2. Automatic differentiation tools like TAMC [67] provide considerable help: the product between the Hessian transposed times vector can be obtained by two consecutive runs of TAMC in forward mode.

2.5.4 Discrete adjoint sensitivities with respect to parameters: general approach

Consider a numerical solution of (2.11):

$$\begin{aligned} y_{n+1} &= \Phi^n\left(y_n, p_n\right), \\ p_{n+1} &= p_n, \\ q_{n+1} &= q_n + \Omega^n\left(y_n, p_n\right), \quad n = 0, \ldots, N-1\,, \end{aligned} \tag{2.42}$$

together with the numerical evaluation of the cost function (2.12)

$$\Psi = g\left(y_N, p\right) + q_N\,. \tag{2.43}$$

Replacing $p_n = p$ for all n in (2.43) leads to the *discrete forward model*:

$$y_{n+1} = \Phi^n\left(y_n, p\right) \; ; \quad q_{n+1} = q_n + \Omega^n\left(y_n, p\right), \quad n = 0, \ldots, N-1 \,. \tag{2.44}$$

Differentiating (2.44) in the direction $\dot{p}$ yields the *discrete tangent linear model*:

$$\begin{aligned} \dot{y}_0 &= 0, \quad \dot{p}_0 = \dot{p}, \quad \dot{q}_0 = 0 \\ \dot{y}_{n+1} &= \Phi_y^n\left(y_n, p_n\right) \cdot \dot{y}_n + \Phi_p^n\left(y_n, p_n\right) \cdot \dot{p}_n, \\ \dot{p}_{n+1} &= \dot{p}_n, \\ \dot{q}_{n+1} &= \dot{q}_n + \Omega_y^n\left(y_n, p_n\right) \cdot \dot{y}_n + \Omega_p^n\left(y_n, p_n\right) \cdot \dot{p}_n. \end{aligned} \tag{2.45}$$

Replacing $p_n = p$ and $\dot{p}_n = \dot{p}$ for all n leads to (2.46)

$$\begin{aligned} \dot{y}_0 &= 0\,, \quad \dot{q}_0 = 0\,, \\ \dot{y}_{n+1} &= \Phi_y^n\left(y_n, p\right) \cdot \dot{y}_n + \Phi_p^n\left(y_n, p\right) \cdot \dot{p}\,, \\ \dot{q}_{n+1} &= \dot{q}_n + \Omega_y^n\left(y_n, p\right) \cdot \dot{y}_n + \Omega_p^n\left(y_n, p\right) \cdot \dot{p}\,. \end{aligned} \tag{2.46}$$

From (2.43) we define the co-state variables at $t_N = t_F$

$$\begin{bmatrix} \lambda_N \\ \mu_N \\ \theta_N \end{bmatrix} = \left(\frac{d\Psi}{d[y_N, p_N, q_N]} \right)^T = \begin{bmatrix} g_y^T(y_N, p) \\ g_p^T(y_N, p) \\ 1 \end{bmatrix} . \tag{2.47}$$

The backwards in time evolution of the adjoint variables is governed by the discrete adjoint equations obtained by differentiating and transposing (2.43)

$$\begin{bmatrix} \lambda_n \\ \mu_n \\ \theta_n \end{bmatrix} = \begin{bmatrix} \Phi_y^n(y_n, p) & \Phi_p^n(y_n, p) & 0 \\ 0 & \boldsymbol{I} & 0 \\ \Omega_y^n(y_n, p) & \Omega_p^n(y_n, p) & \boldsymbol{I} \end{bmatrix}^T \begin{bmatrix} \lambda_{n+1} \\ \mu_{n+1} \\ \theta_{n+1} \end{bmatrix} , \quad n = N-1, \ldots, 1\,. \tag{2.48}$$

or

$$\begin{bmatrix} \lambda_i \\ \mu_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} \Phi_y^i(y_i, p) & 0 & \Omega_y^i(y_i, p) \\ \Phi_p^i(y_i, p) & \boldsymbol{I} & \Omega_p^i(y_i, p) \\ 0 & 0 & \Omega_z^i \end{bmatrix} \begin{bmatrix} \lambda_{i+1} \\ \mu_{i+1} \\ \theta_{i+1} \end{bmatrix} , \quad i = N-1, \ldots, 1.$$

or, equivalently,

$$\begin{aligned} \lambda_n &= \left(\Phi_y^n(y_n, p_n)\right)^T \cdot \lambda_{n+1} + \left(\Omega_y^n(y_n, p_n)\right)^T \cdot \theta_{n+1}\,, \\ \mu_n &= \mu_{n+1} + \left(\Phi_p^n(y_n, p)\right)^T \cdot \lambda_{n+1} + \left(\Omega_p^n(y_n, p)\right)^T \cdot \theta_{n+1}\,, \\ \theta_n &= \Omega_q^n(y_n, p, q_n)\, \theta_{n+1}\,. \end{aligned}$$

and finally

$$\begin{aligned}
\lambda_n &= \left(\Phi_y^n(y_n,p)\right)^T \lambda_{n+1} + \left(\Omega_y^n(y_n,p_n)\right)^T \theta_{n+1}\,, \\
\mu_n &= \mu_{n+1} + \left(\Phi_p^n(y_n,p_n)\right)^T \lambda_{n+1} + \left(\Omega_p^n(y_n,p_n)\right)^T \theta_{n+1}\,, \\
\theta_n &= \theta_{n+1}\,.
\end{aligned} \tag{2.49}$$

From (2.47) and (2.48) one infers that $\theta_n = 1$ for all n. Using this fact, a rearrangement of (2.48) leads to the *discrete adjoint equations*

$$\begin{aligned}
\lambda_N &= g_y^T(y_N,p)\,, \quad \mu_N = g_p^T(y_N,p)\,, \\
\lambda_n &= \left(\Phi_y^n(y_n,p)\right)^T \cdot \lambda_{n+1} + \left(\Omega_y^n(y_n,p)\right)^T\,, \\
\mu_n &= \mu_{n+1} + \left(\Phi_p^n(y_n,p)\right)^T \cdot \lambda_{n+1} + \left(\Omega_p^n(y_n,p)\right)^T\,, \quad n = N-1,\ldots,0\,.
\end{aligned} \tag{2.50}$$

The adjoint values at the initial time represent the sensitivities of the numerical cost function (2.43) with respect to the initial conditions and with parameters, respectively:

$$\left(\frac{\partial \Psi}{\partial y_0}\right)^T = \lambda_0\,, \quad \left(\frac{\partial \Psi}{\partial p}\right)^T = \mu_0\,. \tag{2.51}$$

For details on derivation see [57].

2.5.5 FATODE implementation of discrete adjoint model integration: sensitivities with respect to a vector of parameters

We now consider the case where the adjoint sensitivity is computed with respect to a time-independent vector of parameters $p \in \mathbb{R}^m$ which appears in the right hand side of (2.1). We consider a scalar quantity of interest having the general form (2.10) (with the number of outputs $o = 1$). As shown in Appendix 2.5.4 the numerical solution of (2.11) provides the discrete y_n and q_n. The discrete adjoint model equations calculate the adjoint variables λ_n and μ_n backward in time, such that

$$\lambda_N = \boldsymbol{g_y^T}(y_N,p)\,, \quad \mu_N = \boldsymbol{g_p^T}(y_N,p)\,; \quad \lambda_0 = (\partial\Psi/\partial y_0)^T\,, \quad \mu_0 = (\partial\Psi/\partial p)^T\,. \tag{2.52}$$

For details on derivation see [57] and Appendix 2.5.4.

Runge-Kutta methods

Consider the Runge-Kutta method (2.4) applied to the extended ODE system (2.11)

$$Y_i = y_n + h\sum_{j=1}^{s} a_{i,j}\, f\left(T_j\,,\, Y_j\,,\, p\right)\,, \quad i = 1,\ldots,s\,, \tag{2.53a}$$

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j \, f\left(T_j, Y_j, p\right) , \tag{2.53b}$$

$$q_{n+1} = q_n + h \sum_{j=1}^{s} b_j \, r\left(T_j, Y_j, p\right) . \tag{2.53c}$$

Note that, since r does not depend on q, there is no need to compute the stage values for the quadrature variable.

Using the extended co-state vector and the extended Jacobian (2.8), the discrete adjoint (2.48) of a Runge-Kutta method is

$$\begin{aligned} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} &= h \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{y}}^T(T_i, Y_i, p) & \mathbf{0}_{(d,m)} & \boldsymbol{r}_{\boldsymbol{y}}^T(T_i, Y_i, p) \\ \boldsymbol{f}_{\boldsymbol{p}}^T(T_i, Y_i, p) & \mathbf{0}_{(m,m)} & \boldsymbol{r}_{\boldsymbol{p}}^T(T_i, Y_i, p) \\ \mathbf{0}_{(1,d)} & \mathbf{0}_{(1,m)} & 0_{(1,1)} \end{bmatrix} \\ &\quad \cdot \left(b_i \begin{bmatrix} \lambda_{n+1} \\ \mu_{n+1} \\ \theta_{n+1} \end{bmatrix} + \sum_{j=1}^{s} a_{j,i} \begin{bmatrix} u_j \\ v_j \\ w_j \end{bmatrix} \right) , \quad i = s, \ldots, 1 , \end{aligned} \tag{2.54a}$$

$$\begin{bmatrix} \lambda_n \\ \mu_n \\ \theta_n \end{bmatrix} = \begin{bmatrix} \lambda_{n+1} \\ \mu_{n+1} \\ \theta_{n+1} \end{bmatrix} + \sum_{j=1}^{s} \begin{bmatrix} u_j \\ v_j \\ w_j \end{bmatrix} . \tag{2.54b}$$

From the last equation of (2.54a) we see that $w_i = 0$ for all i, and from (2.54b) we infer that $\theta_n = \theta_{n+1} = \cdots = \theta_N = 1$. The discrete adjoint Runge-Kutta method (2.54) can be rewritten as

$$u_i = h \, \boldsymbol{f}_{\boldsymbol{y}}^T(T_i, Y_i, p) \cdot \left(b_i \, \lambda_{n+1} + \sum_{j=1}^{s} a_{j,i} \, u_j \right) + h \, b_i \, \boldsymbol{r}_{\boldsymbol{y}}^T(T_i, Y_i, p) , \tag{2.55a}$$

$$\begin{aligned} v_i &= h \, \boldsymbol{f}_{\boldsymbol{p}}^T(T_i, Y_i, p) \cdot \left(b_i \, \lambda_{n+1} + \sum_{j=1}^{s} a_{j,i} \, u_j \right) \\ &\quad + h \, b_i \, \boldsymbol{r}_{\boldsymbol{p}}^T(T_i, Y_i, p) , \quad i = s \ldots 1 , \end{aligned} \tag{2.55b}$$

$$\lambda_n = \lambda_{n+1} + \sum_{j=1}^{s} u_j , \tag{2.55c}$$

$$\mu_n = \mu_{n+1} + \sum_{j=1}^{s} v_j . \tag{2.55d}$$

The stages u_i are obtained by solving the system in a similar way as solving system (2.39) and the implementation differs between SDIRK and fully implicit 3-stage Runge-Kutta method. Then v_i can be readily obtained from the right-hand side calculation.

Sensitivities with respect to initial conditions are obtained by setting all derivatives with respect to parameters to zero in (2.55), to obtain (2.36).

Rosenbrock methods

Applying the Rosenbrock method (2.28) to the extended system (2.11) gives the following formula for evaluating the quadrature term in the cost functional (2.10):

$$\widehat{k}_i = h\gamma \left(r(T_i, Y_i) + \sum_{j=1}^{i-1} \frac{c_{i,j}}{h} \widehat{k}_j + h\gamma_i r_t + \boldsymbol{r_y} \cdot k_i \right) , \tag{2.56a}$$

$$q_{n+1} = q_n + \sum_{i=1}^{s} m_i \widehat{k}_i \ . \tag{2.56b}$$

Equation (2.56) is evaluated simultaneously with the ODE integration.

The discrete adjoint (2.48) of a Rosenbrock method is

$$\begin{aligned}
&\begin{bmatrix} \frac{\boldsymbol{I}_{(d,d)}}{h\gamma} - \boldsymbol{f_y}^T(t_n, y_n, p) & 0 & -\boldsymbol{r_y}^T(t_n, y_n, p) \\ -\boldsymbol{f_p}^T(t_n, y_n, p) & \frac{\boldsymbol{I}_{(m,m)}}{h\gamma} & -\boldsymbol{r_p}^T(t_n, y_n, p) \\ 0 & 0 & \frac{1}{h\gamma} \end{bmatrix} \cdot \begin{bmatrix} u_i \\ \overline{u}_i \\ \widehat{u}_i \end{bmatrix} = m_i \begin{bmatrix} \lambda_{n+1} \\ \mu_{n+1} \\ \theta_{n+1} \end{bmatrix} \\
&\qquad + \sum_{j=i+1}^{s} \left(a_{j,i} \begin{bmatrix} v_j \\ \overline{v}_j \\ \widehat{v}_j \end{bmatrix} + \frac{c_{j,i}}{h} \begin{bmatrix} u_j \\ \overline{u}_j \\ \widehat{u}_j \end{bmatrix} \right) , \\
&\begin{bmatrix} v_i \\ \overline{v}_i \\ \widehat{v}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{f_y}^T(T_i, Y_i, p) & 0 & \boldsymbol{r_y}^T(T_i, Y_i, p) \\ \boldsymbol{f_p}^T(T_i, Y_i, p) & 0 & \boldsymbol{r_p}^T(T_i, Y_i, p) \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u_i \\ \overline{u}_i \\ \widehat{u}_i \end{bmatrix} , \quad i = s, s-1, \cdots, 1 , \\
&\begin{bmatrix} \lambda_n \\ \mu_n \\ \theta_n \end{bmatrix} = \begin{bmatrix} \lambda_{n+1} \\ \mu_{n+1} \\ \theta_{n+1} \end{bmatrix} + \sum_{i=1}^{s} \left(\begin{bmatrix} u_i \\ \overline{u}_i \\ \widehat{u}_i \end{bmatrix} \cdot \widetilde{\boldsymbol{H}} \right) \cdot \begin{bmatrix} k_i \\ \overline{k}_i \\ \widehat{k}_i \end{bmatrix} \\
&\qquad + h \begin{bmatrix} \boldsymbol{f_{y,t}}^T(t_n, y_n, p) & 0 & \boldsymbol{r_{y,t}}^T(t_n, y_n, p) \\ \boldsymbol{f_{p,t}}^T(t_n, y_n, p) & 0 & \boldsymbol{r_{p,t}}^T(t_n, y_n, p) \\ 0 & 0 & 0 \end{bmatrix} \cdot \sum_{i=1}^{s} \gamma_i \begin{bmatrix} u_i \\ \overline{u}_i \\ \widehat{u}_i \end{bmatrix} + \sum_{i=1}^{s} \begin{bmatrix} v_i \\ \overline{v}_i \\ \widehat{v}_i \end{bmatrix} .
\end{aligned}$$

Using the derivative notation in Appendix 2.2.2, this equation can be written component by component as follows:

$$\begin{aligned}
&\frac{1}{h\gamma} \widehat{u}_i = m_i \, \theta_{n+1} + \sum_{j=i+1}^{s} \left(a_{j,i} \, \widehat{v}_j + \frac{c_{j,i}}{h} \, \widehat{u}_j \right) , \\
&\left(\frac{\boldsymbol{I}_{(d,d)}}{h\gamma} - \boldsymbol{f_y}^T(t_n, y_n, p) \right) u_i = \boldsymbol{r_y}^T(t, y, p) \, \widehat{u}_i + m_i \, \lambda_{n+1} + \sum_{j=i+1}^{s} \left(a_{j,i} \, v_j + \frac{c_{j,i}}{h} \, u_j \right) , \\
&\frac{1}{h\gamma} \overline{u}_i = \boldsymbol{f_p}^T(t_n, y_n, p) \, u_i + \boldsymbol{r_p}^T(t_n, y_n, p) \, \widehat{u}_i + m_i \, \mu_{n+1} + \sum_{j=i+1}^{s} \left(a_{j,i} \, \overline{v}_j + \frac{c_{j,i}}{h} \, \overline{u}_j \right) ,
\end{aligned}$$

$$
\begin{aligned}
v_i &= \boldsymbol{f_y}^T(T_i, Y_i, p)\, u_i + \boldsymbol{r_y}^T(T_i, Y_i, p)\, \widehat{u}_i\,, \\
\overline{v}_i &= \boldsymbol{f_p^T}(T_i, Y_i, p)\, u_i + \boldsymbol{r_p}^T(T_i, Y_i, p)\, \widehat{u}_i\,, \\
\widehat{v}_i &= 0\,, \\
\lambda_n &= \lambda_{n+1} + \sum_{i=1}^{s} \left((u_i \cdot \boldsymbol{f_{y,y}}) \cdot k_i + (u_i \cdot \boldsymbol{f_{y,p}}) \cdot \overline{k}_i + (\widehat{u}_i \cdot \boldsymbol{r_{y,y}}) \cdot k_i + (\widehat{u}_i \cdot \boldsymbol{r_{y,p}}) \cdot \overline{k}_i \right)\,, \\
&\quad + h\, \boldsymbol{f}_{\boldsymbol{y,t}}^T(t, y, p) \cdot \sum_{i=1}^{s} \gamma_i u_i + h\, \boldsymbol{r}_{\boldsymbol{y,t}}^T(t, y, p) \cdot \sum_{i=1}^{s} \gamma_i \widehat{u}_i + \sum_{i=1}^{s} v_i\,, \\
\mu_n &= \mu_{n+1} + \sum_{i=1}^{s} \left((u_i \cdot \boldsymbol{f_{p,y}}) \cdot k_i + (u_i \cdot \boldsymbol{f_{p,p}}) \cdot \overline{k}_i + (\widehat{u}_i \cdot \boldsymbol{r_{p,y}}) \cdot k_i + (\widehat{u}_i \cdot \boldsymbol{r_{p,p}}) \cdot \overline{k}_i \right)\,, \\
&\quad + h\, \boldsymbol{f}_{\boldsymbol{p,t}}^T(t, y, p) \cdot \sum_{i=1}^{s} \gamma_i u_i + h\, \boldsymbol{r}_{\boldsymbol{p,t}}^T(t, y, p) \cdot \sum_{i=1}^{s} \gamma_i \widehat{u}_i + \sum_{i=1}^{s} \overline{v}_i\,, \\
\theta_n &= \theta_{n+1} + \sum_{i=1}^{s} \widehat{v}_i\,.
\end{aligned}
$$

This equation can be simplified using the following facts.

- Note that $\theta_n = 1$ for all n and $\widehat{v}_i = 0$ for all i, therefore the numbers $\widehat{u}_i$ can be computed by the simple recurrence

$$
\frac{1}{h\gamma}\,\widehat{u}_i \;=\; m_i + \sum_{j=i+1}^{s} \left(\frac{c_{j,i}}{h}\,\widehat{u}_j \right) \quad \Leftrightarrow \quad \widehat{u} = h\,\gamma\,\left(\mathbf{I}_{(s,s)} - \gamma\,\mathbf{c}^T \right)^{-1} \mathbf{m}\,. \tag{2.57}
$$

  Equations (2.57) and (2.29) lead to

$$
\widehat{u} = h\boldsymbol{b}\,.
$$

- Equation (2.28c)

$$
\begin{bmatrix} y_{n+1} \\ p_{n+1} \\ q_{n+1} \end{bmatrix} = \begin{bmatrix} y_n \\ p_n \\ q_n \end{bmatrix} + \sum_{i=1}^{s} m_i \begin{bmatrix} k_i \\ \overline{k}_i \\ \widehat{k}_i \end{bmatrix}
$$

  indicates that $\overline{k}_i = 0$ for all time steps since $p_{n+1} = p_n = p$. The vector k_i and $\widehat{k}_i$ are obtained from the solution of the extended form of forward integration (2.28b) and (2.56).

- The equation for calculating the quantity $\overline{u}$ is not needed to update the adjoint variables λ and μ so that the third equation can be omitted.

With the above simplifications, and using (2.9) and the derivative notation of Appendix 2.2.2, the discrete adjoint Rosenbrock method can be written in a component-wise manner

as follows:

$$
\begin{aligned}
&\frac{1}{h\gamma}\,\widehat{u}_i = m_i + \sum_{j=i+1}^{s}\left(\frac{c_{j,i}}{h}\,\widehat{u}_j\right)\\
&\left(\frac{\boldsymbol{I}_{(d,d)}}{h\gamma} - \boldsymbol{f_y^T}(t_n,y_n,p)\right)\,u_i = \widehat{u}_i\,\boldsymbol{r}_{\boldsymbol{y}}^T(t_n,y_n,p) + m_i\,\lambda_{n+1} + \sum_{j=i+1}^{s}\left(a_{j,i}\,v_j + \frac{c_{j,i}}{h}\,u_j\right)\\
&v_i = \boldsymbol{f_y^T}(T_i,Y_i,p)\cdot u_i + \widehat{u}_i\,\boldsymbol{r}_{\boldsymbol{y}}^T(T_i,Y_i,p)\\
&\overline{v}_i = \boldsymbol{f_p^T}(T_i,Y_i,p)\cdot u_i + \widehat{u}_i\,\boldsymbol{r}_{\boldsymbol{p}}^T(T_i,Y_i,p)\\
&\lambda_n = \lambda_{n+1} + \sum_{i=1}^{s}\Big((u_i\cdot\boldsymbol{f_{y,y}})\cdot k_i + (\widehat{u}_i\cdot\boldsymbol{r_{y,y}})\cdot k_i\Big)\\
&\qquad +h\,\boldsymbol{f}_{\boldsymbol{y,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i u_i + h\,\boldsymbol{r}_{\boldsymbol{y,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i\widehat{u}_i + \sum_{i=1}^{s}v_i\\
&\quad = \lambda_{n+1} + \sum_{i=1}^{s}\Big((\boldsymbol{f_{y,y}}\cdot k_i)^T\cdot u_i + \widehat{u}_i\,\left(\boldsymbol{r_{y,y}}\cdot k_i\right)^T\Big)\\
&\qquad +h\,\boldsymbol{f}_{\boldsymbol{y,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i u_i + h\,\boldsymbol{r}_{\boldsymbol{y,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i\widehat{u}_i + \sum_{i=1}^{s}v_i\\
&\mu_n = \mu_{n+1} + \sum_{i=1}^{s}\Big((u_i\cdot\boldsymbol{f_{p,y}})\cdot k_i + (\widehat{u}_i\cdot\boldsymbol{r_{p,y}})\cdot k_i\Big)\\
&\qquad +h\,\boldsymbol{f}_{\boldsymbol{p,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i u_i + h\,\boldsymbol{r}_{\boldsymbol{p,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i\widehat{u}_i + \sum_{i=1}^{s}\overline{v}_i\\
&\quad = \mu_{n+1} + \sum_{i=1}^{s}\Big((\boldsymbol{f_{p,y}}\cdot k_i)^T\cdot u_i + \widehat{u}_i\,\left(\boldsymbol{r_{p,y}}\cdot k_i\right)^T\Big)\\
&\qquad +h\,\boldsymbol{f}_{\boldsymbol{p,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i u_i + h\,\boldsymbol{r}_{\boldsymbol{p,t}}^T(t_n,y_n,p)\,\cdot\sum_{i=1}^{s}\gamma_i\widehat{u}_i + \sum_{i=1}^{s}\overline{v}_i\,.
\end{aligned}
$$

The result is

$$\widetilde{\mathbf{R}}^T(\gamma,t_n,y_n)\cdot u_i = h\,b_i\,\boldsymbol{r}_{\boldsymbol{y}}^T + m_i\,\lambda_{n+1} + \sum_{j=i+1}^{s}\left(a_{j,i}\,v_j + \frac{c_{j,i}}{h}\,u_j\right) \tag{2.58a}$$

$$v_i = \boldsymbol{f_y^T}(T_i,Y_i,p)\cdot u_i + h\,b_i\,\boldsymbol{r}_{\boldsymbol{y}}^T(T_i,Y_i,p) \tag{2.58b}$$

$$\overline{v}_i = \boldsymbol{f_p^T}(T_i,Y_i,p)\cdot u_i + h\,b_i\,\boldsymbol{r}_{\boldsymbol{p}}^T(T_i,Y_i,p) \tag{2.58c}$$

$$\begin{aligned}\lambda_n = \lambda_{n+1} &+ \sum_{i=1}^{s}(\boldsymbol{f_{y,y}}\cdot k_i)^T\cdot u_i + h\,\sum_{i=1}^{s}b_i\,\left(\boldsymbol{r_{y,y}}\cdot k_i\right)^T\\ &+h\,\boldsymbol{f}_{\boldsymbol{y,t}}^T\cdot\sum_{i=1}^{s}\gamma_i u_i + h^2\,\rho\,\boldsymbol{r}_{\boldsymbol{y,t}}^T + \sum_{i=1}^{s}v_i\end{aligned} \tag{2.58d}$$

$$\mu_n = \mu_{n+1} + \sum_{i=1}^{s} (\boldsymbol{f_{p,y}} \cdot k_i)^T \cdot u_i + h \sum_{i=1}^{s} b_i \, (\boldsymbol{r_{p,y}} \cdot k_i)^T \tag{2.58e}$$
$$+h \, \boldsymbol{f_{p,t}^T} \cdot \sum_{i=1}^{s} \gamma_i u_i + h^2 \, \rho \, \boldsymbol{r_{p,t}^T} + \sum_{i=1}^{s} \overline{v}_i \, ,$$

where $\rho = \sum_{i=1}^{s} \gamma_i \, b_i$ and all derivatives are evaluated at (t_n, y_n, p), unless their arguments are explicitly shown.

When all partial derivatives with respect to parameters are set to zero in (2.58) one obtains the discrete Rosenbrock adjoint (2.41), which calculates sensitivities with respect to initial conditions.

2.5.6 FATODE error estimation and step size control

Variable step size control is routinely adopted by general ODE solvers to control numerical errors and maximize efficiency. FATODE's forward integrators use estimates of the truncation error to decide whether to accept or reject the step, and to compute the next step size. The maximum number of integration steps before an unsuccessful return can be specified by the user.

For the ERK, SDIRK, and Rosenbrock methods the classical error estimators based on embedded solutions are implemented; they proved to work well in practice. Two different error estimation options are provided for fully implicit Runge-Kutta methods. One is the classical error estimation [47] which uses an embedded third order method based on an additional explicit stage. The embedded solution is

$$\hat{y}_{n+1} = y_n + h \left(\hat{b}_0 \, f(t_n, y_n) + \sum_{i=1}^{s} \hat{b}_i \, f(t_n + c_i \, h, y_n + z_i) \right) \tag{2.59}$$

where the increment vectors z_i are already computed in previous stages. The second estimator uses two additional stages: an explicit stage at the beginning of the time step and another SDIRK stage which re-uses the LU decomposition from the solution of the main integrator. The SDIRK stage reads

$$\hat{y}_{n+1} = y_n + h \left(\hat{b}_0 \, f(t_n, y_n) + \sum_{i=1}^{s} \hat{b}_i \, f(t_n + c_i h, y_n + z_i) + \gamma \, f(t_n + h, y_n + z_{s+1}) \right) .$$

The coefficients are chosen such that the order of consistency of the embedded solution $\hat{y}_{n+1}$ is $\hat{p} = p - 1$, where p is the order of y_{n+1}. The difference vector $Est = \hat{y}_{n+1} - y_{n+1}$ is used as a local error estimator. Our experience indicates that the SDIRK error estimator yields better results overall.

The local error test is performed as follows. Let $Tol_k = atol_k + rtol_k \cdot |y_{n+1,k}|$, where $atol$ and $rtol$ are the absolute and relative error tolerance specified by user, and $|y_{n+1,k}|$ is the absolute value of the k-th component of y_{n+1}. The relative and absolute error tolerances can be either vector or scalar (in which case the same tolerance values are used for all components k). The local error test takes the form [47]

$$Err = \sqrt{\frac{1}{d}\sum_{k=1}^{d}\left(\frac{Est_k}{Tol_k}\right)^2} < 1\,. \tag{2.60}$$

If the test passes the step size is accepted, otherwise it rejected and the step is recomputed. The new step size, for both accepted and rejected cases, is given by [47]

$$h_{\text{new}} = h_{\text{old}} \cdot \min\left(\mathcal{F}_{\max}, \max\left(\mathcal{F}_{\min}, \mathcal{F}_{\text{safe}} \cdot Err^{-1/(\hat{p}+1)}\right)\right)\,,$$

where $\mathcal{F}_{\max}$ is the upper bound on step increase factor, $\mathcal{F}_{\min}$ the lower bound on step decrease factor, and $\mathcal{F}_{\text{safe}}$ is a safety factor. The default values of these factors depend on the specific method. All of them can be changed by the user in the parameter settings. If the step size is rejected at the first step, the step increase factor $\mathcal{F}_{\max}$ is set to 1, and the step size is reduced by a factor of 10. Furthermore, the step size can be constrained by minimum ($h_{\min}$) and maximum ($h_{\max}$) values. The starting step size h_{start} can be specified by the user.

For the tangent linear model integration FATODE provides two options for controlling errors in the sensitivities. The first option is to use only the forward error estimates for step size control. The second option is to estimate the truncation errors for both the forward solution and the tangent linear model solution. The solution error is taken as the maximum between the forward truncation error and the truncation error of any column of the sensitivities. This solution error is then used to control the step size.

The discrete adjoint model integration traces the same sequence of steps as the forward integration, in reverse order. Therefore, the choice of the step sizes is completely determined during the forward integration, and the accuracy of the adjoint solution will depend on the error control performed during the forward run.

2.5.7 Computing derivatives required by FATODE

The implicit formulation of the stiff solvers, as well as the formulation of the tangent linear and adjoint methods, require the computation of various derivatives, summarized in Appendix 2.2.2. These derivatives include the Jacobians of the ODE function with respect to the state $\boldsymbol{f_y}$, e.g., in equations (2.20); gradients of the quadrature function $\boldsymbol{r_y}^T$, $\boldsymbol{r_p}^T$, e.g., in (2.55); Jacobian times vector products, e.g., $\boldsymbol{f_y}u$ in (2.30); transposed Jacobians times vector products, e.g., $\boldsymbol{f_y}^T u$ and $\boldsymbol{f_p}^T u$ in (2.58b)–(2.58c); Hessian times vector products, e.g., $(\boldsymbol{f_{y,y}}k)\,u$ in (2.35b) and $(\boldsymbol{f_{p,y}}k)^T\,u$, $(\boldsymbol{r_{p,y}}k)^T\,u$ in (2.58); and time derivatives of Jacobians

transposed times vectors, e.g., $\boldsymbol{f}_{\boldsymbol{p},\boldsymbol{t}}^T u$ in (2.58d). Among the methods implemented in FATODE, the Rosenbrock family requires the calculation of most derivatives, including Hessian vector products.

The derivatives supplied to FATODE can be obtained analytically, by finite differences, or by automatic differentiation. The errors in the derivative terms should be smaller than the local truncation error of the integrator, otherwise a loss of accuracy in the computed sensitivities will be experienced. Therefore utmost care must be exercised with the use of finite difference approximations. If analytical derivatives are not available, automatic differentiation tools like TAMC [67] can provide considerable help. For example, a Jacobian vector product is obtained by one TAMC run in forward mode, a transposed-Jacobian vector product by one TAMC run in reverse mode, and the product between the Hessian transposed times vector can be obtained by two consecutive runs of TAMC in forward mode.

2.5.8 Linear solvers in FATODE

The most computationally intensive part in solving large-scale ODE systems by implicit methods is the solution of linear systems at each step. Linear systems arise from the simplified Newton iterations applied to solve the nonlinear systems in case of fully implicit Runge-Kutta methods and SDIRK methods. For Rosenbrock methods, linear systems appear directly in the formula (2.6). In general, all implicit time stepping methods in FATODE require the solution of linear systems with matrices $\mathbf{R}$ or $\widetilde{\mathbf{R}}$ defined in (2.20). These matrices inherit the sparsity structure of the system Jacobian.

The best efficiency is achieved when taking advantage of the problem-specific characteristics. Consequently, FATODE was designed to allow users to provide their own linear solvers and sparse data structures. We have incorporated three direct methods in current version of FATODE. For dense systems calls to LAPACK [63] routines are provided. For large sparse systems, substantial memory and execution time benefits can be gained by calling the direct sparse solvers UMFPACK [64] and SuperLU [65] and representing the sparse matrices in a compressed column format. Interfaces to both these codes are provided.

All the relevant information is encapsulated in a linear algebra module, as explained in Section 2.4. The module contain interfaces to the following four generic routines: *LS_Init* (initialization and memory allocation required by the specific linear solver), *LS_Decomp* (LU decomposition), *LS_Solve* (solves the triangular systems by substitution), and *LS_Free* (frees the memory allocated and clears the objects created during the initialization stage). The time integrators make calls to these functions without having to consider the underlying solver details. The user can choose one of the linear solvers provided (LAPACK, UMFPACK, SuperLU), or can use them as templates and add a new linear solver to the module. For example adding an iterative solver requires to provide a data structure for the Jacobian, and a corresponding Jacobian-vector product routine; *LS_Decomp* remains empty, while the iterative solver is called from within *LS_Solve*. All the required code modifications are within

the linear algebra module.

2.6 Numerical experiments with nonstiff solvers on the two-dimensional shallow water equations

In this section we illustrate the capabilities of FATODE and evaluate the performance of each family of methods using a semi-discretized system of partial differential equations. We compare FATODE with the well established code CVODES within SUNDIALS [69] for the forward solution, as well as for the direct and adjoint sensitivity analysis. Note that CVODES is implemented in the C programming language while FATODE is written in Fortran. To reduce the influence of different compilers we use the same Fortran implementations of the right hand side function and its Jacobian, and call them from CVODES via C interfaces.

All the experiments are performed on a workstation with dual Intel Xeon E5-2630 CPUs (2.3GHz) running Fedora 17 (x86_64) Linux. PGI Fortran Version 12.10-5 and GCC Version 4.7.2 are used for compilation, with level O3 optimization. Unless stated otherwise, the default settings for parameters are used in all FATODE calls.

We consider the two-dimensional shallow water equations [70]

$$\begin{aligned}
&\frac{\partial}{\partial t} h + \frac{\partial}{\partial x}(uh) + \frac{\partial}{\partial y}(vh) = 0, \\
&\frac{\partial}{\partial t}(uh) + \frac{\partial}{\partial x}\left(u^2 + \frac{1}{2}gh^2\right) + \frac{\partial}{\partial y}(uvh) = 0, \\
&\frac{\partial}{\partial t}(vh) + \frac{\partial}{\partial t}(uvh) + \frac{\partial}{\partial y}\left(v^2 h + \frac{1}{2}gh^2\right) = 0,
\end{aligned} \tag{2.61}$$

where $u(t,x,y)$, $v(t,x,y)$ are the components of the velocity field, $h(t,x,y)$ is the fluid layer thickness, and g denotes the gravitational acceleration. The spatial domain is $\Omega = [-3,\, 3]^2$, and the simulation time interval is $[t_0, t_F] = [0, 0.02]$ time units. The spatial domain is covered by a grid of size 40×40, and third order upwind finite differences are used to perform the spatial discretization. This results in a large, non-stiff ODE system of dimension $40 \times 40 \times 3 = 4800$ which is solved by FATODE.

A reference solution of the ODE system is obtained by using LSODE [71], a well known but relatively slow ODE solver, with a very tight relative and absolute tolerances of 10^{-14}. The solution relative error is defined as

$$Err = \|y(t_F) - y_{\text{ref}}(t_F)\|_2 \,/\, \|y_{\text{ref}}(t_F)\|_2 \ , \tag{2.62}$$

where $y(t_F)$ is the numerical solution at the final step t_F, and $y_{\text{ref}}(t_F)$ is the reference solution at t_F. A similar metric is used for the relative sensitivity errors

$$Err = \|s - s_{\text{ref}}\|_2 \,/\, \|s_{\text{ref}}\|_2 \ . \tag{2.63}$$

where s is a numerical sensitivity vector. The reference values s_{ref} are computed using Adam-Moulton methods in CVODES [69] with relative and absolute tolerances of 10^{-14} for adjoint sensitivities, and with a relative tolerance of 10^{-9} and an absolute tolerance of 10^{-10} for tangent linear sensitivities.

We report below numerical results with the nonstiff integrators in FATODE and CVODES. Results with the stiff solvers in FATODE and CVODES on the shallow water test can be found in the Supplementary material [61, part 2.7].

2.6.1 Forward solution

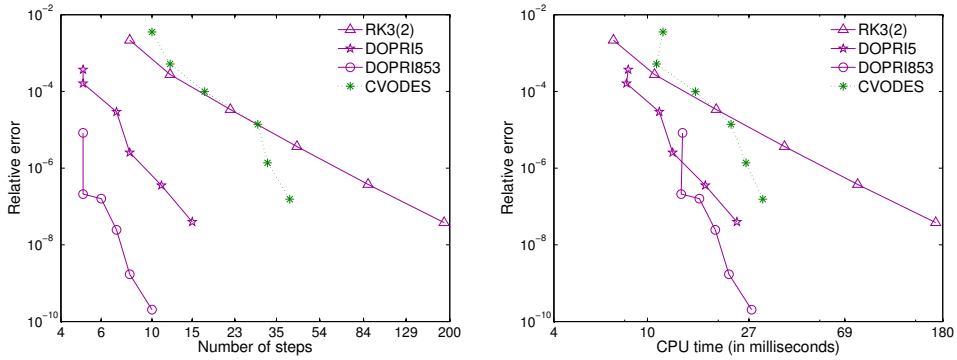
CVODES [69] is a the sensitivity analysis-enabled version of CVODE, which uses the Adams-Moulton methods for non-stiff ODE systems and the backward differentiation formulas (BDF methods) for stiff ODE systems. Both methods are implemented in a variable-order variable-step form. In our test, we employ the Adams-Moulton method in CVODE. We select three ERK methods of orders three, five, and eight in FATODE. The Gustafsson predictive error controller is used for all of them.

With each solver we vary both absolute and relative error tolerances from 10^{-2} to 10^{-7} to obtain solutions of different levels of accuracy (the absolute and relative error tolerances are equal to each other). The resulting work-precision diagrams shown in Figures 2.2(a) (error versus step size) and 2.2(b) (error versus CPU time). For the same level of accuracy, the ERK methods in FATODE take fewer steps than the Adams-Moulton method in CVODES, but the cost per step is higher. Nevertheless, FATODE's ERK methods of orders five and eight outperform CVODES, with the best overall CPU time being achieved by FATODE's order eight method.

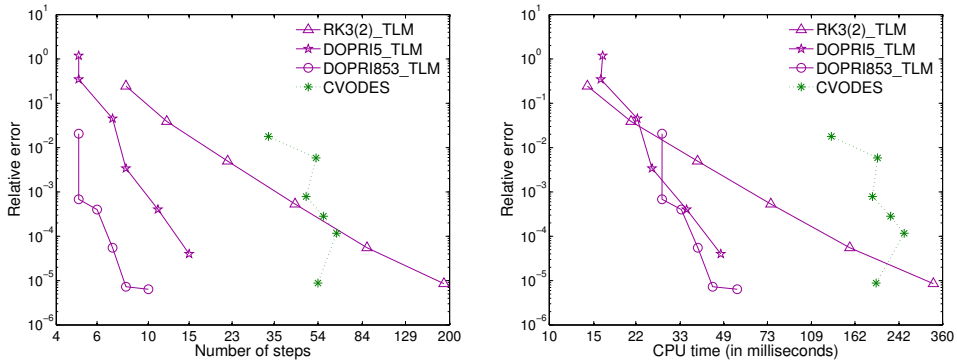
2.6.2 Direct sensitivity analysis

We now calculate the sensitivities of all solution components at the final time with respect to the initial value of the first solution component $\partial y_i(t_F)/\partial y_1(t_0)$, $i = 1 \dots d$, using tangent linear model integration.

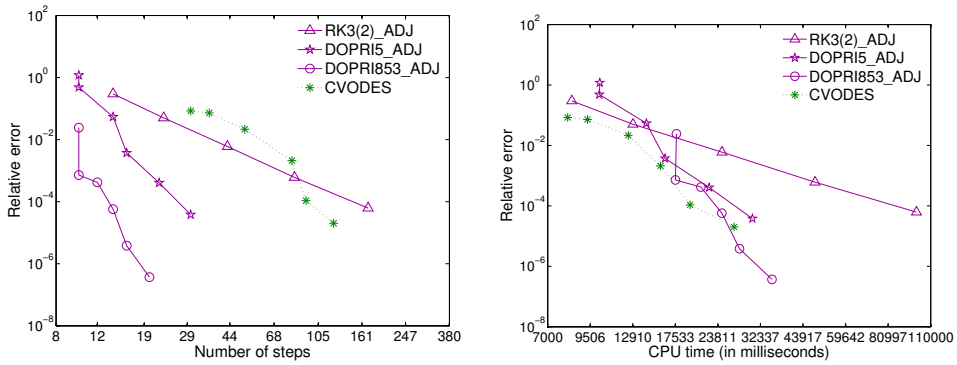
The performance results of nonstiff tangent linear integrators are shown in Figures 2.2(c) (error versus step size) and 2.2(d) (error versus CPU time). FATODE's ERK TLM methods of all orders are considerably more efficient than the nonstiff forward sensitivity solver in CVODES in terms of both CPU time and number of steps.



(a) ODE solution error vs. number of steps (b) ODE solution error vs. CPU time



(c) Forward sensitivity error vs. number of steps (d) Forward sensitivity error vs. CPU time



(e) Adjoint sensitivity error vs. number of steps (f) Adjoint sensitivity error vs. CPU time

Figure 2.2: ODE solution and sensitivity analysis of the shallow water equations (2.62) using nonstiff solvers. Comparison is made between three ERK methods in FATODE and the Adams-Moulton method in CVODE. Different points in each plot correspond to different tolerances levels in the range $10^{-2}, 10^{-3}, \ldots, 10^{-7}$ (with the absolute tolerances equal to the relative tolerances). The tangent linear models compute the sensitivity $\partial y_i(t_F)/\partial y_1(t_0),\ i = 1 \ldots d$, and the adjoint models compute the sensitivity $\partial y_1(t_F)/\partial y_i(t_0),\ i = 1 \ldots d$.

2.6.3 Adjoint sensitivity analysis

We next calculate the sensitivities of the first solution component at the final time with respect to all initial values $\partial y_1(t_F)/\ \partial y_i(t_0)$, $i = 1 \ldots d$, using adjoint model integration.

Work-precision diagrams for the adjoint explicit solvers are shown inFigures 2.2(e) (error versus step size) and 2.2(f) (error versus CPU time). The ERK adjoint methods in FATODE take fewer steps, but run slightly slower than the Adams-Moulton method in CVODES. This is due to the fact that the explicit Runge-Kutta method in FATODE requires more Jacobian evaluations and Jacobian-vector products during the adjoint backward run.

2.7 Numerical experiments with stiff solvers on the two dimensional shallow water problem

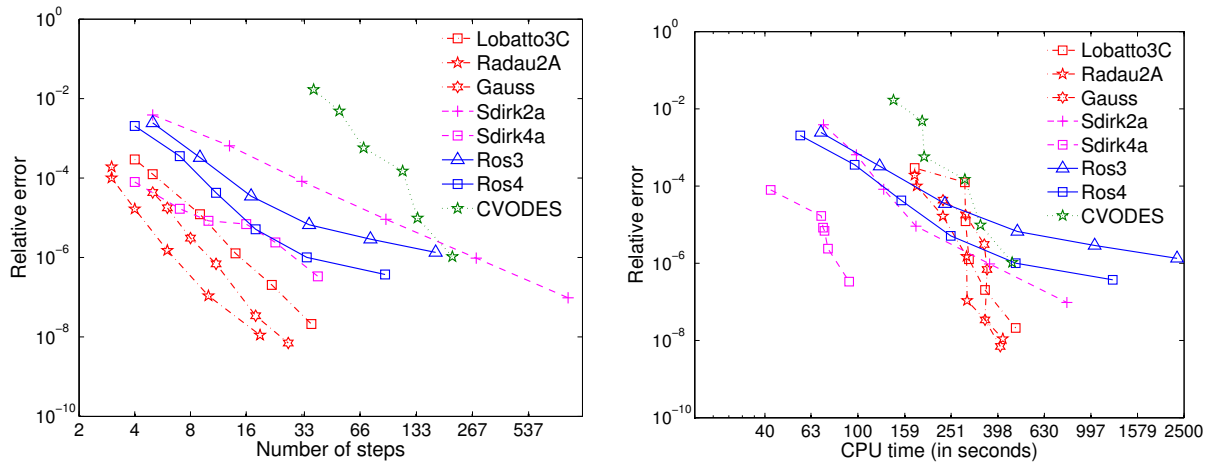
2.7.1 Forward solution

In our test, we choose the option of the BDF method for the comparison with implicit methods in FATODE. From each family of implicit integrators in FATODE we select several representative methods with different orders for the tests: Lobatto3C, Radau2A, and Gauss (fully implicit Runge-Kutta); Ros3 and Ros4 (Rosenbrock); Sdirk4a, Sdirk2a (singly diagonally implicit Runge-Kutta). The Gustafsson predictive error controller is used for all integrators. An additional SDIRK stage was used in the error estimator in the fully implicit Runge-Kutta integrator.

Since solution of linear systems dominates the computational cost of implicit integration, especially for large-scale ODE systems, it is necessary to use the same linear solvers for both CVODE and FATODE for a fair performance comparison. In our comparison experiments we use the direct linear solver (DGETRF and DGETRS) from LAPACK simply for comparison purposes. Note that sparse linear solvers should be used in practice for best efficiency. In all cases, the full Jacobian is supplied.

We vary both absolute and relative error tolerances from 10^{-2} to 10^{-7} to obtain solutions of different levels of accuracy (the absolute and relative error tolerances are equal to each other). The simulation time interval is $[t_0, t_F] = [0, 0.02]$ time units. A reference solution is obtained by using LSODE [71], a well known but relatively slow ODE solver, with a very tight relative and absolute tolerances of 10^{-14}. The relative error is defined by (2.62).

The work-precision diagrams for the stiff solvers are shown in Figure 2.3. The results indicate that singly diagonally implicit and fully implicit Runge-Kutta methods implemented in FATODE outperform the BDF method implemented in CVODE, requiring fewer time steps and considerably smaller CPU times to reach a desired accuracy. The Rosenbrock method is also more efficient than CVODE for accuracy levels below 10^{-6}.



(a) Relative error vs number of steps

(b) Relative error vs CPU time

Figure 2.3: Forward integration of the shallow water equations (2.62) using stiff integrators. Comparison is made between FIRK, SDIRK, and Rosenbrock methods in FATODE with the BDF method in CVODE. Different points on the plots correspond to different absolute and relative tolerances levels in the range $10^{-2}, \ldots, 10^{-7}$.

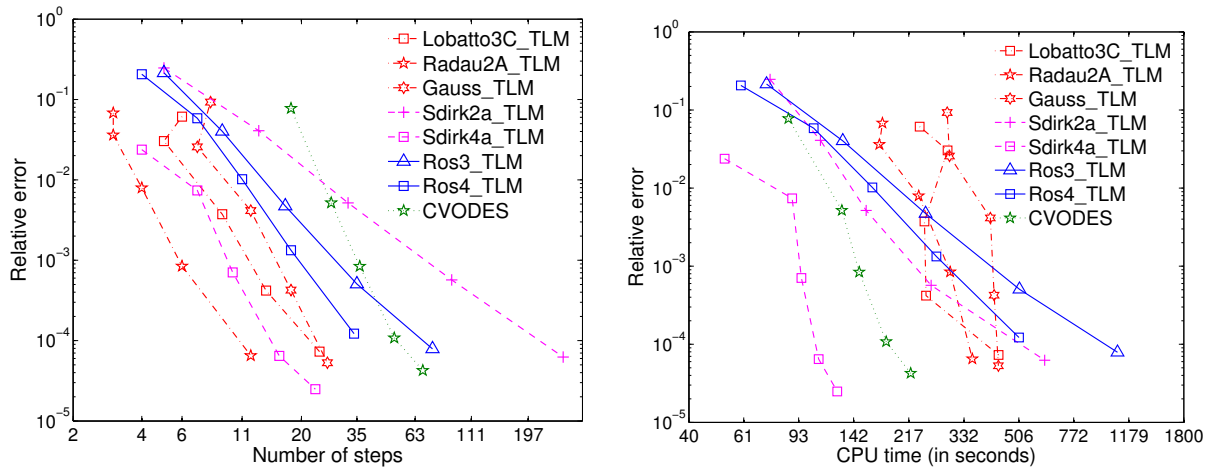
2.7.2 Direct sensitivity analysis

We now calculate the sensitivities of all solution components at the final time with respect to the initial value of the first solution component $\partial y_i(t_F)/\partial y_1(t_0)$, $i = 1 \ldots n$ using tangent linear model integration. The FATODE results are compared against those obtained with CVODES [18], an extension of CVODE capable to perform sensitivity analysis. The LAPACK linear solvers are used in both CVODES and FATODE.

Tangent linear model results with the stiff integrators are shown in Figure 2.4. The three implicit methods in FATODE requires considerably fewer steps than CVODES. The performance of the SDIRK and Rosenbrock methods is comparable to that of the BDF method in CVODES in terms of accuracy versus CPU time. The fully implicit Runge-Kutta method is nearly three times more expensive since it solves either a large real-valued system or a complex-valued system at each step.

2.7.3 Adjoint sensitivity analysis

We calculate the sensitivities of the first solution component at the final time with respect to all initial values $\partial y_1(t_F)/\partial y_i(t_0), \quad i = 1 \ldots d$ using adjoint model integration. Work-precision diagrams for the implicit solvers are shown in Figure 2.5. All implicit methods in FATODE outperform CVODES in terms of both number of steps and CPU time required for a given solution accuracy. The highest efficiency is achieved by the Rosenbrock method, despite the



(a) Relative error vs number of steps

(b) Relative error vs CPU time

Figure 2.4: Tangent linear model integration of the shallow water equations (2.62) using stiff solvers. Comparison is made between implicit methods in FATODE with the BDF method in CVODES. Different points on the plots correspond to different absolute and relative tolerances levels in the range $10^{-2}, \ldots, 10^{-6}$. The tangent linear model computes the sensitivity $\partial y_i(t_F)/\partial y_1(t_0),\ i = 1 \ldots d$.

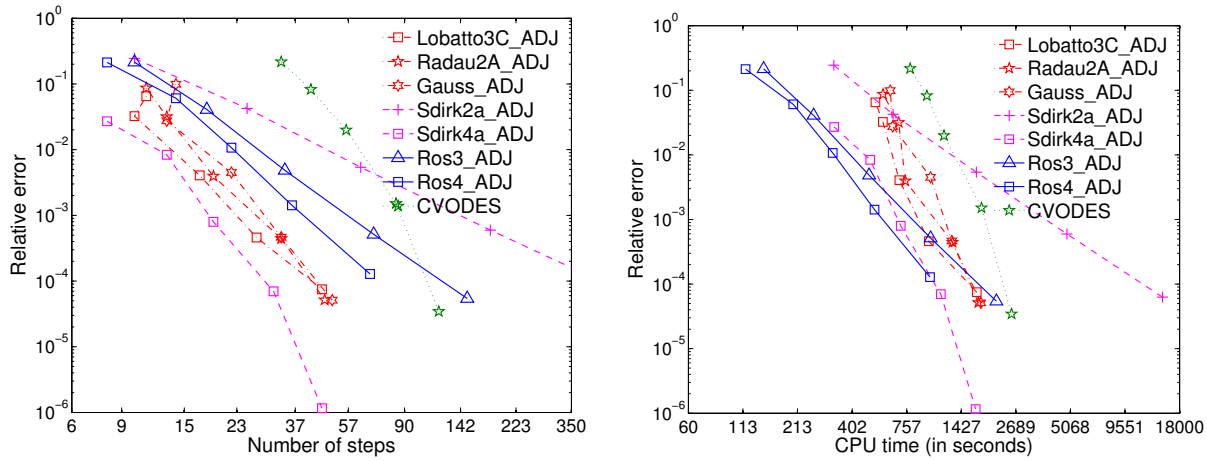
computation of Hessian-vector products.

2.8 Numerical experiments with a very stiff problem: the Carbon Bond-IV chemical system

We now apply FATODE to the integration and parameter sensitivity analysis for a very stiff system. The experiments are carried out on the same platform and with the same Fortran compiler and optimization options as for the shallow water test. The relative errors are evaluated using formulas (2.62) and (2.63).

The Carbon Bond-IV Mechanism (CBM-IV), which consists of 32 species and 81 reactions, was developed for simulating urban smog and modeling regional atmospheric pollution [53]. Complete descriptions of the CBM-IV chemical mechanism, and of the setting of the numerical experiments, are given in the Table 2.8, 2.8, and 2.4. All numerical experiments integrate the CBM-IV system over a 72 hours interval (three diurnal cycles).

It is shown in Table 2.5 that the first six dominant eigenvalues of the Jacobian matrix vary in magnitude from -1.4×10^9 to -2.4×10^{-1}, which indicates that the system is very stiff and requires implicit methods with good stability properties. We select two L-stable and stiffly-accurate methods (a high order one and a low order one) from each of the three categories



(a) Relative error vs number of steps

(b) Relative error vs CPU time

Figure 2.5: Adjoint integration of the shallow water equations (2.62) using stiff solvers. Comparison is made between implicit methods in FATODE with the BDF method in CVODES. Different points on the plots correspond to different absolute and relative tolerances levels in the range $10^{-2}, \ldots, 10^{-6}$. The adjoint model computes the sensitivity $\partial y_1(t_F)/\partial y_i(t_0),\ i = 1 \ldots d$.

of implicit methods in FATODE. Their performance is compared against the BDF method implemented in CVODES.

The sensitivities of individual species concentrations at the final time with respect to a variety of reaction rate coefficients are calculated using adjoint integration. These sensitivities quantify the change in concentrations due to small perturbations in the reaction rate coefficients, and help identify the most important reactions. Of particular interest is the response of the gas-phase species O_3, NO_2, $HONO$, N_2O_5, and HNO_3 to perturbations of reaction rates. The sensitivities of these species with respect to 24 constant reaction rates are shown in Table 2.6 (the table displays the full reactions whose rates are included in the calculation).

To obtain a reference solution both adjoint CVODES and the adjoint RADAU5 integrator in FATODE were run with very tight tolerance settings. For the ODE solution we use $rtol = 10^{-14}$ and $atol = 10^{-8} \times atol_0$ for both codes. The of entries of the vector of absolute tolerances $atol_0$ reflect the magnitude of typical concentrations of individual chemical species. CVODES requires additional tolerances for sensitivity analysis, which are set to $rtol\_sen = 10^{-14}$, $atol\_sen = 10^{-17}$. The two codes yield sensitivity results that are very close to each other. To provide the most favorable comparison setting for CVODES we use its results as the reference solutions in (2.62) and (2.63).

For the numerical tests each solver is run with a series of progressively tighter tolerances $\{atol, rtol\} = \{atol_0/(5^k), 10^{-3}/(5^k)\}$ for $k = 0, \ldots, 7$ (for CVODES we use $k = 0, \ldots, 9$ to better capture its behavior at high accuracy levels). The forward ODE integration results

Table 2.2: List of species in CBM-IV

No.	Species	Initial concentrations (in molecules/cm^3)	Absolute tolerances ($ATOL_0$)
1	O1D	3.65E−2	1E−9
2	H2O2	3.47E11	1E4
3	PAN	2.56E3	1E−4
4	CRO	3.35E−23	1E−30
5	TOL	5.29E−20	1E−27
6	N2O5	1.31E7	1
7	XYL	0	1E−14
8	XO2N	1.99E1	1E−6
9	HONO	3.84E8	1E1
10	PNA	2.68E8	1E1
11	TO2	1.59E−24	1E−31
12	HNO3	1.19E12	1E5
13	ROR	4.89E−5	1E−12
14	CRES	5.46E−21	1E−28
15	MGLY	1.94E−23	1E−30
16	CO	2.33E12	1E5
17	ETH	2.11E−30	1E−37
18	XO2	5.51E8	1E1
19	OPEN	2.63E−21	1E−28
20	PAR	8.12E3	1E−4
21	HCHO	3.00E10	1E3
22	ISOP	0	1E−14
23	OLE	0	1E−14
24	ALD2	2.69E2	1E−5
25	O3	2.06E12	1E5
26	NO2	1.56E10	1E3
27	OH	3.67E7	1
28	HO2	9.89E8	1E1
29	O	1.25E4	1E−3
30	NO3	3.36E7	1
31	NO	2.73E9	1E2
32	C2O3	6.53	1E−7

shown in Figures 2.7(a), 2.7(b) reveal that the most efficient solvers are FATODE's Rodas4 (for lower accuracies) and Radau2A (for higher accuracies), followed closely by CVODES.

The work-precision diagrams for the adjoint sensitivity results are shown in Figure 2.7(c) (with respect to the number of backward steps) and in Figure 2.7(d) (with respect to CPU time). For FATODE the number of backward steps is the same as the number of accepted steps during forward integration, while for CVODES the number of backward steps varies for each different adjoint problem. Figure 2.7(c) reveals that the number of backward steps in FATODE is in general smaller that in CVODES. The results in Figure 2.7(d) show that for moderate accuracies (relative error above 10^{-5}) the SDIRK and the Rosenbrock adjoint methods in FATODE are more effective than CVODES. For high accuracies the most efficient is FATODE's FIRK adjoint method Radau2A. Note that CVODES implements a variable-order BDF method with maximum order five. The most efficient method Radau2A has order five as well, while the other schemes in FATODE have lower orders. So in this sense the

Table 2.3: List of reactions in CBM-IV

No.	Reactions	No.	Reactions
1	NO2+hv=NO+O	42	ALD2+O=C2O3+OH
2	O+O2+M=O3	43	ALD2+OH=C2O3
3	O3+NO=NO2	44	ALD2+NO3=C2O3+HNO3
4	O+NO2=NO	45	ALD2+hv+2O2=HCHO+XO2+CO+2HO2
5	O+NO2=NO3	46	C2O3+NO=HCHO+XO2+HO2+NO2
6	O+NO=NO2	47	C2O3+NO2=PAN
7	O3+NO2=NO3	48	PAN=C2O3+NO2
8	O3+hv=O	49	2C2O3=2HCHO+2XO2+2HO2
9	O3+hv=O1D	50	C2O3+HO2=0.79HCHO+0.79XO2+0.79HO2+0.79OH
10	O1D=O	51	OH=HCHO+XO2+HO2
11	O1D+H2O=2OH	52	PAR+OH=0.87XO2+0.13XO2N+0.11HO2+0.11ALD2 +0.76ROR-0.11PAR
12	O3+OH=HO2	53	ROR=1.1ALD2+0.96XO2+0.94HO2+0.04XO2N +0.02ROR-2.10PAR
13	O3+HO2=OH	54	ROR=HO2
14	NO3+hv=0.89NO2+0.89O+0.11NO	55	ROR+NO2=PROD
15	NO3+NO=2NO2	56	O+OLE=0.63ALD2+0.38HO2+0.28XO2+0.3CO +0.2 HCHO+0.02XO2N+0.22PAR+0.2OH
16	NO3+NO2=NO+NO2	57	OH+OLE=HCHO+ALD2+XO2+HO2-PAR
17	NO3+NO2=N2O5	58	O3+OLE=0.5ALD2+0.74HCHO+0.33CO+0.44HO2 + 0.22XO2+0.1OH-PAR
18	N2O5+H2O=2HNO3	59	NO3+OLE=0.91XO2+HCHO+ALD2+0.09XO2N +NO2-PAR
19	N2O5=NO3+NO2	60	O+ETH=HCHO+0.7XO2+CO+1.7HO2+0.3OH
20	2NO=2NO2	61	OH+ETH=XO2+1.56HCHO+HO2+0.22ALD2
21	NO+NO2+H2O=2HONO	62	O3+ETH=HCHO+0.42CO+0.12HO2
22	OH+NO=HONO	63	OH+TOL=0.08XO2+0.36CRES+0.44HO2+0.56TO2
23	HONO+hv=OH+NO	64	TO2+NO=0.9NO2+0.9OPEN+0.9HO2
24	OH+HONO=NO2	65	TO2=HO2+CRES
25	2HONO=NO+NO2	66	OH+CRES=0.4CRO+0.6XO2+0.6HO2+0.3OPEN
26	OH+NO2=HNO3	67	NO3+CRES=CRO+HNO3
27	OH+HNO3=NO3	68	CRO+NO2=PROD
28	HO2+NO=OH+NO2	69	OH+XYL=0.7HO2+0.5XO2+0.2CRES+0.8MGLY +1.10 PAR+0.3TO2
29	HO2+NO2=PNA	70	OH+OPEN=XO2+C2O3+2HO2+2CO+HCHO
30	PNA=HO2+NO2	71	OPEN+hv=C2O3+CO+HO2
31	OH+PNA=NO2	72	O3+OPEN=0.03ALD2+0.62C2O3+0.7HCHO+0.03XO2 +0.69CO+0.08OH+0.76HO2+0.2MGLY
32	2HO2=H2O2	73	OH+MGLY=XO2+C2O3
33	2HO2+H2O=H2O2	74	MGLY+hv=C2O3+CO+HO2
34	H2O2+hv=2OH	75	O+ISOP=0.6HO2+0.8ALD2+0.55OLE+0.5XO2 +0.5CO+0.45ETH+0.9PAR
35	OH+H2O2=HO2	76	OH+ISOP=HCHO+XO2+0.67HO2+0.4MGLY +0.2C2O3+ETH+0.2ALD2+0.13XO2N
36	OH+CO=HO2	77	O3+ISOP=HCHO+0.4ALD2+0.55ETH+0.2MGLY +0.06CO+ 0.1PAR+0.44HO2+0.1OH
37	HCHO+OH=HO2+CO	78	NO3+ISOP=XO2N
38	HCHO+hv+2O2=2HO2+CO	79	XO2+NO=NO2
39	HCHO+hv=CO	80	2XO2=PROD
40	HCHO+O=OH+HO2+CO	81	XO2N+NO=PROD
41	HCHO+NO3=HNO3+HO2+CO		

Table 2.4: List of reaction rates in CBM-IV

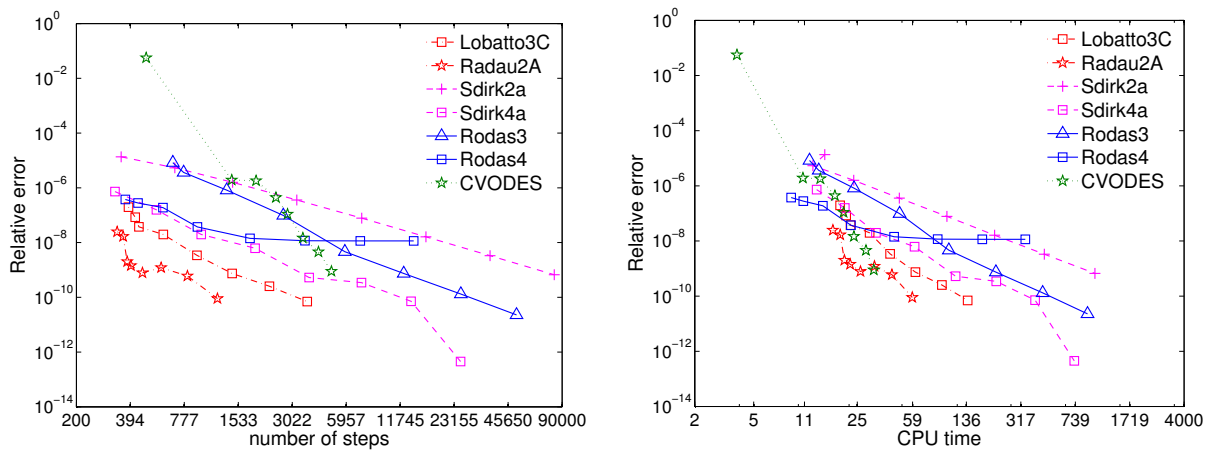
No.	Reaction rate	No.	Reaction rate
1	(8.89E−3)∗SUN	42	ARR2(1.2E−11, −9.86E2)
2	ARR2(1.4E3, 1.175E3)	43	ARR2(7.0E−12, 2.5E2)
3	ARR2(1.8E−12, −1.37E3)	44	2.5E−15
4	9.3E−12	45	(4.0E−6)∗SUN
5	ARR2(1.6E−13, 6.87E2)	46	ARR2(5.4E−12, 2.5E2)
6	ARR2(2.2E−13, 6.02E2)	47	ARR2(8.0E−20, 5.5E3)
7	ARR2(1.2E−13, −2.45E3)	48	ARR2(9.4E16, −1.4E4)
8	(3.556E−4)∗SUN	49	2.0E−12
9	(2.489E−5)∗SUN	50	6.5E−12
10	ARR2(1.9E8, 3.9E2)	51	ARR2(1.1E2, −1.71E3)
11	2.2E−10	52	8.1E−13
12	ARR2(1.6E−12, −9.4E2)	53	ARR2(1.0E15, −8.0E3)
13	ARR2(1.4E−14, −5.8E2)	54	1.6E3
14	(1.378E−1)∗SUN	55	1.5E−11
15	ARR2(1.3E−11, 2.5E2)	56	ARR2(1.2E−11, −3.24E2)
16	ARR2(2.5E−14, −1.23E3)	57	ARR2(5.2E−12, 5.04E2)
17	ARR2(5.3E−13, 2.56E2)	58	ARR2(1.4E−14, −2.105E3)
18	1.3E−21	59	7.7E−15
19	ARR2(3.5E14, −1.0897E4)	60	ARR2(1.0E−11, −7.92E2)
20	ARR2(1.8E−20, 5.3E2)	61	ARR2(2.0E−12, 4.11E2)
21	4.399 99E−40	62	ARR2(1.3E−14, −2.633E3)
22	ARR2(4.5E−13, 8.06E2)	63	ARR2(2.1E−12, 3.22E2)
23	(1.511E−3)∗SUN	64	8.1E−12
24	6.6E−12	65	4.2
25	1.0E−20	66	4.1E−11
26	ARR2(1.0E−12, 7.13E2)	67	2.2E−11
27	ARR2(5.1E−15, 1.0E3)	68	1.4E−11
28	ARR2(3.7E−12, 2.40E2)	69	ARR2(1.7E−11, 1.16E2)
29	ARR2(1.2E−13, 7.49E2)	70	3.0E−11
30	ARR2(4.8E13, −1.0121E4)	71	(5.334E−5)∗SUN
31	ARR2(1.3E−12, 3.8E2)	72	ARR2(5.4E−17, −5.0E2)
32	ARR2(5.9E−14, 1.15E3)	73	1.7E−11
33	ARR2(2.2E−38, 5.8E3)	74	(1.654E−4)∗SUN
34	(6.312E−6)∗SUN	75	1.8E−11
35	ARR2(3.1E−12, −1.87E2)	76	9.6E−11
36	2.2E−13	77	1.2E−17
37	1.0E−11	78	3.2E−13
38	(2.845E−5)∗SUN	79	8.1E−12
39	(3.734E−5)∗SUN	80	ARR2(1.7E−14, 1.3E3)
40	ARR2(3.0E−11, −1.55E3)	81	6.8E−13
41	6.3E−16		

1. The coefficient SUN is updated with time. It is 0 during the night. During daytime, it is computed as $(1 + \cos(\pi * ((2 * tl - tsr - tss)/(tss - tsr))^2))/2$ where tl is the local time, tsr and tss correspond to the sunrise time and sunset time respectively. In our experiments, we use $tsr = 4.5$ (4:30am) and $tss = 19.5$ (7:30pm).

2. Function ARR2(a, b) is defined as $a * \exp(b/temperature)$.

Table 2.5: The first 6 dominant eigenvalues of Jacobian matrix in CBM-IV

Rank	Eigenvalues
1	−1.404 53E9
2	−7.219 91E4
3	−3.749 62E3
4	−4.222 09
5	−2.272 99
6	−2.442 18E−1



(a) Relative error vs number of steps

(b) Relative error vs CPU time

Figure 2.6: Forward integration of CBM-IV using stiff integrators. Comparison is made between FIRK, SDIRK, and Rosenbrock methods in FATODE with the BDF method in CVODES. Different points on each curve correspond to a series of absolute and relative tolerances decreasing by a factor of 5.

comparison between Radau2A and CVODES seems to be more fair. Although CVODES has low cost per step, advantage in stability property saves Radau2A a significant number of integration steps resulting in less CPU time. Comparing the two plots we can see that the shapes of the curves almost stay the same for all the solvers. This is due to the fact that the time cost of backward integration dominates the total time cost for this problem.

The species concentrations in CBM-IV can take widely different values (e.g., $H_2O_2 \sim 10^{11}$ while $ROR \sim 10^{-8}$ molecules/cm^3). We have observed that calculating sensitivities of species with small concentrations is challenging for both codes. To validate the results we have approximated these sensitivities using complex finite differences [72]; a comparison revealed that FATODE provides results that are closer to the complex-step approximations than CVODES.

When calculating the sensitivities of other species to the reaction rates, we have observed that

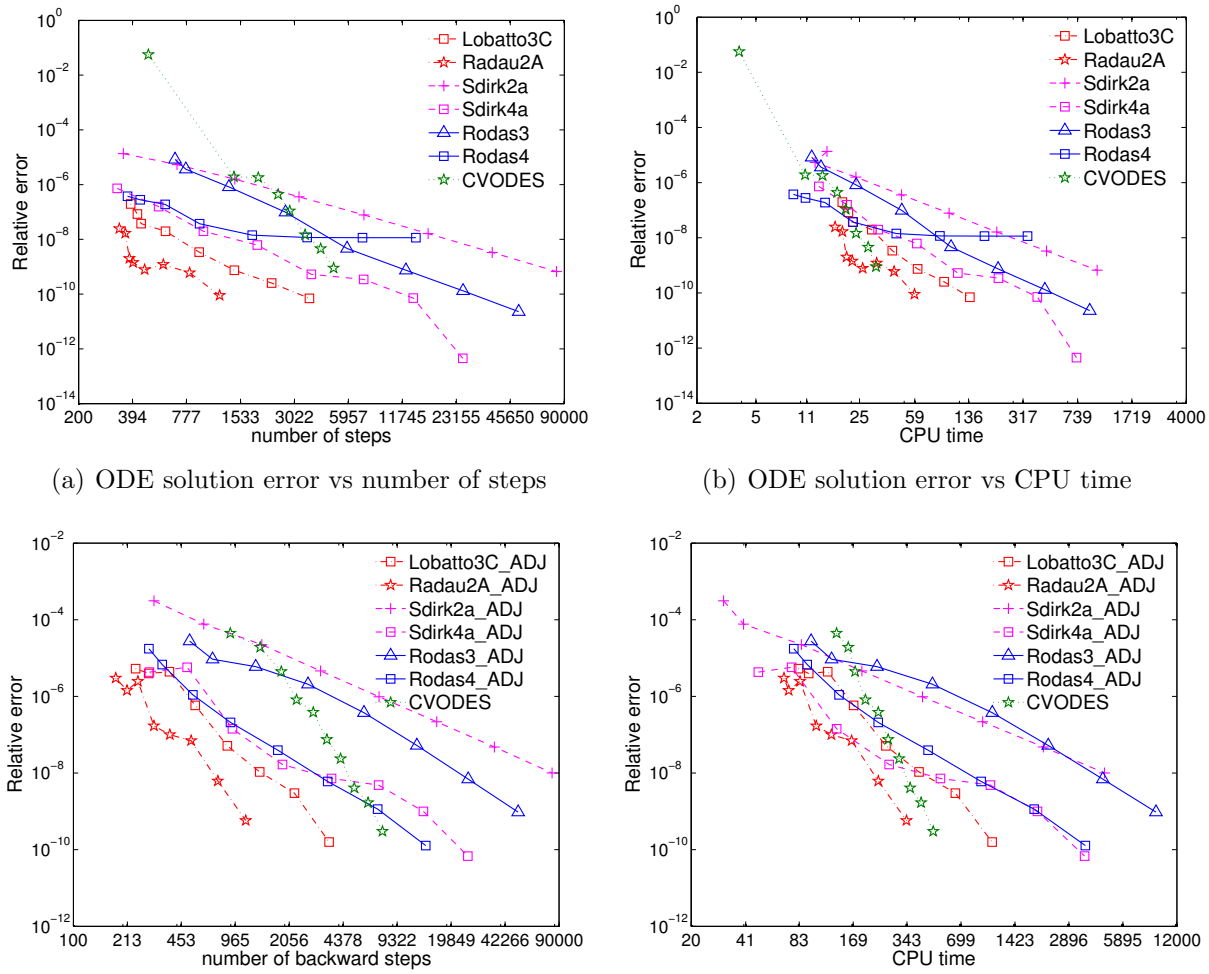
Table 2.6: The sensitivities of five species to reaction rates at a temperature of 298K

Reaction No.	N_2O_5	$HONO$	HNO_3	O_3	NO_2
4	−8.28E13	2.36E15	−2.21E16	−1.44E19	−2.10E14
11	1.39E16	5.36E17	−1.72E18	−1.27E21	3.03E17
18	−1.21E27	−4.39E27	2.10E29	−1.61E31	−1.85E29
21	2.14E37	1.52E41	−1.50E41	−1.15E42	−3.57E39
24	−3.28E15	−8.91E18	8.89E18	1.64E20	2.78E17
25	−4.94E16	−1.79E20	1.70E20	−2.62E21	6.06E18
36	−6.93E18	−1.11E20	−1.37E20	−2.57E23	−1.03E20
37	−4.24E16	−4.69E17	−1.79E18	−1.73E21	−4.20E17
41	−1.88E19	2.88E20	−5.84E20	−2.93E24	−2.43E21
44	3.67E9	−6.93E10	2.35E12	6.27E14	−2.94E11
49	−8.63E2	2.03E4	−1.85E5	−1.65E8	2.35E3
50	−2.18E8	5.25E9	2.64E11	−4.40E13	−4.22E10
52	−6.94E8	9.39E10	1.21E13	−3.66E14	−3.30E11
54	−1.43E−6	−1.45E−5	−5.27E−2	−7.29E−2	−6.78E−4
55	−3.65E5	−3.03E6	−1.26E10	−1.95E10	−1.66E8
64	2.53E−16	4.80E−16	9.13E−12	1.95E−11	1.17E−13
65	−4.88E−28	−9.26E−28	−1.76E−23	−3.76E−23	−2.26E−25
66	5.67E−15	1.39E−14	1.74E−10	4.38E−10	2.24E−12
67	−1.05E−14	−2.54E−14	−3.22E−10	−8.15E−10	−4.15E−12
68	−5.72E−20	−1.94E−19	−5.36E−18	−4.71E−15	−1.93E−19
70	−3.25E−17	−9.29E−16	2.15E−15	3.86E−13	−8.02E−16
74	6.77E−19	9.25E−18	3.11E−17	3.12E−14	7.26E−18
79	4.82E16	−1.73E18	1.53E19	8.36E21	−1.37E17
81	3.86E6	2.99E8	−4.22E10	−3.63E11	6.04E9

for species with extremely small concentrations, e.g. much smaller than machine accuracy, the CVODES result does not agree with the FATODE result, while for species with relatively large concentrations, they are close to each other. The difficulty of calculating sensitivities accurately for species with extremely small concentrations may due to computational errors such as roundoff error and truncation error. To further verify the solution, we approximate the sensitivities using complex-step methods [72] which are usually not very accurate but can serve as an alternative way to estimate the sensitivities. The comparison between these three different methods shows that the FATODE result appears to be closer to the complex-step approximation in extreme cases and our results for species with relatively large concentrations are reliable. A part of the numerical results is illustrated in Table 2.7, where the concentration of H2O2 is around 3.23E11 and that of ROR is around 5.42E−8 at the end of integration time.

2.9 Comparison of different option choices in FATODE

FATODE offers a multitude of methods, and a wide range of tunable parameters for each method. This allows the user to configure the solvers such as to best meet the needs of the application at hand. This section discusses several important choices of methods and parameters in FATODE, and illustrates their impact on the solver performance with the help



(a) ODE solution error vs number of steps

(b) ODE solution error vs CPU time

(c) Adjoint sensitivity error error vs number of backward steps

(d) Adjoint sensitivity error vs CPU time

Figure 2.7: Work-precision diagrams for the ODE solution and for adjoint sensitivities for the stiff CBM-IV test problem. Adjoint sensitivities of five chosen species (O_3, NO_2, $HONO$, N_2O_5, and HNO_3) with respect to 24 constant reaction rate coefficients are computed.

Table 2.7: Comparison between complex-step approximation, FATODE and CVODE

Reaction	H2O2			ROR		
	CVODES	FATODE	CVM	CVODES	FATODE	CVM
NO2+O=NO	−1.58E18	−1.58E18	−1.58E18	4.44E4	5.58E−1	7.75E−1
O1D+H2O=2OH	1.17E20	1.17E20	1.17E20	−2.13E7	−4.52E2	−4.52E2
N2O5+H2O=2HNO3	7.47E29	7.47E29	7.40E29	6.04E16	1.83E12	1.82E12
NO2+NO+H2O=2HONO	1.86E41	1.86E41	−3.01E44	−2.66E28	−7.05E23	4.62E27
HONO+OH=NO2	−2.29E19	−2.29E19	−2.27E19	1.27E6	8.15E1	8.11E1
2HONO=NO2+NO	−1.56E20	−1.56E20	−1.60E25	1.80E7	1.18E3	2.04E8
CO+OH=HO2	2.04E23	2.04E23	2.03E23	2.96E9	2.75E5	2.72E5
HCHO+OH=CO+HO2	−2.10E20	−2.10E20	−2.09E20	3.70E7	1.02E3	1.01E3
HCHO+NO3=HNO3+CO+HO2	−4.24E23	−4.24E23	−4.21E23	1.16E9	−1.17E5	−1.12E5
ALD2+NO3=HNO3+C2O3	5.51E13	5.51E13	−5.83E19	5.13	−3.44E−4	8.26E2
2C2O3=2XO2+2HCHO+2HO2	−2.60E7	−2.60E7	−5.77E16	−1.03E−8	1.13E−11	1.07
HO2+C2O3=0.79XO2+0.79HCHO+0.79OH	−8.41E12	−8.41E12	−2.17E16	−6.46E−1	−7.04E−6	3.14E−1
PAR+OH=0.13XO2N+0.76ROR+0.87XO2	−1.51E14	−1.51E14	−1.97E17	−4.69E5	−3.93E5	−3.80E5
ROR=HO2	3.31E−3	3.31E−3	−7.97E1	3.85E−11	3.09E−11	3.06E−11
ROR+NO2=	−2.69E8	−2.69E8	−8.48E15	5.56E−1	4.34E−1	5.71E−1
TO2+NO=0.9OPEN+0.9NO2+0.9HO2	2.86E−12	2.86E−12	−1.52E16	−8.92E−24	−2.45E−30	2.59E−1
TO2=CRES+HO2	−5.51E−24	−5.51E−24	−2.93E4	2.30E−35	4.74E−42	4.99E−13
CRES+OH=0.4CRO+0.6XO2+0.3OPEN+0.6HO2	4.22E−11	4.22E−11	−3.00E15	−3.38E−22	−1.02E−28	5.11E−2
CRES+NO3=CRO+HNO3	−7.90E−11	−7.90E−11	−5.60E15	9.90E−24	1.96E−28	9.52E−2
CRO+NO2=	−5.01E−13	−2.18E−15	−8.80E15	6.76E−26	5.01E−32	1.50E−1
OPEN+OH=2CO+XO2+HCHO+2HO2+C2O3	1.53E−12	1.53E−12	−4.11E15	1.32E−24	3.69E−30	6.98E−2
MGLY+OH=XO2+C2O3	−1.81E−14	−1.81E−14	−7.25E15	1.44E−26	7.19E−33	1.23E−1
XO2+NO=NO2	1.06E21	1.06E21	1.05E21	−3.47E8	−2.15E2	−2.12E2
XO2N+NO=	−5.46E11	−5.46E11	−1.72E17	−3.50	2.19E−6	2.98

of the shallow water and carbon bond example problems. For a detailed discussion of all available choices the reader should consult FATODE User's guide [49].

2.9.1 Choice of method

Generally speaking, higher order methods are more efficient than lower order methods when medium to high accuracies are sought. Figure 2.2 illustrates this in the context of the shallow water problem: the eighth-order DOPRI853 is the most efficient explicit method for both the ODE solution and the sensitivity analysis.

In case of implicit schemes the solution of nonlinear systems dominates the total cost when large ODE systems are solved. The test results on the shallow water equations (reported in the supplementary material)reveal that Sdirk4a is the most efficient implicit solver for both forward and tangent linear integration. For smaller ODE systems like CBM-IV the highest efficiency can be obtained by FIRK methods, e.g., Radau2A in Figure 2.7. The sixth-order method Gauss takes too many steps to be competitive on CBM-IV. This is due to its weak A-stability. For larger systems the stage coupling inherent with FIRK methods impacts the cost of the linear solvers, which explains why SDIRK and Rosenbrock methods perform best on the shallow water test.

Results with both examples confirm the fact that Rosenbrock methods are particularly favorable for low accuracy requirements. Ros4 and Ros3 have the same orders with Rodas4 and Rodas3, but take fewer stages, thus are expected to behave similarly on problems that are mildly stiff. Additional tests with CBM-IV confirm that on very stiff problems the stiffly accurate schemes Rodas4 and Rodas3 perform considerably better than Ros4 and Ros3,

respectively.

2.9.2 Choice of linear solver

We have assessed the efficiency of sparse linear solvers incorporated in FATODE by comparing their performance with LAPACK full linear solvers in the shallow water test. The Jacobian matrix is of dimension 4800, but only 100,800 elements (0.4375%) are non-zero. The incorporation of sparse linear solvers leads to significant computational savings and allows for very efficient forward, tangent linear, and adjoint model integration. The compute times for various implicit integrators in FATODE using different linear solvers are shown in Table 2.8. As expected, FATODE benefits significantly from the use of sparse linear solvers. UMFPACK is slightly faster than SuperLU for this test, and both sparse solvers give essentially the same results as the full algebra linear solver. Users can link their own linear algebra routines, e.g., preconditioned Krylov based iterative methods.

Table 2.8: Overall compute times (in seconds) using different linear solvers in FATODE for the shallow water test problem. The tolerances are $atol = rtol = 10^{-6}$ and the time interval is $t_0 = 0$ to $t_F = 0.02$ (time units).

Solver	Full algebra	Sparse algebra	
	LAPACK	UMFPACK	SuperLU
Ros4	474.7	21.6	31.7
Ros4_TLM	506.8	58.3	71.4
Ros4_ADJ	991.7	96.8	120.2
Lobatto3C	342.0	8.8	14.8
Lobatto3C_TLM	431.3	32.9	40.0
Lobatto3C_ADJ	1722.4	69.0	102.2
Sdirk4a	74.5	3.8	5.7
Sdirk4a_TLM	125.2	46.0	49.0
Sdirk4a_ADJ	1685.9	69.0	107.4

2.9.3 Choice of the direct versus adjoint approach

In the shallow water test, we used tangent linear model to calculate $\partial y_i(t_F)/\partial y_1(t_0)$, $i = 1 \ldots d$ (with one tangent linear variable) and adjoint model to calculate $\partial y_i(t_F)/\partial y_1(t_0)$, $i = 1 \ldots d$ (with one adjoint variable). These two vectors correspond to the first column and the first row of the full sensitivity matrix $(\partial y_i(t_F)/\partial y_j(t_0)) \in \mathbb{R}^{d \times d}$, respectively. The two approach compute the same sensitivities: the tangent linear result and the adjoint result converge to the same value for $\partial y_1(t_F)/\partial y_1(t_0)$. The computational costs are, however, different. The adjoint approach is considerably more efficient when only a small number

of rows in the sensitivity matrix are needed, and the direct approach is considerably more efficient when only a few columns are computed. A comparison of Figure 2.4b and Figure 2.5b reveals that the tangent linear approach (to compute a column) is at least twice as fast as the adjoint approach (to compute a row of the sensitivity matrix) for the same accuracy level and same integration method. This can also be expected when calculating the full sensitivity matrix where the number of parameters is equal to the dimension of the system.

2.9.4 Solution of the sensitivity system

Both the forward sensitivity system (2.13) and the adjoint sensitivity system (2.17) are linear with respect to the corresponding sensitivity variables. Consequently, at each step of an implicit solver, the stage variables are the solutions of a linear system , e.g., Eqn. (2.30) for tangent linear Runge-Kutta and Eqn.(2.55) for discrete adjoint Runge-Kutta methods. FATODE offers two choices to solve these linear systems: directly, at the expense of additional LU decompositions each step, and iteratively, via simplified Newton iterations that reuse the LU decomposition for each stage.

We investigate how the solution approach affects the accuracy and cost of sensitivities computed with FIRK and SDIRK methods for the shallow water and CBM-IV problems. The results for Sdirk4a and Radau2A methods are shown in Table 2.9. For the large shallow water problem the direct solve is significantly more expensive than the iterative approach, while providing similar accuracy. In contrast, the direct solve is both more efficient and slightly more accurate for the CBM-IV test.

Table 2.9: Timings and accuracy of the CBM-IV test and the shallow water test solving the linear adjoint system directly (or via simplified Newton iterations)

Method	Tolerances		Relative error	CPU time
	Rtol	Atol		
			CBM-IV	
Sdirk4a	1E-3	$atol_0$	4.307E-6 (1.933E-5)	52.25ms (93.43ms)
	2E-4	$0.2 \times atol_0$	5.757E-6 (1.231E-5)	77.41ms (14.73ms)
Radau2A	1E-3	$atol_0$	3.047E-6 (2.997E-6)	55.95ms (69.99ms)
	2E-4	$0.2 \times atol_0$	1.454E-6 (1.436E-6)	61.04ms (73.25ms)
			Shallow water	
Sdirk4a	1E-2	1E-2	2.697E-2 (2.697E-2)	14.41s (11.68s)
	1E-3	1E-3	8.349E-2 (8.349E-2)	20.22s (17.37s)
Radau2A	1E-2	1E-2	8.669E-2 (8.669E-2)	160.06s (32.85s)
	1E-3	1E-3	3.170E-2 (3.170E-2)	203.75s (39.82s)

The relative efficiency of the two choices is both problem dependent and method dependent.

The iterative approach is preferable for large problems, while the direct approach works best with smaller systems. For the s-stage FIRK methods the direct solution involves one coupled system of dimension $sd \times sd$, while for SDIRK methods there are s systems of dimension $d \times d$. Considering the cost scaling, the direct approach is the default option for SDIRK methods and the iterative approach is the default for FIRK methods. This can be changed by users to best match the problem at hand.

2.10 Conclusions

This chapter presents the FATODE library for the integration of stiff ODE systems, and for performing direct and discrete adjoint sensitivity analyses. The software is based on our KPP numerical library; while KPP solves only chemical kinetic systems, FATODE offers a general purpose implementation that makes it potentially useful for a wide range of applications. Important areas that can benefit from using FATODE include uncertainty quantification, inverse problems such as parameter estimation and data assimilation, and optimization of systems governed by ODEs.

FATODE implements explicit Runge-Kutta methods for non-stiff problems, and fully implicit Runge-Kutta, singly diagonally implicit Runge-Kutta, and Rosenbrock methods for stiff problems. This distinguishes the software from CVODES, which is based on linear multistep methods.

FATODE employs a *discrete adjoint sensitivity analysis* approach, i.e., it computes the derivatives of the numerical solution. This approach sets it apart from both DENSERKS and CVODES, which implement a continuous adjoint approach. The discrete adjoint approach gives gradients of the numerical cost function that are exact, to roundoff error. Such gradients are highly suitable for numerical optimization problems, as well as for aposteriori error estimation in complex systems. It has been shown in [58] that the discrete Runge-Kutta adjoints are dual consistent. Consequently, the discrete adjoints computed by FATODE represent solutions of the continuous adjoint ODEs, that are as accurate as the solutions obtained by the continuous adjoint approach. We note that controlling the sensitivity error may require several runs. Since sensitivity coefficients can vary over many orders of magnitude, a good setting of the absolute tolerances may require several trial and error iterations. In the adjoint approach, the sensitivity error depends on both the forward and the adjoint integration errors - this issue is common to both continuous and discrete approaches. While the independent adjoint step size control in the continuous approach is apparently an advantage, current implementations do not account for the interpolation errors. Moreover, in practice, the interplay between forward, interpolation, and adjoint discretization errors is very difficult to control directly. A good guideline is to use tighter tolerances than required by the application during the sensitivity calculations.

All methods in FATODE use local truncation error estimation and step size control for effi-

ciency. The step size is also adjusted to ensure rapid convergence of the linear and nonlinear system solvers at each step. Checkpointing, needed for adjoint runs, is performed in a manner that is completely transparent to the user. The discrete implicit sensitivity equations require the solution of different linear systems for each stage, or of linear systems that couple all stages of a method. Two options are offered in FATODE: the first one constructs each linear system and solves it individually, the second one performs simplified Newton iterations that allow to reuse one LU factorization across all stages. FATODE controls the iteration number, and possibly the step size, such that the iteration error in the latter approach is smaller than the local truncation error at the current step.

FATODE contains stand-alone linear system solvers for different types of systems. The package includes direct solvers for both dense (LAPACK) and sparse (SUPERLU, UMFPACK) systems. The decoupling between the time integration routines and the linear solvers allows users to easily add their own methods, e.g., preconditioned iterative solvers tuned for the application at hand. Details can be found in the User's guide [49].

The implicit formulation of the stiff solvers, as well as the formulation of the tangent linear and adjoint methods, require the user to provide various derivatives such as Jacobian-vector, transposed Jacobian-vector, and Hessian-vector products. These derivatives can be obtained analytically, by finite differences, or by automatic differentiation. Since the accuracy of the derivative approximations directly impacts the overall accuracy of the sensitivity results, care must be exercised when finite differences are employed.

Numerical experiments presented here reveal that FATODE compares favorably with the current state of the art package CVODES. For the (larger) shallow water problem in Section 2.6 the stiff FATODE solvers are considerably more efficient, in terms of work-precision, than the backward differentiation formulas implemented in CVODES. For non-stiff solvers the linear multistep methods provide better performance. FATODE sensitivity solutions are considerably more accurate than CVODES' for the stiff chemical system discussed in Section 2.8.

Code availability

The source code and the User's manual are available online from the FATODE web site [50]. The code has been tested under the following compilers: Portland group's pgf90, Lahey's lf95, Sun's sunf90, gfortran, g95, and Absoft.

Chapter 3

IMEX general linear methods

3.1 Introduction

Many science and engineering problems require numerical simulations of multiphysics and mulstiscale systems, i.e., systems driven by multiple simultaneous physical processes evolving at different time scales.

Such problems can be expressed concisely as the system of ordinary differential equations (ODEs)

$$y' = f(t, y) + g(t, y) \quad t_0 \leq t \leq t_F \,, \quad y(t_0) = y_0 \,, \tag{3.1}$$

where f corresponds to the nonstiff term, and g corresponds to the stiff term. In case of systems of partial differential equations (PDEs) the system (3.1) appears after semi-discretization in space. As an example consider advection-diffusion-reaction systems where the advection is slow while the diffusion and reaction are typically fast [3, 4, 5]. Another example is the semi-implicit time integration of PDEs, where g represents the linear part of the discretized spatial operator, and f is its nonlinear part [73, 74, 75].

The best numerical solution strategy for a particular problem depends on its dynamics. For non-stiff processes explicit time discretizations are the most efficient, due to their low cost per step. For stiff processes explicit methods require prohibitively small time steps (limited by the fastest time scale in the system). In this case implicit methods, designed such that their step sizes are not limited by stability considerations, are more efficient [51, 47]. The numerical solution of multiphysics systems (3.1) is challenging as neither purely explicit nor purely implicit methods are completely satisfactory. Explicit methods have restricted time steps (due to g), while implicit methods require the solution of (non)linear systems of equations that involve *all* the processes in the model.

The implicit-explicit (IMEX) approach alleviates these difficulties by combining an implicit scheme for the stiff component with an explicit scheme for the non-stiff component. The pair

is chosen such that the overall discretization of (5.1) has the desired stability and accuracy properties. IMEX linear multistep (LM) methods have been proposed in [29, 30, 76] and IMEX Runge-Kutta (RK) schemes in [77, 33, 34, 78]. The orders of consistency of these methods is typically lower than five. High order IMEX RK methods are difficult to construct due to the large number of order conditions. IMEX LM methods have decreasing stability properties with increasing order.

The general linear method (GLM) family proposed by J.C Butcher [22, 79] generalizes both Runge-Kutta and linear multistep methods. The added complexity improves the flexibility to develop methods with better stability and accuracy properties. While Runge-Kutta and linear multistep methods are special cases of GLMs, the framework allows for the construction of many other methods as well. Here we focus on the diagonally implicit multistage integration methods (DIMSIM) [23, 80], which are both efficient and accurate, and great potentials for practical use. GLM can overcome the limitations of both linear multistep methods (lack of A-stability at high orders) and of Runge-Kutta methods (low stage order leading to order reduction). A complete treatment of GLMs can be found in the book of Jackiewicz [24].

In this study we develop the concept of partitioned DIMSIM methods, and develop an order conditions theory for a family of such methods. This shows that partitioned GLM is a great framework for developing multi-methods (composite methods). Next, we propose a new family of implicit-explicit methods based on pairs of DIMSIMs, and develop second and third order methods on this class.

In our earlier work [43, 81] we have developed second order IMEX-GLM schemes. While this research was under study, we became aware of an effort to construct IMEX-GLM schemes for Hamiltonian systems [82]. The theoretical results presented in this chapter have also been used in the construction of extrapolation-based IMEX-GLM schemes [83].

This chapter is organized as follows. Section 3.2 reviews the class of general linear methods. The new concept of partitioned DIMSIM schemes is proposed in Section 3.3, and the order conditions theory is developed. IMEX-DIMSIM schemes are constructed in Section 3.4. Linear stability is analyzed in section 3.4.5, and Prothero-Robinson convergence in section 3.4.5. IMEX methods of second and third order are built in Sections 3.5.1 and 3.5.2, respectively. Numerical results for van der Pol system and for the two dimensional gravity waves equations are presented in Section 3.6. Section 3.7 draws conclusions and points to future work.

3.2 General linear methods

3.2.1 Representation of general linear methods

Consider the initial value problem for an autonomous system of differential equations in the form

$$y'(t) = f(y)\,, \quad t_0 \le t \le t_F\,, \quad y(t_0) = y_0\,, \tag{3.2}$$

with $f : \mathbb{R}^d \to \mathbb{R}^d$ and $y(t) \in \mathbb{R}^d$. GLMs [24] for (3.2) can be represented by the abscissa vector $\mathbf{c} \in \mathbb{R}^s$, and four coefficient matrices $\mathbf{A} \in \mathbb{R}^{s\times s}$, $\mathbf{U} \in \mathbb{R}^{s\times r}$, $\mathbf{B} \in \mathbb{R}^{r\times s}$ and $\mathbf{V} \in \mathbb{R}^{r\times r}$ which can be represented compactly in the following tableau

$$\begin{array}{c|c} \mathbf{A} & \mathbf{U} \\ \hline \mathbf{B} & \mathbf{V} \end{array}\,.$$

On the uniform grid $t_n = t_0 + nh$, $n = 0, 1, \ldots, N$, $Nh = t_F - t_0$, one step of the GLM reads

$$Y_i = h\sum_{j=1}^{s} a_{i,j}\, f(Y_j) + \sum_{j=1}^{r} u_{i,j}\, y_j^{[n-1]},\ i = 1, \ldots, s, \tag{3.3a}$$

$$y_i^{[n]} = h\sum_{j=1}^{s} b_{i,j}\, f(Y_j) + \sum_{j=1}^{r} v_{i,j}\, y_j^{[n-1]},\ i = 1, \ldots, r, \tag{3.3b}$$

where s is the number of internal stages and r is the number of external stages. Here, h is the step size, Y_i is an approximation to $y(t_{n-1} + c_i h)$ and $y_i^{[n]}$ is an approximation to the linear combination of the derivatives of y at the point t_n. The method (3.3) can be represented in vector form

$$Y = h\ (\mathbf{A} \otimes \mathbf{I}_{d\times d})\ F(Y) + (\mathbf{U} \otimes \mathbf{I}_{d\times d})\ y^{[n-1]}, \tag{3.4a}$$

$$y^{[n]} = h\ (\mathbf{B} \otimes \mathbf{I}_{d\times d})\ F(Y) + (\mathbf{V} \otimes \mathbf{I}_{d\times d})\ y^{[n-1]}, \tag{3.4b}$$

where $\mathbf{I}_{d\times d}$ is an identity matrix of the dimension of the ODE system.

3.2.2 Stability considerations

The linear stability of method (3.3) is analyzed in terms of its stability matrix

$$\mathbf{M}(z) = \mathbf{V} + z\,\mathbf{B}\ (\mathbf{I}_{s\times s} - z\mathbf{A})^{-1}\ \mathbf{U}\,, \tag{3.5}$$

and the corresponding stability function

$$p(w, z) = \det(w\mathbf{I}_{r\times r} - \mathbf{M}(z)), \tag{3.6}$$

where $w, z \in \mathbb{C}$. A desirable property is the inherent Runge-Kutta stability [84, 85]. This means that the stability function (3.6) has the form

$$p(w,z) = w^{s-1}\left(w - R(z)\right), \tag{3.7}$$

where $R(z)$ is the stability function of Runge Kutta method of order $p = s$.

3.2.3 Accuracy considerations

We assume that the components of the input vector $y_i^{[n-1]}$ for the next step in (3.3) satisfy

$$y_i^{[n-1]} = \sum_{k=0}^{p} q_{i,k} h^k \, y^{(k)}(t_{n-1}) + \mathcal{O}(h^{p+1}), \quad i = 1, \ldots, r, \tag{3.8}$$

for some real parameters $q_{i,k}$, $i = 1, \ldots, r$, $k = 0, 1, \ldots, p$.

The method (3.3) has *order* p if the output vector $y_i^{[n]}$ satisfies

$$y_i^{[n]} = \sum_{k=0}^{p} q_{i,k} h^k y^{(k)}(t_n) + \mathcal{O}(h^{p+1}), \quad i = 1, \ldots, r, \tag{3.9}$$

for the same parameters $q_{i,k}$ of (3.8).

The method (3.3) has *stage order* q if the internal stage vectors $Y_i^{[n]}$ are approximations of order q to the solution at the time points $t_{n-1} + c_i\, h$

$$Y_i^{[n]} = y(t_{n-1} + c_i\, h) + \mathcal{O}(h^{q+1}), \quad i = 1, \ldots, s\,. \tag{3.10}$$

We collect the parameters $q_{i,k}$ in the matrix $\mathbf{W}$ for convenience

$$\mathbf{W} = [\mathbf{q}_0 \quad \mathbf{q}_1 \quad \cdots \quad \mathbf{q}_p] = \begin{bmatrix} q_{1,0} & q_{1,1} & \cdots & q_{1,p} \\ q_{2,0} & q_{2,1} & \cdots & q_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ q_{r,0} & q_{r,1} & \cdots & q_{r,p} \end{bmatrix}. \tag{3.11}$$

Theorem 3.2.1 (GLM order conditions [24])**.** *Assume that $y^{[n-1]}$ satisfies (3.8). Then the GLM (3.3) has order p (3.9) and stage order $q = p$ (3.10) if and only if*

$$e^{cz} = z\mathbf{A}e^{cz} + \mathbf{U}w(z) + \mathcal{O}(z^{p+1}), \tag{3.12a}$$

$$e^{z}w(z) = z\mathbf{B}e^{cz} + \mathbf{V}w(z) + \mathcal{O}(z^{p+1})\,, \tag{3.12b}$$

where

$$e^{cz} = [e^{c_1 z}, \ldots, e^{c_s z}]^T\,, \quad w(z) = \sum_{j=0}^{p} \mathbf{q}_j\, z^j\,.$$

For stage order $q = p - 1$ condition (3.12a) *is replaced by*

$$e^{cz} \;=\; z\mathbf{A}e^{cz} + \mathbf{U}w(z) + \left(\frac{\mathbf{c}^p}{p!} - \frac{\mathbf{A}\,\mathbf{c}}{(p-1)!} - \mathbf{U}\mathbf{q}_p\right) z^p + \mathcal{O}(z^{p+1})\,. \tag{3.12c}$$

Proof. See Butcher and Jackiewicz [22] [24, Section 2.4]. □

It is shown in [22, 24] that a GLM (3.3) has order p and stage order q with $q = p = r = s$ if and only if

$$\mathbf{B} = \mathbf{B}_0 - \mathbf{A}\mathbf{B}_1 - \mathbf{V}\mathbf{B}_2 + \mathbf{V}\mathbf{A}, \tag{3.13}$$

where the matrices $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2 \in \mathbb{R}^{s\times s}$ are defined by

$$(\mathbf{B}_0)_{i,j} = \frac{\int_0^{1+c_i} \phi_j(x)dx}{\phi_j(c_j)}\,, \quad (\mathbf{B}_1)_{i,j} = \frac{\phi_j(1+c_i)}{\phi_j(c_j)}\,, \quad (\mathbf{B}_2)_{i,j} = \frac{\int_0^{c_i} \phi_j(x)dx}{\phi_j(c_j)}\,, \tag{3.14}$$

with

$$\phi_i(x) = \prod_{j=1, j\neq i}^{s} (x - c_j), \quad i = 1, \ldots, s$$

(cf. [22, Thm. 5.1],[24, Thm. 3.2.1]).

3.2.4 Starting and ending procedures

Assumption (3.8) requires to compute the initial vector $y^{[0]}$ by a starting procedure satisfying

$$y_i^{[0]} = \sum_{k=0}^{p} q_{i,k} h^k y^{(k)}(t_0) + \mathcal{O}(h^{p+1}), \quad i = 1, \ldots, r\,. \tag{3.15}$$

Dense output is based on derivative approximations of the form

$$h^k y^{(k)}(t_n) \approx \sum_{i=0}^{s} h\beta_{k,i} f(Y_i) + \sum_{j=0}^{r} \gamma_{k,j} y_j^{[n-1]}, \quad k = 0, 1, \ldots, r\,. \tag{3.16}$$

It is shown in [22, 86] that (3.16) is accurate within $\mathcal{O}(h^{p+1})$ if and only if

$$[1, z, \ldots, z^p]^T e^z = z\widetilde{\mathbf{B}}e^{cz} + \widetilde{\mathbf{V}}w(z) + \mathcal{O}(z^{p+1}) \tag{3.17}$$

where $\widetilde{\mathbf{B}} = [\beta_{k,i}]$ and $\widetilde{\mathbf{V}} = [\gamma_{k,i}]$. The finishing procedure uses (3.16) with $k = 0$ to generate the solution at the last step

$$y(t_n) \approx \sum_{i=0}^{s} h\beta_{0,i} f(Y_i) + \sum_{j=0}^{r} \gamma_{0,j} y_j^{[n-1]}. \tag{3.18}$$

3.2.5 Diagonally implicit multistage integration methods

Diagonally implicit multistage integration methods (DIMSIMs) are a subclass of GLMs characterized by the following properties [22]:

1. $\mathbf{A}$ is lower triangular with the same element $a_{i,i} = \lambda$ on the diagonal;
2. $\mathbf{V}$ is a rank-1 matrix with the nonzero eigenvalue equal to one to guarantee preconsistency;
3. The order p, stage order q, number of external stages r, and number of internal stages s are related by $q \in \{p-1, p\}$ and $r \in \{s, s+1\}$.

In this work we focus on DIMSIMs with $p = q = r = s$, $\mathbf{U} = \mathbf{I}_{s\times s}$, and $\mathbf{V} = \mathbf{1}_s\, v^T$, where $v^T\, \mathbf{1}_s = 1$ [24]. DIMSIMs can be categorized into four types according to [22]. Type 1 or type 2 methods have $a_{i,j} = 0$ for $j \geq i$ and are suitable for a sequential computing environment, while type 3 and type 4 methods have $a_{i,j} = 0$ for $j \neq i$ and are suitable for parallel computation. Methods of type 1 and 3 are explicit ($a_{i,i} = 0$), while methods of type 2 and 4 are implicit ($a_{i,i} = \lambda \neq 0$) and potentially useful for stiff systems.

Next we briefly review the construction of type 1 DIMSIMs with $p = q = r = s$, $\mathbf{U} = \mathbf{I}$, and $\mathbf{V} = ev^T$, where $v^T e = 1$ since we will need to follow some procedures here when constructing IMEX-DIMSIM schemes. For details regarding construction of Type 2 methods, we refer readers to Jackiewicz's book[24].

Imposing the appropriate stage order and order conditions and additional stability requirement results in large systems of polynomial equations for the remaining unknown parameters of the methods. If the order of the methods is less than 5, these systems can be generated and solved symbolic manipulation packages such as `MATHEMATICA` or `MAPLE`.

It can be demonstrated that the stability function of DIMSIM of type 1 is a polynomial of the form

$$p(w, z) = w^s - p_1(z)w^{s-1} + \cdots + (-1)^{s-1}p_{s-1}(z)w + (-1)^s p_s(z), \tag{3.19}$$

where

$$\begin{aligned}
p_1(z) &= 1 + p_{1,1}z + p_{1,2}z^2 + \cdots + p_{1s}z^s,\\
p_2(z) &= p_{21}z + p_{22}z^2 + \cdots + p_{2s}z^s,\\
&\vdots\\
p_{s-1}(z) &= p_{s-1,s-2}z^{s-2} + p_{s-1,s-1}z^{s-1} + p_{s-1,s}z^s,\\
p_s(z) &= p_{s,s-1}z^{s-1} + p_{ss}z^s.
\end{aligned}$$

Note that the coefficients $p_{i,j}$ of the polynomials $p_i(z)$ depend on $a_{i,j}$, v_i. This leads to the system of $(s-1)(s+2)/2$ nonlinear equations

$$p_{kl} = 0,\ \ k = 2, 3, \ldots, s,\ \ l = k-1, k, \ldots, s, \tag{3.20}$$

with respect to $(s-1)(s+2)/2$ unknowns $a_{i,j}$ and v_i.

3.3 Partitioned general linear methods

Consider the partitioned system of ODEs

$$y' = \begin{bmatrix} y_{\{1\}} \\ \vdots \\ y_{\{N\}} \end{bmatrix}' = \begin{bmatrix} f_{\{1\}}(y_{\{1\}}, \ldots, y_{\{N\}}) \\ \vdots \\ f_{\{N\}}(y_{\{1\}}, \ldots, y_{\{N\}}) \end{bmatrix} = f(y)\,, \tag{3.21}$$

where the solution vector is separated into components $y_{\{m\}}$, $m = 1, \ldots, N$, each of which may be itself a vector.

A *partitioned general linear method* solves (3.21) by applying a different GLM to each component. If not explicitly stated otherwise , we use the subscript $\{m\}$ to denote the coefficients specific to the m-th component of the partitioned system. We have the following

Definition 1 (Partitioned GLM)**.** *One step of a partitioned GLM has the form*

$$\begin{aligned} Y_{\{m\}i} &= h \sum_{j=1}^{s} a_{\{m\}i,j}\, f_{\{m\}}(Y_{\{1\}j}, Y_{\{2\}j}, \ldots, Y_{\{N\}j}) && (3.22a) \\ &\quad + \sum_{j=1}^{r} u_{\{m\}i,j}\, y_{\{m\}j}^{[n-1]}, \quad i = 1, \ldots, s, \\ y_{\{m\}i}^{[n]} &= h \sum_{j=1}^{s} b_{\{m\}i,j}\, f_{\{m\}}(Y_{\{1\}j}, Y_{\{2\}j}, \ldots, Y_{\{N\}j}) && (3.22b) \\ &\quad + \sum_{j=1}^{r} v_{\{m\}i,j}\, y_{\{m\}j}^{[n-1]}, \quad i = 1, \ldots, r, \end{aligned}$$

where $a_{\{m\}i,j}, u_{\{m\}i,j}, b_{\{m\}i,j}$, and $c_{\{m\}i}$ for $m = 1, \ldots, N$ represent the coefficients of N different GLMs.

Definition 2 (Internal consistency)**.** *A partitioned GLM* (3.22) *is internally consistent if all component methods share the same abscissae, $\mathbf{c}_{\{m\}i} = \mathbf{c}_i$ for $m = 1, \ldots, N$.*

Internal consistency means that all stage components approximate the solution components at the same time point, i.e., $[Y_{\{1\}j}, \ldots, Y_{\{N\}j}]^T \approx y(t_n + c_j h)$, for all $j = 1, \ldots, s$. An internally consistent partitioned GLM method (3.22) can be represented compactly as

$$\mathbf{c}_{\{m\}} = \mathbf{c}\,, \qquad \begin{array}{c|c} \mathbf{A}_{\{m\}} & \mathbf{U}_{\{m\}} \\ \hline \mathbf{B}_{\{m\}} & \mathbf{V}_{\{m\}} \end{array}\,. \tag{3.23}$$

Definition 3 (Order of partitioned GLM)**.** *Assume that each component of the input vector satisfies* (3.8)

$$y_{\{m\}\,i}^{[n-1]} = \sum_{k=0}^{p} q_{\{m\}i,k} h^k \, y_{\{m\}}^{(k)}(t_{n-1}) + \mathcal{O}(h^{p+1}), \quad i = 1, \ldots, r\,. \tag{3.24}$$

The partitioned GLM (3.22) *has order* p *if each component of the output vector satisfies*

$$y_{\{m\}\,i}^{[n]} = \sum_{k=0}^{p} q_{\{m\}\,i,k} h^k y_{\{m\}}^{(k)}(t_n) + \mathcal{O}(h^{p+1}), \quad i = 1, \ldots, r\,, \quad m = 1, \ldots, N\,, \tag{3.25}$$

for the same parameters $q_{\{m\}i,k}$ *as in* (3.24)*. The partitioned GLM* (3.22) *has stage order* q *if each component of the internal stages* $Y_i^{[n]}$ *satisfies*

$$Y_{\{m\}\,i}^{[n]} = y_{\{m\}}(t_{n-1} + c_{\{m\}i}\, h) + \mathcal{O}(h^{q+1}), \quad i = 1, \ldots, s\,, \quad m = 1, \ldots, N\,. \tag{3.26}$$

Theorem 3.3.1 (Order conditions for partitioned GLMs)**.** *Assume that each component* $y_{\{m\}j}^{[n-1]}$ *satisfies (3.8). Then the internally consistent partitioned GLM* (3.23) *has order* p (3.25) *and stage order* $q \in \{p-1, p\}$ (3.26) *if and only if each component method* $(\mathbf{A}_{\{m\}}, \mathbf{B}_{\{m\}}, \mathbf{U}_{\{m\}}, \mathbf{V}_{\{m\}})$ *has order* p (3.9) *and stage order* q (3.10)*.*

Remark 1. *Each component method needs to independently meet its own order conditions* (3.12)*. No additional "coupling" conditions are needed for the partitioned GLM (i.e., no order conditions contain coefficients from multiple component schemes).*

Proof. We first prove the "only if" part: if the partitioned GLM satisfies (3.25)–(3.26) with order p and stage order $q \in \{p-1, p\}$, then each component method satisfies its own order conditions (3.9)–(3.10) with the same p and q. This can be seen immediately by employing the same component method for all partitions, $\big(\mathbf{A}_{\{k\}}, \mathbf{B}_{\{k\}}, \mathbf{U}_{\{k\}}, \mathbf{V}_{\{k\}}\big) \equiv \big(\mathbf{A}_{\{m\}}, \mathbf{B}_{\{m\}}, \mathbf{U}_{\{m\}}, \mathbf{V}_{\{m\}}\big)$ for $k = 1, \ldots, N$. The partitioned method (3.23) is the traditional GLM method $\big(\mathbf{A}_{\{m\}}, \mathbf{B}_{\{m\}}, \mathbf{U}_{\{m\}}, \mathbf{V}_{\{m\}}\big)$ and has to satisfy the traditional order conditions (3.9) and (3.10).

We next prove the "if" part: if each component method satisfies (3.9)–(3.10) with order p and stage order $q \in \{p-1, p\}$, then the partitioned GLM (3.23) has order p and stage order q. Denote

$$Y_j = \begin{bmatrix} Y_{\{1\}j} \\ \vdots \\ Y_{\{N\}j} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_s \end{bmatrix},$$

and

$$y(t_{n-1} + c_j h) = \begin{bmatrix} y_{\{1\}}(t_{n-1} + c_j h) \\ \vdots \\ y_{\{N\}}(t_{n-1} + c_j h) \end{bmatrix}, \quad y(t_{n-1} + \mathbf{c}h) = \begin{bmatrix} y(t_{n-1} + c_1 h) \\ \vdots \\ y(t_{n-1} + c_s h) \end{bmatrix}.$$

Consider the stage equations of the individual method m with exact solution arguments

$$\begin{aligned} y(t_{n-1}+c_i h) &= h\sum_{j=1}^{s} a_{\{m\}i,j}\, f\left(y(t_{n-1}+c_j h)\right) \\ &\quad + \sum_{j=1}^{r} u_{\{m\}i,j}\left(\sum_{k=0}^{p} q_{\{m\}\,i,k} h^k y^{(k)}(t_{n-1})\right) + \mathcal{O}\left(h^{q+1}\right), \quad i=1,\ldots,s\,. \end{aligned} \tag{3.27}$$

The error size is given by the stage order q of each individual method (3.10). Using the assumption (3.24) each component of the sum $\sum_{k=0}^{p} q_{\{m\}\,i,k} h^k y^{(k)}_{\{m\}}(t_{n-1})$ can be replaced by the numerical approximations $y^{[n-1]}_{\{m\}j}$, which differ from their exact counterparts by $\mathcal{O}(h^{p+1})$; therefore their use in (3.27) does not change the asymptotical error size. The m-th component of relation (3.27) then reads

$$\begin{aligned} y_{\{m\}}(t_{n-1}+c_i h) &= h\sum_{j=1}^{s} a_{\{m\}i,j}\, f_{\{m\}}\left(y(t_{n-1}+c_j h)\right) \\ &\quad + \sum_{j=1}^{r} u_{\{m\}i,j}\, y^{[n-1]}_{\{m\}j} + \mathcal{O}\left(h^{q+1}\right), \quad i=1,\ldots,s\,. \end{aligned} \tag{3.28}$$

Subtracting (3.28) from the stage equation (3.22a) gives

$$Y_{\{m\}i} - y_{\{m\}}(t_{n-1}+c_i h) = h\sum_{j=1}^{s} a_{\{m\}i,j}\,\left(f_{\{m\}}(Y_j) - f_{\{m\}}(y(t_{n-1}+c_j h))\right) + \mathcal{O}(h^{q+1})$$

and therefore

$$\left\|Y_{\{m\}i} - y_{\{m\}}(t_{n-1}+c_i h)\right\|_\infty \;\le\; h\left\|\mathbf{A}_{\{m\}}\right\|_\infty L_m\, \left\|Y - y(t_{n-1}+\mathbf{c}h)\right\|_\infty + \mathcal{O}(h^{q+1})$$

where L_m is the Lipschitz constant of $f_{\{m\}}$. It follows that [24]

$$\left\|Y - y(t_{n-1}+\mathbf{c}h)\right\|_\infty = \mathcal{O}(h^{q+1}) \tag{3.29}$$

for all sufficiently small step sizes

$$h < \tau = \left(\max_m \left\|\mathbf{A}_{\{m\}}\right\|_\infty L_m\right)^{-1}.$$

Equation (3.29) proves the stage order of the partitioned GLM method.

Continuing, (3.29) implies that

$$\begin{aligned} h\, f_{\{m\}}\left(Y_i\right) &= h\, f_{\{m\}}\left(y(t_{n-1}+c_i h)\right) + \mathcal{O}(h^{q+2})\,, \\ &= h\, f_{\{m\}}\left(y(t_{n-1}+c_i h)\right) + \mathcal{O}(h^{p+1}) \end{aligned} \tag{3.30}$$

where we have used the fact that $q+2 \geq p+1$. Consider the solution step of the individual method m with exact solution arguments

$$\begin{aligned} \sum_{k=0}^{p} q_{\{m\}\,i,k} h^k y^{(k)}(t_n) &= h \sum_{j=1}^{s} b_{\{m\}i,j}\, f\left(y(t_{n-1}+c_j h) \right) \\ &\quad + \sum_{j=1}^{r} v_{\{m\}i,j} \left(\sum_{k=0}^{p} q_{\{m\}\,i,k} h^k y^{(k)}(t_{n-1}) \right) + \mathcal{O}\left(h^{p+1}\right) \end{aligned} \tag{3.31}$$

for $i = 1, \ldots, r$, where the size of the error term reflects the fact that each individual method has order p.

Use (3.30) and the assumption (3.24) into the m-th component of (3.31) to obtain

$$\begin{aligned} \sum_{k=0}^{p} q_{\{m\}\,i,k} h^k y^{(k)}(t_n) &= h \sum_{j=1}^{s} b_{\{m\}i,j}\, f\left(Y_j \right) + \sum_{j=1}^{r} v_{\{m\}i,j}\, y_{\{m\}\,j}^{[n-1]} + \mathcal{O}\left(h^{p+1}\right) \\ &= y_{\{m\}i}^{[n]} + \mathcal{O}\left(h^{p+1}\right), \quad i = 1, \ldots, r, \end{aligned} \tag{3.32}$$

The last equality follows from the partitioned method solution equation (3.22b). This establishes the order p of the partitioned GLM.

□

3.4 Implicit-explicit general linear methods

3.4.1 Construction procedure

The derivation of IMEX-GLM schemes relies on the partitioned GLM theory developed in Section 3.3. We transform the additively partitioned system (3.1) into a component partitioned system (3.21) via the following transformation [32]

$$\begin{aligned} y &= x + z\,, \\ x' &= \tilde{f}(x,z) = f(x+z)\,, && (3.33a) \\ z' &= \tilde{g}(x,z) = g(x+z)\,. && (3.33b) \end{aligned}$$

Equation (3.33a) is discretized with an explicit (type 1) GLM

$$X_i = h \sum_{j=1}^{i-1} a_{i,j}\, f(X_j + Z_j) + \sum_{j=1}^{r} u_{i,j}\, x_j^{[n-1]}, \quad i = 1, \ldots, s, \tag{3.34a}$$

$$x_i^{[n]} = h \sum_{j=1}^{s} b_{i,j}\, f(X_j + Z_j) + \sum_{j=1}^{r} v_{i,j}\, x_j^{[n-1]}, \quad i = 1, \ldots, r\,. \tag{3.34b}$$

Similarly, equation (3.33b) is discretized with an diagonally implicit (type 2) GLM

$$Z_i = h\sum_{j=1}^{i} \widehat{a}_{i,j}\, g(X_j + Z_j) + \sum_{j=1}^{r} \widehat{u}_{i,j}\, z_j^{[n-1]},\ i = 1,\ldots,s, \tag{3.35a}$$

$$z_i^{[n]} = h\sum_{j=1}^{s} \widehat{b}_{i,j}\, g(X_j + Z_j) + \sum_{j=1}^{r} \widehat{v}_{i,j}\, z_j^{[n-1]},\ i = 1,\ldots,r. \tag{3.35b}$$

Combining (3.34) and (3.35) we obtain

$$\begin{aligned} X_i + Z_i &= h\left(\sum_{j=1}^{i-1} a_{i,j}\, f(X_j + Z_j) + \sum_{j=1}^{i} \widehat{a}_{i,j}\, g(X_j + Z_j)\right) \qquad (3.36a)\\ &\quad + \sum_{j=1}^{r}\left(u_{i,j}\, x_j^{[n-1]} + \widehat{u}_{i,j}\, z_j^{[n-1]}\right),\quad i = 1,\ldots,s,\\ x_i^{[n]} + z_i^{[n]} &= h\left(\sum_{j=1}^{s} b_{i,j}\, f(X_j + Z_j) + \sum_{j=1}^{s} \widehat{b}_{i,j}\, g(X_j + Z_j)\right) \qquad (3.36b)\\ &\quad + \sum_{j=1}^{r}\left(v_{i,j}\, x_j^{[n-1]} + \widehat{v}_{i,j}\, z_j^{[n-1]}\right),\quad i = 1,\ldots,r, \end{aligned}$$

We consider pairs of explicit (3.34) and diagonally implicit (3.35) schemes that

- share the same abscissa vector $\mathbf{c} = \widehat{\mathbf{c}}$ so that the partitioned GLM is internally consistent, and
- share the same coefficient matrices $\mathbf{U} = \widehat{\mathbf{U}}$ and $\mathbf{V} = \widehat{\mathbf{V}}$.

For this class of schemes all internal stage vectors can be combined. Specifically, let $Y_i = X_i + Z_i$ and $y_i = x_i + z_i$. The scheme (3.36) becomes the following method/

Definition 4 (IMEX-GLM methods)**.** *One step of an implicit-explicit general linear method applied to* (3.1) *advances the solution using*

$$Y_i = h\sum_{j=1}^{i-1} a_{i,j}\, f(Y_j) + h\sum_{j=1}^{i} \widehat{a}_{i,j}\, g(Y_j) + \sum_{j=1}^{r} u_{i,j}\, y_j^{[n-1]},\quad i = 1,\ldots,s, \tag{3.37a}$$

$$y_i^{[n]} = h\sum_{j=1}^{s}\left(b_{i,j}\, f(Y_j) + \widehat{b}_{i,j}\, g(Y_j)\right) + \sum_{j=1}^{r} v_{i,j}\, y_j^{[n-1]},\quad i = 1,\ldots,r\,. \tag{3.37b}$$

We note that in (3.37) $x_i^{[n]}$ and $z_i^{[n]}$ need not to be known individually once they are initialized in the first step. The combined solution $y_i^{[n]} = x_i^{[n]} + z_i^{[n]}$ is advanced at each step as regular

GLMs do. The IMEX-GLM (3.37) is represented compactly by the Butcher tableau

$$\begin{array}{c|c|c|c} \mathbf{c} & \mathbf{A} & \widehat{\mathbf{A}} & \mathbf{U} \\ \hline & \mathbf{B} & \widehat{\mathbf{B}} & \mathbf{V} \end{array}. \tag{3.38}$$

3.4.2 Starting procedures

An IMEX GLM (3.37) of order p requires a starting procedure that approximates linear combinations of derivatives as follows

$$x_i^{[0]} = \sum_{k=0}^{r} q_{i,k} h^k x^{(k)}(t_0) + \mathcal{O}(h^p) \quad \text{and} \quad z_i^{[0]} = \sum_{k=0}^{r} \widehat{q}_{i,k} h^k z^{(k)}(t_0) + \mathcal{O}(h^p) \tag{3.39}$$

respectively, where the i-th column of coefficient matrix q and $\widehat{q}$, denoted by q_i and $\widehat{q}_i$ for short, can be computed by

$$q_0 = \mathbf{1}_s, \quad q_i = \frac{\mathbf{c}^i}{i!} - \frac{\mathbf{A}\,\mathbf{c}^{i-1}}{(i-1)!}; \quad \widehat{q}_0 = \mathbf{1}_s, \quad \widehat{q}_i = \frac{\mathbf{c}^i}{i!} - \frac{\widehat{\mathbf{A}}\,\mathbf{c}^{i-1}}{(i-1)!}. \tag{3.40}$$

Thus

$$\begin{aligned} y_i^{[0]} &= x_i^{[0]} + z_i^{[0]} \\ &= x(t_0) + z(t_0) + q_{i,1} h x'(t_0) + \widehat{q}_{i,1} h z'(t_0) + \sum_{k=2}^{r} q_{i,k} h^k x^{(k)}(t_0) + \sum_{k=2}^{r} \widehat{q}_{i,k} h^k z^{(k)}(t_0) \\ &= y_0 + q_{i,1} h f(y_0) + \widehat{q}_{i,1} h g(y_0) + \sum_{k=2}^{r} q_{i,k} h^k x^{(k)}(t_0) + \sum_{k=2}^{r} \widehat{q}_{i,k} h^k z^{(k)}(t_0). \end{aligned}$$

Evaluation of the first three terms is straightforward. But approximations of the other terms containing derivatives $x^{(k)}(t_0)$ and $y^{(k)}(t_0)$ for $k \geq 2$ requires additional work if their analytical expressions are difficult to obtain.

To initialize an IMEX GLM we approximate *independently* the vectors $h^k x^{(k)}(t_0)$, $h^k z^{(k)}(t_0)$, $k = 1, \ldots, r$, using finite differences and the solution information provided by several steps of an IMEX Runge-Kutta method.

For better accuracy, the IMEX RK method uses a small step size $\tau < h$, and produces the numerical solutions $y_i^{\text{start}} \approx y(t_0 + i\tau)$. In the following we show how to compute the terms $\tau^k x^{(k)}(t_0)$; each of these terms is then rescaled by $(h/\tau)^k$ to reflect the integration step h. We have that

$$\begin{bmatrix} \tau x'(t_0) \\ \tau^2 x''(t_0) \\ \vdots \\ \tau^r x^{(r)}(t_0) \end{bmatrix} = \tau \mathbf{D} \begin{bmatrix} x'(t_0) \\ x'(t_1) \\ \vdots \\ x'(t_{r-1}) \end{bmatrix} + \mathcal{O}(\tau^{r+1}) = \tau \mathbf{D} \begin{bmatrix} f(y_0) \\ f\left(y_1^{\text{start}}\right) \\ \vdots \\ f\left(y_{r-1}^{\text{start}}\right) \end{bmatrix} + \mathcal{O}(\tau^{r+1}) \tag{3.41}$$

where the coefficient matrix $D \in \mathbb{R}^{r\times r}$ is derived by expanding the right hand side in Taylor series and comparing the coefficients of each term. For the cases $r = 2$ and $r = 3$ the coefficients are

$$\mathbf{D}|_{r=2} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{D}|_{r=3} = \begin{bmatrix} 1 & 0 & 0 \\ -3/2 & 2 & -1/2 \\ 1 & -2 & 1 \end{bmatrix},$$

respectively. The same procedure is applied to obtain $\tau^k z^{(k)}(t_0)$. We note that the initialization procedure requires the function values $f(y)$ and $g(y)$ evaluated at the starting solution steps y_i^{start}, and that there is no need to compute x_i or z_i separately. A description of the starting procedure in vector form can be found in Section 4.2 of the next chapter.

3.4.3 Finishing procedures

If we choose the last abscissa coordinate c_s to be 1, the approximation to the ODE solution using IMEX GLMs with stage order $q = p$ can be given by the final stage value to the order p. But for IMEX GLMs with stage order $q = p - 1$, a finishing procedure need to be constructed.

To generate the solution at the last time step $y(t_F)$ using (3.18), a general finishing procedure reads

$$y(t_n) \approx \sum_{i=1}^{s} h\beta_{0,i} f(Y_i) + \sum_{j=1}^{r} \gamma_{0,j} x_j^{[n-1]} + \sum_{i=1}^{s} h\widehat{\beta}_{0,i} g(Y_i) + \sum_{j=1}^{r} \widehat{\gamma}_{0,j} z_j^{[n-1]}. \tag{3.42a}$$

In order to avoid the difficulty of evaluating of $x_j^{[n-1]}$ and $z_j^{[n-1]}$ separately we require that $\gamma_{0,j} = \widehat{\gamma}_{0,j}$ for all j. In this case the finishing procedure reads

$$y(t_n) \approx \sum_{i=1}^{s} h\beta_{0,i} f(Y_i) + \sum_{i=1}^{s} h\widehat{\beta}_{0,i} g(Y_i) + \sum_{j=1}^{r} \gamma_{0,j} y_j^{[n-1]}. \tag{3.42b}$$

The construction of the procedure can be simplified by choosing the abscissa vector $[0, c_2, \ldots, c_{s-1}, 1]$. For explicit (type 1) GLMs, $c_1 = 0$ implies that $q_{1,0} = 1$ and $q_{1,j} = 0$ for $j \geq 1$ due to order conditions. According to the formula (3.9), the first element of the output vector is exactly the solution at the current step, $y_1^{[n]} \approx y(t_n)$. In this case, β_0 is equal to the first row of the coefficient matrix $\mathbf{B}$, and γ_0 is the first row of $\mathbf{V}$. Similarly, $c_1 = 0$ results in $\widehat{q}_{1,j} = 0$ for $j \geq 1$ for implicit (type 2) GLMs. According to the order condition (3.12b), we have

$$e^z \sum_{j=0}^{p} \widehat{q}_{1,j}\, z^j = z \sum_{j=1}^{s} \widehat{b}_{1,s} e^{c_j z} + \sum_{i=1}^{s} v_{1,i} \sum_{j=0}^{p} \widehat{q}_{i,j}\, z^j + \mathcal{O}(z^{p+1}),$$

which can be written as

$$e^z = z\sum_{j=1}^{s}\widehat{b}_{1,s}\,e^{c_j z} - z\,\widehat{q}_{1,1}\,e^z + \sum_{j=1}^{s} v_{1,j}\sum_{j=0}^{p}\widehat{q}_{1,j}\,z^j + \mathcal{O}(z^{p+1})\,,$$

Comparing it with (3.17) and assuming $c_s = 1$, we can simply use $\widehat{\beta}_{0,i} = \widehat{b}_{1,i}$ for $i = 0,\ldots,s-1$, $\widehat{\beta}_{0,s} = \widehat{b}_{1,s} - \widehat{q}_{1,1}$ and $\widehat{\gamma_0} = \gamma_0$ to guarantee the finishing procedure gives order p in accuracy for the implicit part. Consequently, we can use the following procedure in practice for the IMEX schemes:

$$\begin{aligned} y(t_n) &\approx \sum_{i=1}^{s} h\,b_{1,i}\,f(Y_i) + \sum_{i=1}^{s} h\,\widehat{b}_{1,i}\,g(Y_i) - \widehat{q}_{1,1}\,g(Y_s) + \sum_{j=1}^{r} v_{1,j}\,y_j^{[n-1]} \\ &= y_1^{[n]} - \widehat{q}_{1,1}\,g(Y_s)\,. \end{aligned} \tag{3.43}$$

Notice that it can also be used on IMEX GLMs with $q = p$ though it is designed for the case $q = p-1$. But our experience shows that there is no obvious advantage doing so compared with using the final stage value which can give very accurate approximations usually.

3.4.4 Linear stability analysis

For convenience, we write the IMEX-GLM (3.37) in the vector form

$$Y = h\mathbf{A}F(Y) + h\widehat{\mathbf{A}}G(Y) + \mathbf{U}\,y^{[n-1]} \tag{3.44a}$$

$$y^{[n]} = h\mathbf{B}F(Y) + h\widehat{\mathbf{B}}G(Y) + \mathbf{V}\,y^{[n-1]}\,. \tag{3.44b}$$

We consider the generalized linear test equation

$$y' = \xi y + \widehat{\xi} y\,, \quad t \geq 0, \tag{3.45}$$

where ξ and $\widehat{\xi}$ are complex numbers. We consider ξy to be the nonstiff term and $\widehat{\xi} y$ the stiff term, and denote $w = h\xi$ and $\widehat{w} = h\widehat{\xi}$.

Applying (3.44) to the test equation (3.45) leads to

$$Y = h\left(\xi\mathbf{A} + \widehat{\xi}\widehat{\mathbf{A}}\right)Y + \mathbf{U}\,y^{[n-1]}, \tag{3.46a}$$

$$y^{[n]} = h\left(\xi\mathbf{B} + \widehat{\xi}\widehat{\mathbf{B}}\right)Y + \mathbf{V}\,y^{[n-1]}\,. \tag{3.46b}$$

Assuming $\mathbf{I}_{s\times s} - w\mathbf{A} - \widehat{w}\widehat{\mathbf{A}}$ is nonsingular we obtain

$$y^{[n]} = \mathbf{M}(w,\widehat{w})\,y^{[n-1]},$$

where the stability matrix is defined by

$$\mathbf{M}(w,\widehat{w}) = \mathbf{V} + \Big(w\,\mathbf{B} + \widehat{w}\,\widehat{\mathbf{B}}\Big)\Big(\mathbf{I}_{s\times s} - w\,\mathbf{A} - \widehat{w}\,\widehat{\mathbf{A}}\Big)^{-1}\,\mathbf{U}\,. \tag{3.47}$$

Let $S \subset \mathbb{C}$ and $\widehat{S} \subset \mathbb{C}$ be the stability regions of the explicit GLM and of the implicit GLM, respectively. The *combined stability region* is defined by

$$\Big\{\, w \in S,\, \widehat{w} \in \widehat{S} \;:\; \rho\big(\mathbf{M}(w,\widehat{w})\big) \le 1 \,\Big\} \subset S \times \widehat{S} \subset \mathbb{C} \times \mathbb{C}\,. \tag{3.48}$$

For a practical analysis of stability we define a *desired stiff stability region*, e.g.,

$$\widehat{\mathcal{S}}_\alpha = \{\widehat{w} \in \widehat{S} \cap \mathbb{C}^- \;:\; |\mathrm{Im}(\widehat{w})| \le \tan(\alpha)\,|\mathrm{Re}(\widehat{w})|\}\,,$$

and compute numerically the corresponding non-stiff stability region:

$$\mathcal{S}_\alpha = \Big\{ w \in S \;:\; \rho\big(\mathbf{M}(w,\widehat{w})\big) \le 1\,,\;\; \forall\, \widehat{w} \in \widehat{\mathcal{S}}_\alpha \Big\}\,. \tag{3.49}$$

The IMEX-GLM method is stable if the constrained non-stiff stability region $\mathcal{S}_\alpha$ is non-trivial (has a non-empty interior) and is sufficiently large for a prescribed (problem-dependent) value of α, e.g., $\alpha = 90°$.

3.4.5 Prothero-Robinson convergence

We now study the possible order reduction for very stiff systems. We consider the Prothero-Robinson (PR) [87] test problem written as a split system (3.1)

$$y' = \underbrace{\mu\,(y - \phi(t))}_{g(y)} + \underbrace{\phi'(t)}_{f(y)}\;, \quad \mu < 0\;, \quad y(0) = \phi(0)\;, \tag{3.50}$$

where the exact solution is $y(t) = \phi(t)$. A numerical method is said to be PR-convergent with order p if its application to (3.50) gives a solution whose the global error decreases as $\mathcal{O}(h^p)$ for $h \to 0$ and $h\mu \to -\infty$.

Theorem 3.4.1 (Prothero-Robinson convergence of IMEX-GLM)**.** *Consider the IMEX GLM method (3.37). Without loss of generality we consider that* $\mathbf{U} = \mathbf{I}$*. The explicit part is of order* p *and stage order* $q \in \{p-1, p\}$*, and the implicit part has order* $\widehat{p} = p$ *and stage order* $\widehat{q} \in \{p-1, p\}$*. Assume that* $h\mu \in \widehat{S}$ *for all* $h > 0$*. Then the IMEX GLM method (3.37) is PR-convergent with order* $\min(p, q)$*.*

Remark 2. *If the explicit stage order is* $q = p$*, then the PR order of convergence is* p*. It is convenient to construct IMEX GLM methods (3.37) with explicit stage order* $q = p$*, even if* $\widehat{q} = p - 1$*, as such methods do not suffer from stiff order reduction on the PR problem.*

Proof. Let

$$\phi^{[n]} = \phi\left(t_{n-1} + \mathbf{c}\, h\right) = \left[\phi(t_{n-1} + c_1\, h), \ldots, \phi(t_{n-1} + c_s\, h)\right]^T \ .$$

and

$$\psi^{[n]} = \left[\phi(t_{n-1}), h\, \phi'(t_{n-1}), \ldots, h^p\, \phi^{(p)}(t_{n-1})\right]^T \ .$$

The method (3.37) applied to (3.50) reads:

$$Y^{[n]} = h\, \mathbf{A}\, \phi'^{[n]} + h\, \mu\, \widehat{\mathbf{A}}\, \left(Y^{[n]} - \phi^{[n]}\right) + \mathbf{U}\, y^{[n-1]}\,, \tag{3.51a}$$

$$y^{[n]} = h\, \mathbf{B}\, \phi'^{[n]} + h\, \mu\, \widehat{\mathbf{B}}\, \left(Y^{[n]} - \phi^{[n]}\right) + \mathbf{V}\, y^{[n-1]}\,. \tag{3.51b}$$

Consider the global stage errors

$$E^{[n]} = Y^{[n]} - \phi^{[n]}\,.$$

To obtain the global error in $y^{[n]}$ we consider separately the global errors in the nonstiff and stiff components:

$$\begin{aligned} e_n^{\text{nonstiff}} &= x^{[n]} - \sum \mathbf{q}_k\, h^k\, x^{(k)}(t_n)\,, \\ e_n^{\text{stiff}} &= z^{[n]} - \sum \widehat{\mathbf{q}}_k\, h^k\, z^{(k)}(t_n)\ , \\ &= \phi^{[n]} - x^{[n]} - \sum \widehat{\mathbf{q}}_k\, h^k\, \left(\phi^{(k)} - x^{(k)}\right)(t_n) \\ &= \phi^{[n]} - x^{[n]} \end{aligned}$$

since the exact solution of the nonstiff system is $x(t) = \phi(t)$. Consequently, the total error is

$$\begin{aligned} e_n &= e_n^{\text{nonstiff}} + e_n^{\text{stiff}} \\ &= \phi^{[n]} - \sum \mathbf{q}_k\, h^k\, \phi^{(k)}(t_n) \\ &= \phi^{[n]} - \mathbf{W}\, \psi^{[n]}\,. \end{aligned}$$

Write the stage equation (3.51a) in terms of the exact solution and global errors

$$E^{[n]} + \phi^{[n]} = h\, \mathbf{A}\, \phi'^{[n]} + h\, \mu\, \widehat{\mathbf{A}}\, E^{[n]} + e_{n-1} + \mathbf{U} \sum_{k=0}^{p} \mathbf{q}_k\, h^k\, \phi^{(k)}(t_{n-1})\,,$$

to obtain

$$\begin{aligned} \left(\mathbf{I}_{s\times s} - h\, \mu\, \widehat{\mathbf{A}}\right)\, E^{[n]} &= e_{n-1} + h\, \mathbf{A}\, \phi'\left(t_{n-1} + \mathbf{c}\, h\right) \\ &\quad + \mathbf{U} \sum_{k=0}^{p} \mathbf{q}_k\, h^k\, \phi^{(k)}(t_{n-1}) - \phi(t_{n-1} + \mathbf{c}h)\,. \end{aligned} \tag{3.52}$$

The exact solution is expanded in Taylor series about t_{n-1}:

$$\phi\left(t_{n-1} + \mathbf{c}\, h\right) - \mathbf{1}_s\, \phi(t_{n-1}) = \sum_{k=1}^{\infty} \frac{h^k \mathbf{c}^k}{k!} \phi^{(k)}(t_{n-1})\,,$$

$$h\,\phi'\left(t_{n-1}+\mathbf{c}\,h\right) \;=\; \sum_{k=1}^{\infty}\frac{kh^k\mathbf{c}^{k-1}}{k!}\phi^{(k)}(t_{n-1})\,.$$

Inserting the above Taylor expansions in (3.52) leads to

$$\begin{aligned}\left(\mathbf{I}_{s\times s}-h\,\mu\,\widehat{\mathbf{A}}\right)E^{[n]} &= e_{n-1}-\mathbf{1}_s\,\phi(t_{n-1})+\mathbf{U}\mathbf{q}_0\,\phi(t_{n-1})\\ &\quad+\sum_{k=1}^{\infty}\left(k\,\mathbf{A}\,\mathbf{c}^{k-1}+k!\,\mathbf{U}\,\mathbf{q}_k-\mathbf{c}^k\right)\frac{h^k}{k!}\phi^{(k)}(t_{n-1})\\ &= e_{n-1}+\mathcal{O}\left(h^{q+1}\right)\end{aligned}$$

where q is the stage order of the explicit method. We have used the facts that $\mathbf{q}_0=\mathbf{1}_s$, $\mathbf{U}\,\mathbf{1}_s=\mathbf{1}_s$, and the order conditions (3.12a) and (3.12c) for the cases where $q=p$ and $q=p-1$, respectively.

Similarly, we write the solution equation (3.51b) in terms of the exact solution and global errors:

$$\begin{aligned}e_n+\sum_{k=0}^{p}\mathbf{q}_k\,h^k\,\phi^{(k)}(t_n) &= h\,\mathbf{B}\,\phi'(t_{n-1}+\mathbf{c}h)+h\,\mu\,\widehat{\mathbf{B}}\,E^{[n]}+\mathbf{V}\,e_{[n-1]}\\ &\quad+\mathbf{V}\sum_{k=0}^{p}\mathbf{q}_k\,h^k\,\phi^{(k)}(t_{n-1})\,.\end{aligned}$$

After rearranging the expression we obtain

$$\begin{aligned}e_n &= \left(h\,\mu\,\widehat{\mathbf{B}}\,\left(\mathbf{I}_{s\times s}-h\,\mu\,\widehat{\mathbf{A}}\right)^{-1}+\mathbf{V}\right)e_{n-1}+h\,\mathbf{B}\,\phi'(t_{n-1}+\mathbf{c}h)+\mathbf{V}\sum_{k=0}^{p}\mathbf{q}_k\,h^k\,\phi^{(k)}(t_{n-1})\\ &\quad-\sum_{k=0}^{p}\mathbf{q}_k\,h^k\,\phi^{(k)}(t_n)+\mathcal{O}\left(h^{q+1}\right)\,.\end{aligned}$$

By Taylor series expansion we have

$$\sum_{k=0}^{p}\mathbf{q}_k\,h^k\,\phi^{(k)}(t_n)=\sum_{k=0}^{p}\left(\sum_{\ell=0}^{k}\frac{\mathbf{q}_{k-\ell}}{\ell!}\right)h^k\,\phi^{(k)}(t_{n-1})$$

and therefore

$$e_n \;=\; \widehat{\mathbf{M}}(h\mu)\,e_{n-1}+\sum_{k=1}^{\infty}\left(k\,\mathbf{B}\,\mathbf{c}^{k-1}+k!\,\mathbf{V}\,\mathbf{q}_k-k!\sum_{\ell=0}^{k}\frac{\widehat{q}_{k-\ell}}{\ell!}\right)\frac{h^k}{k!}\phi^{(k)}(t_{n-1})+\mathcal{O}\left(h^{\widehat{q}+1}\right) \tag{3.53}$$

The order condition (3.12b) of the nonstiff scheme reads

$$e^z w(z)=z\mathbf{B}\,e^{cz}+\mathbf{V}w(z)+\mathcal{O}\left(z^{p+1}\right)$$

$$\sum_{\ell\geq 0}\sum_{k=0}^{p}\frac{\mathbf{q}_k z^{k+\ell}}{\ell!} = \sum_{k=0}^{\infty}\mathbf{B}\frac{\mathbf{c}^k z^{k+1}}{k!} + \sum_{k=0}^{p}\mathbf{V}\mathbf{q}_k z^k + \mathcal{O}\left(z^{p+1}\right) .$$

Identification of powers of z^k leads to

$$\sum_{\ell=0}^{k} p\frac{\widehat{q}_{k-\ell} z^k}{\ell!} = \mathbf{B}\frac{\mathbf{c}^{k-1} z^k}{(k-1)!} + \mathbf{V}\mathbf{q}_k z^k , \quad k = 1, \ldots, p .$$

The error recurrence (3.53) becomes

$$e_n \;=\; \widehat{\mathbf{M}}(h\mu)\, e_{n-1} + \mathcal{O}\left(h^{\min(q+1,p+1)}\right) . \tag{3.54}$$

Assume that the initial error is $e_0 = \mathcal{O}(h^p)$. The error amplification matrix $\widehat{\mathbf{M}}(h\mu)$ is the stability matrix of the implicit method. Therefore its spectral radius is uniformly bounded below one for all argument values $h\mu$ of interest. By standard numerical ODE arguments [51] the equation (3.54) implies convergence of global errors to zero at a rate $\|e_n\| = \mathcal{O}\left(h^{\min(p,q)}\right)$. □

3.5 Construction of implicit-explicit methods of orders two and three

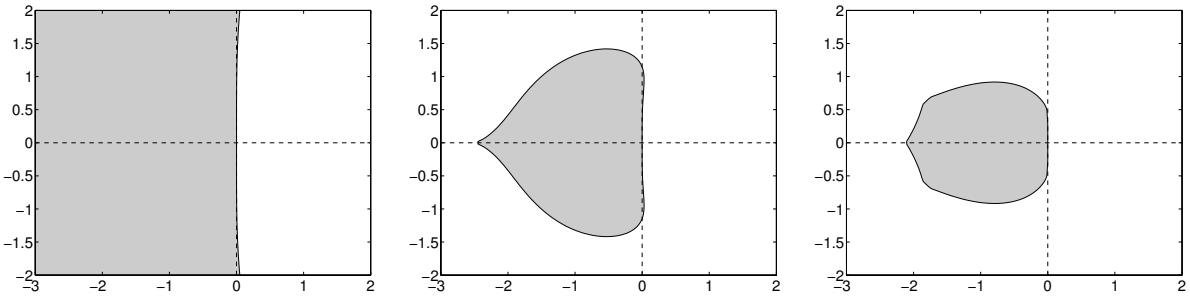
We now construct IMEX-DIMSIM methods as summarized in Section 3.2.5. Specifically, we focus on DIMSIMs with $p = q = r = s$, $\mathbf{U} = \mathbf{I}_{s\times s}$, and $\mathbf{V} = \mathbf{1}_s\, v^T$, where $v^T\, \mathbf{1}_s = 1$ [24].

3.5.1 Two-stage, second-order pairs with $p = q = r = s = 2$

The pair of explicit and implicit schemes developed in [81] is named IMEX-DIMSIM-2A and consists of a type 2 DIMSIM from [22] with the same stability of SDIRK method of order 2, and a type 1 derived DIMSIM. Both of them share the same abscissa vector $\mathbf{c} = [0, 1]^T$ and the same coefficient matrix $\mathbf{V}$. The IMEX-DIMSIM-2A coefficients in the tableau (3.38) representation are

$$\begin{array}{c|cc|cc|cc}
0 & 0 & 0 & \frac{2-\sqrt{2}}{2} & 0 & 1 & 0 \\
1 & 2 & 0 & \frac{2\sqrt{2}+6}{7} & \frac{2-\sqrt{2}}{2} & 0 & 1 \\
\hline
 & \frac{3\sqrt{2}-1}{4} & \frac{3-\sqrt{2}}{4} & \frac{73-34\sqrt{2}}{28} & \frac{4\sqrt{2}-5}{4} & \frac{3\sqrt{2}-3}{4} & \frac{1-\sqrt{2}}{4} \\
 & \frac{3\sqrt{2}-3}{4} & \frac{1-\sqrt{2}}{4} & \frac{87-48\sqrt{2}}{28} & \frac{-45+34\sqrt{2}}{28} & \frac{3-\sqrt{2}}{2} & \frac{\sqrt{2}-1}{2}
\end{array} .$$

The choice of $\lambda = (2-\sqrt{2})/2$ ensures the type implicit part of IMEX-DIMSIM-2A is L-stable. Inherent Runge-Kutta stability is a desirable property, but there are not enough free parameters to enforce this property on both methods of the IMEX pair at the same time.



(a) Stability region $\widehat{S}$ of the implicit method (b) Stability region S of the explicit method (c) Constrained stability region $\mathcal{S}_\alpha$ (3.49) for $\alpha = 90°$

Figure 3.1: Stability regions for the IMEX-DIMSIM-2B pair

For a given implicit scheme we construct the explicit method by maximizing the constrained stability region (3.49). We have observed that simply maximizing the explicit stability region S is insufficient and can lead to a very poor constrained stability region for the IMEX method. The matrix $\mathbf{B}$ can be determined by $\mathbf{A}$, $\mathbf{c}$ and $\mathbf{V}$ according to the order condition (3.13). The only free parameter is $a_{2,1}$ in matrix $\mathbf{A}$, and it is chosen such as to maximize IMEX stability. First, we use a Matlab Differential Evolution package [1] as a heuristic for global optimization to generate a starting point. Then we run the Matlab routine `fminsearch` multiple times until the result converges; each run is initialized with the previous result. The resulting stability regions are reported in Figure 3.1.

This procedure led to another explicit scheme that maximizes the IMEX stability

$$\mathbf{A} = \begin{bmatrix} 0 & 0 \\ 1.5 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{3-\sqrt{2}}{4} \\ \frac{\sqrt{2}-1}{2} & \frac{3-\sqrt{2}}{4} \end{bmatrix};$$

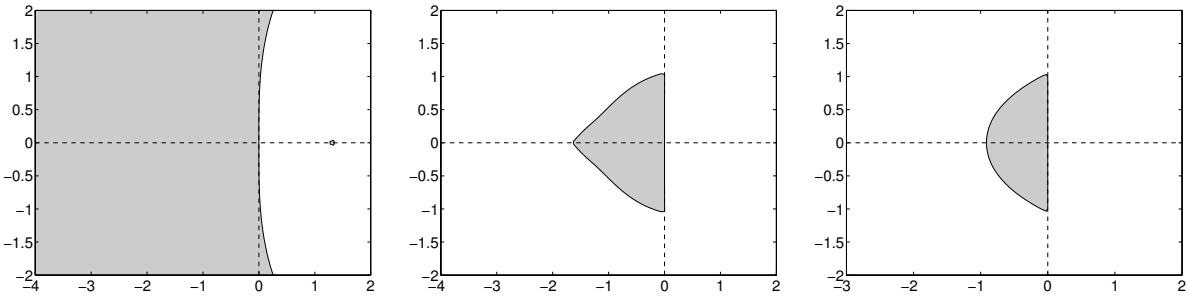
$\mathbf{U}$ and $\mathbf{V}$ are the same. We call the new pair IMEX-DIMSIM-2B.

3.5.2 Three-stage, third-order pairs with $p = q = r = s = 3$

We construct two implicit-explicit pairs named IMEX-DIMSIM-3A and IMEX-DIMSIM-3B starting from two existing implicit methods. All coefficients are obtained from the numerical solution of order conditions using Mathematica. The calculations are performed with 24 digits of accuracy such as to reduce the impact of roundoff errors on the resulting coefficient values.

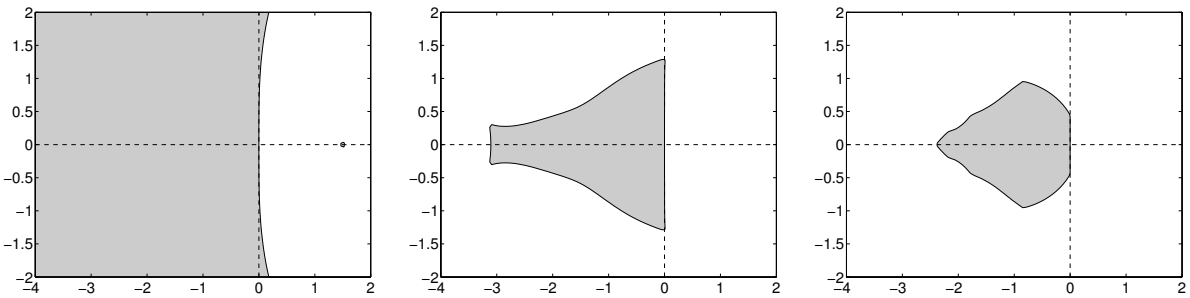
IMEX-DIMSIM-3A. According to [88] there are five A-stable type 2 DIMSIMs with the choice $\lambda = 1/2$ and $\mathbf{c} = [0, 1/2, 1]^T$. We select the implicit component in Table 3.1 which

[1] http://www.mathworks.com/matlabcentral/fileexchange/18593-differential-evolution



(a) Stability region $\widehat{S}$ of the implicit method (b) Stability region S of the explicit method (c) Constrained stability region $\widehat{\mathcal{S}}_\alpha$ (3.49) for $\alpha = 90°$

Figure 3.2: Stability regions for the IMEX-DIMSIM-3A pair of schemes



(a) Stability region $\widehat{S}$ of the implicit method (b) Stability region S of the explicit method (c) Constrained stability region $\widehat{\mathcal{S}}_\alpha$ (3.49) for $\alpha = 90°$

Figure 3.3: Stability regions for the IMEX-DIMSIM-3B pair of schemes

has a balanced set of coefficients.

The explicit component is obtained by a numerical maximization of the constrained stability region, as discussed in the previous section. The resulting coefficients are shown in Table 3.2. The IMEX stability regions are drawn in Figure 3.2.

IMEX-DIMSIM-3B. The choice of $\lambda = 0.435866521508459$ and $\mathbf{c} = [0, 1/2, 1]^T$ leads to the L-stable type 2 DIMSIM reported in [88]. The coefficients of the implicit component and the explicit component are presented in Table 3.3 and 3.4 respectively. The IMEX stability regions are drawn in Figure 3.3.

3.6 Numerical results

We test the IMEX-GLM methods on two test problems. The first one is the van der Pol equation, a commonly used small nonlinear ODE system that emphasizes convergence under

stiffness. The second test is a PDE problem arising in atmospheric modeling. We implemented our algorithms in a discontinuous Galerkin finite element model developed by Blaise et al. [45], which has efficient parallel scalability. We report the results obtained with IMEX-DIMSIM-2B and IMEX DIMSIM-3B methods, since they have the better accuracy and stability properties among their peers of the same order. Both IMEX Runge-Kutta methods and IMEX BDF methods are included for comparison.

3.6.1 Van der Pol equation

We consider the nonlinear van der Pol equation with a split right hand side

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = f(y,z) + g(y,z) = \begin{bmatrix} z \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \left((1-y^2)z - y\right)/\varepsilon \end{bmatrix} \tag{3.55}$$

on the time interval $[0, 0.5]$, with initial values

$$y(0) = 2, \quad z(0) = -\frac{2}{3} + \frac{10}{81}\varepsilon - \frac{292}{2187}\varepsilon^2 - \frac{1814}{19683}\varepsilon^3 + \mathcal{O}(\varepsilon^4). \tag{3.56}$$

We consider $\varepsilon = 10^{-6}$, a stiff case in which many methods suffer from order reduction [26].

The initialization (3.39) was done using the analytic derivatives. The reference solution is obtained with Radau-5, a stiffly accurate method [51], with very tight tolerances of $atol = rtol = 5 \times 10^{-15}$. We compare the new methods with IMEX-DIRK$(3,4,3)$, a L-stable three-stage third-order IMEX Runge-Kutta method proposed in [32], and IMEX-BDF3, a third-order IMEX BDF method [30].

Figure 3.4 shows the global error, measured in the L_2 norm, against step size h. A geometric sequence of step sizes, τ, $\tau/2$, $\tau/4$ and so on, were used. Order reduction can be clearly observed for the IMEX Runge-Kutta method, which yields second-order convergence. The IMEX DIMSIM converges at the theoretical third order and gives more accurate result than the other two methods compared when same step size is applied. Second-order IMEX DIMSIMs also produced no order reduction; detailed results have been reported in [81]. These results indicate that the high stage order of IMEX DIMSIMs make them particularly attractive for solving stiff problems where Runge-Kutta methods may suffer from order reduction, and IMEX DIMSIMs are also favourable for obtaining high accuracy with relatively large time steps.

3.6.2 Gravity waves

To assess the potential of the IMEX-DIMSIM schemes for solving partial differential equations, we consider the simulation of an idealized atmospheric phenomena: the propagation of a two-dimensional inertia-gravity wave [89]. Such a phenomena can be described by the

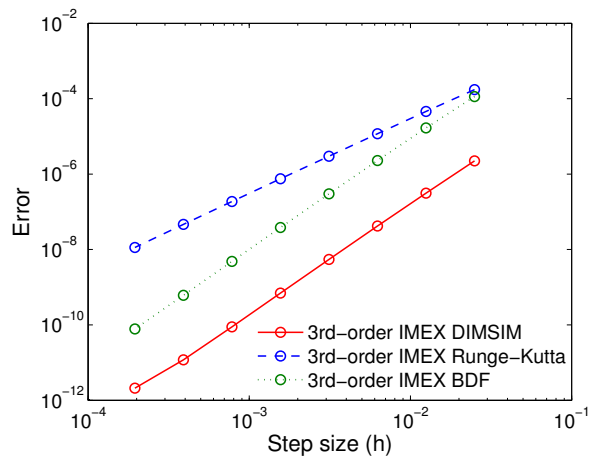


Figure 3.4: Convergence results for third-order IMEX schemes on the van der Pol equation.

compressible Euler equations, whose formulation is slightly modified to account for non-hydrostatic atmospheric processes [90]:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u} + p\mathbf{I}) &= -\rho g \widehat{\mathbf{e}}_{\mathbf{z}} \\ \frac{\partial \rho \theta}{\partial t} + \nabla \cdot (\rho \theta \mathbf{u}) &= 0\,, \end{aligned} \tag{3.57a}$$

where ρ is the density, $\mathbf{u}$ is the two-dimensional $xz-$velocity, θ is the potential temperature, and $\mathbf{I}$ is a 2×2 identity matrix. The gravitational acceleration is denoted g while $\widehat{\mathbf{e}}_{\mathbf{z}}$ is a unit vector pointing upwards. The prognostic variables are ρ, $\rho\mathbf{u}$ and $\rho\theta$. The pressure p in the momentum equation is computed by the equation of state

$$p = p_0 \left(\frac{\rho \theta R_d}{p_0} \right)^{\frac{c_p}{c_v}}, \tag{3.57b}$$

where $p_0 = 10^5$ Pa is the surface pressure, R_d is the gas constant, while c_p and c_v are the specific heat of the air for constant pressure and volume. For the details on sptial discretization, maintaining the hydrostatic state and performing the IMEX splitting for the temporal integration, the reader is referred to Chapter 4 Section 4.4.3

The inertia-gravity wave test-case is described in [89]. It is started with an initial atmosphere of constant horizontal velocity $u_x = 20$ ms^{-1} and constant Brunt-Väisälä frequency $N = g\frac{\mathrm{d}(\ln\bar{\theta})}{\mathrm{d}z} = 10^{-2}\mathrm{s}^{-1}$ in a channel of length $L = 300$ km and height $H = 10$ km. The waves are excited by a initial perturbation of the potential temperature

$$\theta_{\text{pert}} = \Delta\theta_0 \frac{\sin(\pi z/H)}{1 + (x - x_c)^2/a^2}, \tag{3.58}$$

with $\Delta\theta_0 = 0.01$ °C, $a = 5$ km and x_c is located at 100 km at the right of the left boundary. Figure 3.5 shows the initial solution and computational mesh with horizontal and vertical resolutions of respectively 5 km and 1.1 km. Third-order polynomials are used on each element, corresponding to actual resolutions of about 1.7 km in the horizontal direction and 0.4 km in the vertical direction. Radiative boundary conditions are considered for vertical boundaries, while non-flux (i.e. wall) boundary conditions are used along the bottom and top boundaries. Once the simulation is started, initial gravity waves are triggered by the initial perturbation of the potential temperature and propagate towards the lateral boundaries (Figure 3.5). The background velocity field has an influence on the solution by translating the perturbation towards the right of the domain.

All the experiments are performed on a workstation with 4 Intel Xeon E5-2630 Processors (24 cores in total) using 12 MPI threads. The parallelization is performed via a decomposition of the spatial domain. However, it has an influence upon the efficiency of the time-stepping because of the implicit system to solve which is distributed among the different processors. Note that it would be possible to keep the elements sharing the same column on the same processor and only treat the vertical dynamics implicitly. Despite resulting in a more restrictive time step condition, this technique would allow each implicit system to be solved locally, one system corresponding to a column of element.

Here we compare the performance of IMEX methods for a simulation window of 30 seconds. The second order methods are IMEX-DIMSIM-2B and L-stable, two-stage, second-order IMEX DIRK$(2, 3, 2)$ [32]. The third order methods are IMEX-DIMSIM-3B and IMEX DIRK$(3, 4, 3)$ [32]. The integrated L_2 errors for all prognostic variables are measured against a reference solution. The reference solution was obtained by applying an explicit RK method to solve the original (non-split) model with a very small time step $h = 0.005$.

The error versus computational effort diagrams are shown in Figure 3.6. All the methods display the theoretical orders of convergence. IMEX DIMSIMs and IMEX RK methods perform similarly, with IMEX DIMSIMs yielding slightly better accuracy when the same time steps are chosen. Also, IMEX DIMSIMs are slightly more efficient in terms of CPU time than the IMEX RK methods of the same order. Note that the termination procedure has been applied after each each time step to recovered the solution. The implementation can be optimized such as to apply the termination procedure only once at the end of the simulation; this would result in additional savings in computational cost. As the order increases, the number of stages required by an IMEX RK method grows rapidly due to order conditions, while an IMEX DIMSIM typically uses a number of stages equal to its order. Consequently, we expect that IMEX DIMSIM methods will become even more competitive for higher orders.

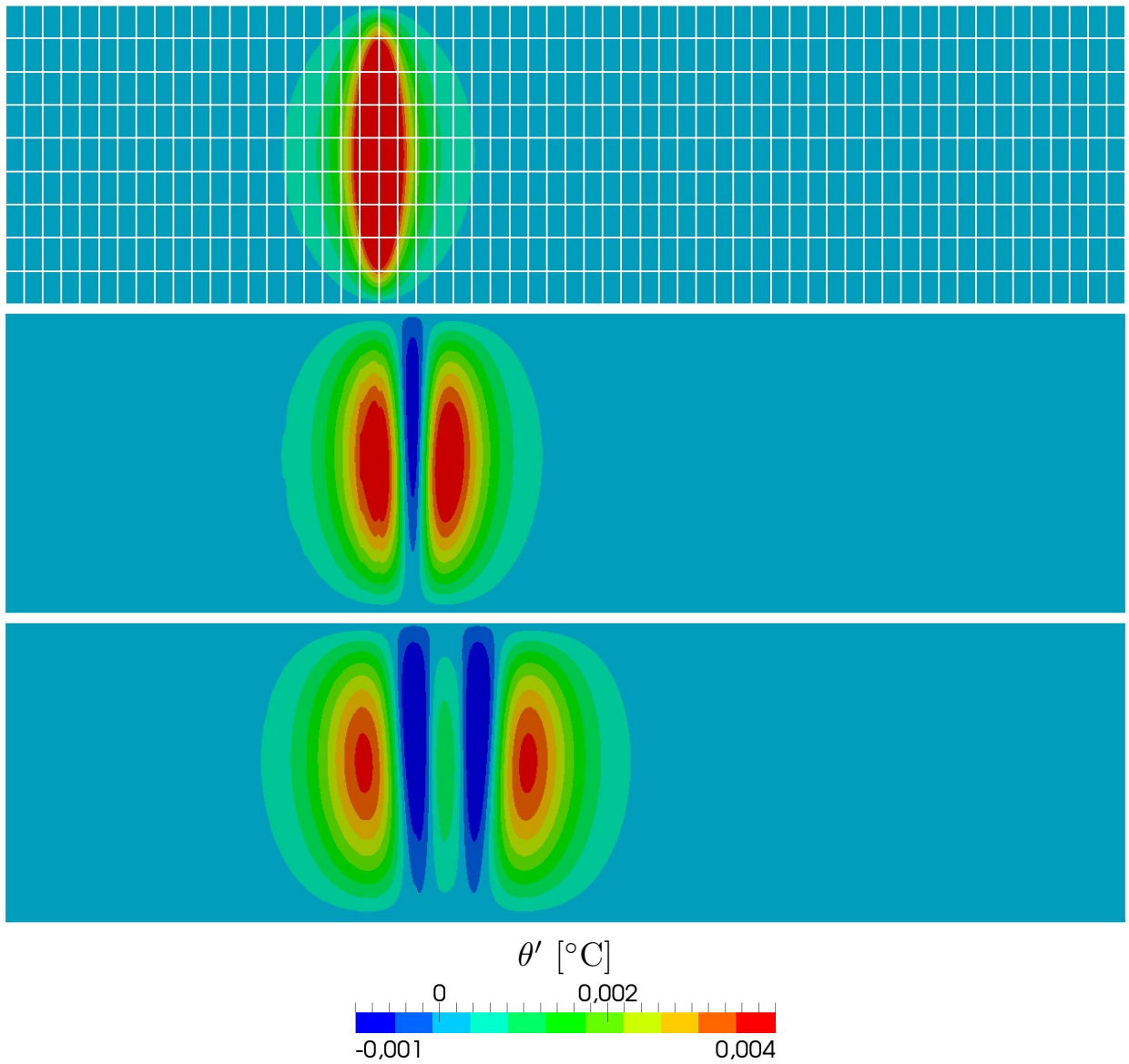
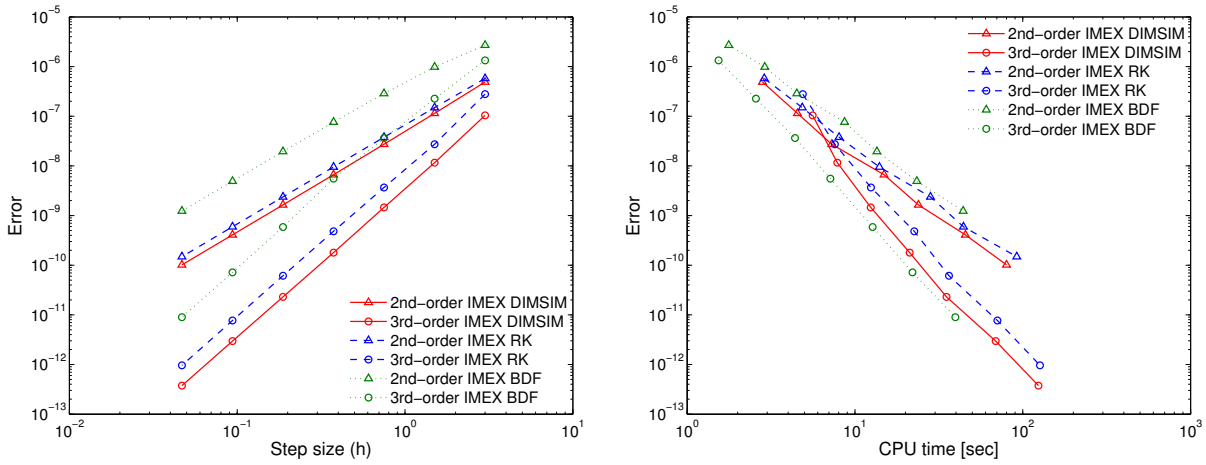


Figure 3.5: Evolution of the gravity wave: perturbation of the potential temperature at the initial time (TOP), after 450 seconds (MIDDLE) and after 900 seconds (BOTTOM). The computational mesh is visible on the first panel. The results are obtained with a third-order discontinuous Galerkin space discretization and third-order IMEX DIMSIM time integration. For visualization purposes, a stretch is applied in the vertical direction.



(a) Convergence (b) Work-precision diagram

Figure 3.6: Integrated L_2 errors against time steps (a) and CPU time (b) for different IMEX schemes. The errors are computed after 30 s of simulation. A geometric sequence of step sizes, τ, $\tau/2$, $\tau/4$ and so on, is used ($\tau = 4$).

3.7 Conclusions

In this chapter we introduce a new family of partitioned time integration methods based on high stage order general linear methods. We prove that the general linear framework is well suited for the construction of multi-methods (composite methods). Specifically, owing to the high stage orders, no coupling conditions are needed to ensure the order of accuracy of the partitioned GLM.

We apply the partitioned general linear framework to construct new implicit-explicit GLM pairs, together with appropriate starting and ending procedures. The linear stability analysis proposes the use of constrained stability functions to quantify the joint stability of the IMEX pair. A Prothero-Robinson convergence analysis reveals that the order of an IMEX GLM scheme on very stiff problems is dictated by the stage order of its non-stiff component; in particular, no order reduction appears if the explicit method has a full stage order. This result indicates that IMEX GLMs are particularly attractive for solving stiff problems, where other multistage methods may suffer from order reduction.

We discuss the construction of practical IMEX GLM pairs starting from known implicit schemes and adding an appropriate explicit counterpart. This strategy is applied to build second and third order IMEX diagonally-implicit-explicit multi-stage integration methods. Numerical experiments with the van der Pol equation confirm the fact that IMEX GLMs converge at full order while IMEX RK methods suffer from order reduction. The two dimensional gravity wave system is an important step towards solving real PDE-based problems. The new IMEX-DIMSIM schemes perform slightly better than the IMEX RK methods of

the same order.

Table 3.1: Coefficients of the implicit method of the IMEX-DIMSIM-3A pair.

0.5	0	0	1	0	0
0.200835027145109	0.5	0	0	1	0
−1.30998408899641	1.01685248853025	0.5	0	0	1
1.01640094894605	0.632229903531054	−0.408057475882764	0.910428360600012	0.358564648055175	−0.268993008655188
0.724734282279383	1.46556323686439	−0.6505591694540	0.910428360600012	0.358564648055175	−0.268993008655188
−0.333784872917534	4.34945403578847	−1.481964185810437	0.910428360600012	0.358564648055175	−0.268993008655188

Table 3.2: Coefficients of the explicit method of the IMEX DIMSIM-3A pair.

0	0	0	1	0	0
0.773142038041842	0	0	0	1	0
−0.574721803854933	1.40234019763932	0	0	0	1
0.568615416356845	0.349254080830621	0.226439028444830	0.910428360600012	0.358564648055175	−0.268993008655188
0.776948749690179	−0.317412585836046	0.411630323736322	0.910428360600012	0.358564648055175	−0.268993008655188
0.332941885384188	1.22294134041526	−0.239193093951542	0.910428360600012	0.358564648055175	−0.268993008655188

Table 3.3: Coefficients of the implicit method of the IMEX-DIMSIM-3B pair.

0.435866521508459	0	0	1	0	0
0.250514880897719	0.435866521508459	0	0	1	0
−1.211594287777006	1.00127459988119	0.435866521508459	0	0	1
0.833790728250125	0.645998912146314	−0.315827085512970	0.552090962040363	0.734856659871292	−0.286947621911655
0.606257540075000	1.28693181000502	−0.479741676094274	0.552090962040363	0.734856659871292	−0.286947621911655
−0.308416769489771	3.80342155052421	−1.12072253825515	0.552090962040363	0.734856659871292	−0.286947621911655

Table 3.4: Coefficients of the explicit method of the IMEX DIMSIM-3B pair.

0	0	0	1	0	0
0.753076872681821	0	0	0	1	0
−0.4897243738259477	1.28728279647947	0	0	0	1
0.755324932592235	0.24363012413977	0.245110297813246	0.552090962040363	0.734856659871292	−0.286947621911655
0.963658265925568	−0.423036542526896	0.450366758464759	0.552090962040363	0.734856659871292	−0.286947621911655
0.634708802779431	0.772145180244847	0.0396529488674508	0.552090962040363	0.734856659871292	−0.286947621911655

Chapter 4

High order general linear methods

4.1 Introduction

Many problems in science and engineering are modeled by time-dependent systems of equations involving both stiff and nonstiff terms. Examples include advection-diffusion-reaction equations, fluid-structure interactions, and Navier-Stokes equations, and arise in application areas such as mechanical and chemical engineering, astrophysics, meteorology and oceanography, and environmental science.

A method-of-lines approach is frequently employed to separate the spatial and temporal terms in the governing partial differential equations. After the spatial terms are discretized by techniques such as finite differences, finite volumes, and finite elements, the resulting system of ordinary differential equations (ODEs) is integrated in time. Stiffness may result from different time scales involved (e.g., convective versus acoustic waves), from local processes such as chemical reactions, and from grids with complex geometry [91].

Explicit numerical integration schemes have maximum allowable time steps bounded by the fastest time scales in the system; for example, the time steps are restricted by the CFL stability condition. Implicit integration schemes can avoid the step size restrictions but require the solution of large nonlinear systems at each step, and are therefore computationally expensive. It is therefore of considerable interest to construct numerical integration schemes that avoid the time step restrictions while maintaining a high computational efficiency. In the implicit-explicit (IMEX) framework computational efficiency is achieved by performing an implicit integration only for the stiff components of the system.

High order methods usually qualify more accuracy and better efficiency than low order methods. Many modern PDE solvers are able to employ high order spatial discretizations, e.g., by using high degree polynomials in a discontinuous Galerkin (DG) approach. There is a need to develop high order time stepping formulas to be used in conjunction with high

order spatial discretizations. This need motivates the current work.

Existing high order IMEX methods face challenges when applied to practical problems. High order IMEX linear multistep methods suffer from a marked reduction of the stability region with increasing order. IMEX Runge-Kutta methods are known to suffer from possible order reduction for stiff problems, which reduces the efficiency of high order methods to that of low order methods. The order reduction of the former could be avoided by incorporating additional order conditions [92]. Some possible remedies for the latter for Runge-Kutta methods have also been proposed in the literature [93]. However, these strategies require special treatment of boundaries which may bring in extra computational cost and complexity; in addition, some of them only work for special cases such as linear boundary conditions. To the best of our knowledge, there is no effective way for IMEX Runge-Kutta methods to handle the order reduction in a general way. Furthermore, the considerable increase in the number of coupling conditions makes the construction of high order methods difficult.

This work develops and tests new high order time stepping schemes in the framework of implicit-explicit general linear methods (IMEX-GLMs) that we have recently developed [94, 81]. In our earlier work[94, 81] we have developed second and third-order IMEX-GLM schemes that showed considerable promise.

This study develops fourth and fifth order IMEX-GLMs with optimized stability properties. Numerical experiments confirm that these methods do not suffer from order reduction, and are considerably more efficient than IMEX-RK methods on a suite of problems ranging from two-dimensional Allen-Cahn and Burgers' equations to three-dimensional compressible Euler equations.

This chapter is organized as follows. Section 4.2 reviews the class of general linear methods. The construction of high order IMEX-GLMs with desired stability properties is discussed in Section 4.3. This section first introduces desirable stability properties building upon existing stability theory for Runge-Kutta methods. Numerical results are reported in Section 4.4. Conclusions are drawn in Section 4.5.

4.2 IMEX general linear methods

IMEX time stepping methods are used to solve systems of ordinary differential equations (ODEs) of the form

$$y' = f(t, y) + g(t, y) \quad t_0 \leq t \leq t_F\,, \quad y(t_0) = y_0 \in \mathbb{R}^d\,, \tag{4.1}$$

where f is a nonstiff term, and g is a stiff term. Many systems of partial differential equations (PDEs) solved in the methods of lines framework lead to partitioned ODE systems (4.1) after semi-discretization in space. The nonstiff and stiff driving physical processes are captured by f and g, respectively.

Partitioned and IMEX general linear methods are described in Chapter 3. An implicit-explicit general linear method applied to (4.1) advances the solution for one step using:

$$Y_i = h\sum_{j=1}^{i-1} a_{i,j}\, f(Y_j) + h\sum_{j=1}^{i} \widehat{a}_{i,j}\, g(Y_j) + \sum_{j=1}^{r} u_{i,j}\, y_j^{[n-1]}\,, \quad i = 1,\ldots,s, \tag{4.2a}$$

$$y_i^{[n]} = h\sum_{j=1}^{s} \Big(b_{i,j}\, f(Y_j) + \widehat{b}_{i,j}\, g(Y_j)\Big) + \sum_{j=1}^{r} v_{i,j}\, y_j^{[n-1]}\,, \quad i = 1,\ldots,r\,. \tag{4.2b}$$

Such a method is denoted IMEX-GLM(p,q,s,r) (p,q,s and r stand for order, stage order, number of internal stages, and number of external stages, respectively). The implicit and the explicit components share the same abscissa vector $\mathbf{c}$ and the same coefficients $\mathbf{U}$ and $\mathbf{V}$. The IMEX-GLM (4.2) is represented compactly by the Butcher tableau

$$\begin{array}{c|c|c|c} \mathbf{c} & \mathbf{A} & \widehat{\mathbf{A}} & \mathbf{U} \\ \hline & \mathbf{B} & \widehat{\mathbf{B}} & \mathbf{V} \end{array}\,. \tag{4.3}$$

To study the method (4.2) in [94, 81] the additively partitioned original system (4.1) is written in an equivalent component partitioned form:

$$y = x + z\,, \tag{4.4a}$$

$$x' = \tilde{f}(x,z) = f(x+z)\,, \tag{4.4b}$$

$$z' = \tilde{g}(x,z) = g(x+z)\,. \tag{4.4c}$$

The external vector $y_i^{[n-1]}$ is defined as a pth-order approximation of linear combinations of derivatives

$$y_i^{[n-1]} = \sum_{k=0}^{r} q_{i,k} h^k x^{(k)}(t_{n-1}) + \sum_{k=0}^{r} \widehat{q}_{i,k} h^k z^{(k)}(t_{n-1}) + \mathcal{O}(h^{p+1}), \quad i = 1,\ldots,r, \tag{4.5}$$

for some real parameters $q_{i,k}$, $i = 1,\ldots,r$, $k = 0,1,\ldots,p$. Note that in (4.2) $x_i^{[n]}$ and $z_i^{[n]}$ need not to be known individually once they are initialized in the first step. Only the combined external vector $y_i^{[n]} = x_i^{[n]} + z_i^{[n]}$ is advanced at each step, similar to how regular GLMs proceed.

To initialize $y_i^{[0]}$ the starting procedure developed in [94] advances the ODE solution by taking $r-1$ steps with a small step size τ to obtain the solutions $y_0, y_1^{\text{start}}, \ldots, y_{r-1}^{\text{start}}$. The derivative terms are approximated using only the function evaluations at these r points. The starting value for the external vector $y_i^{[0]}$ is calculated via the formula

$$y_i^{[0]} = y_0 + q_{i,1} h f(y_0) + \widehat{q}_{i,1} h g(y_0)$$

$$+\sum_{k=2}^{r}\sum_{j=1}^{r} q_{i,k}h^k/\tau^{k-1}d_{k,j}f\left(y_j^{\text{start}}\right)+\sum_{k=2}^{r}\sum_{j=1}^{r}\widehat{q}_{i,k}h^k/\tau^{k-1}d_{k,j}g\left(y_k^{\text{start}}\right).$$

In vector form it can be written as

$$y^{[0]}=\mathbf{1}_r\otimes y_0+\tau\left(\mathbf{QD}\otimes\mathbf{I}_{d\times d}\right)\left((\mathbf{R}\otimes\mathbf{I}_{d\times d})F^{\text{start}}\right)+\tau\left(\widehat{\mathbf{Q}}\mathbf{D}\otimes\mathbf{I}_{d\times d}\right)\left((\mathbf{R}\otimes\mathbf{I}_{d\times d})G^{\text{start}}\right), \quad (4.6)$$

where F^{start} and G^{start} consist of function values evaluated at the r starting points, e.g. $F^{\text{start}}=\left[f\left(y_0^{\text{start}}\right),f\left(y_1^{\text{start}}\right),\cdots,f\left(y_{r-1}^{\text{start}}\right)\right]^T$.

The $r\times r$ coefficient matrices $\mathbf{Q}$, $\mathbf{D}$, and $\mathbf{R}$ are computed as follows.

1. $\mathbf{Q}$, $\widehat{\mathbf{Q}}$ are determined by the method coefficients $\mathbf{A}$, $\widehat{\mathbf{A}}$ and the abscissa vector $\mathbf{c}$. These matrices can be computed column-wise via the order conditions [22]

$$q_0=\mathbf{1}_s,\quad q_i=\frac{\mathbf{c}^i}{i!}-\frac{\mathbf{A}\,\mathbf{c}^{i-1}}{(i-1)!};\quad \widehat{q}_0=\mathbf{1}_s,\quad \widehat{q}_i=\frac{\mathbf{c}^i}{i!}-\frac{\widehat{\mathbf{A}}\,\mathbf{c}^{i-1}}{(i-1)!}. \quad (4.7)$$

2. Starting with the following approximation

$$\begin{bmatrix}\tau x'(t_0)\\ \tau^2x''(t_0)\\ \vdots\\ \tau^r x^{(r)}(t_0)\end{bmatrix}=\tau\mathbf{D}\begin{bmatrix}x'(t_0)\\ x'(t_1)\\ \vdots\\ x'(t_{r-1})\end{bmatrix}+\mathcal{O}(\tau^{r+1}), \quad (4.8)$$

   expanding the right hand side in Taylor series, and comparing the coefficients of each term, allows to identify each entry of $\mathbf{D}$.

3. $\mathbf{R}$ is a diagonal rescaling matrix which has the form

$$\mathbf{R}=\operatorname{diag}\left(h/\tau,h^2/\tau^2,\ldots,h^r/\tau^r\right). \quad (4.9)$$

Note that this starting procedure enables to compute the initial approximations with a smaller step size $\tau\le h$. The initial approximations can be computed with a regular method of choice; the very small time steps ensure accurate initial solutions, and also circumvent possible numerical stability issues with the auxiliary scheme. The starting procedure used for the experiments in this chapter employs the IMEX-RK scheme. Considering the possible low accuracy caused by order reduction, in the starting procedure we use a step size half as large as the step size for the following integration. We point out that using the same step size typically works well based on our experience.

4.3 Construction of high order IMEX-GLMs

We now consider the construction oh high order IMEX-GLMs. The partitioned GLM theory described in Chapter 3 ensures that, if the stage order is high, the IMEX-GLM method has the desired order without the need for coupling conditions. One imposes the order and stage order conditions independently on the implicit and on the explicit component GLMs.

The order conditions for constructing arbitrary GLMs are complicated. In this chapterr we choose the explicit and implicit components from a subclass of GLMs —- diagonally implicit multistage integration methods (DIMSIMs), for which the order conditions are more manageable.

Following Chapter 3 we are particularly interested in DIMSIMs with $p = q = r = s$, $\mathbf{U} = \mathbf{I}_{s\times s}$, and $\mathbf{V} = \mathbf{1}_s\, v^T$, where $v^T\, \mathbf{1}_s = 1$ [24]. The order conditions are satisfied if the coefficient matrix $\mathbf{B}$ is computed from the relation (3.13). Therefore to obtain high order DIMSIMs there is no need to solve complex nonlinear systems as one usually does in the construction of Runge-Kutta methods.

The important challenge that remains in the construction of IMEX-GLM methods is to achieve the desirable stability properties. This section first introduces desirable stability properties building upon existing stability theory for Runge-Kutta methods. A numerical optimization process used to maximize the IMEX stability regions is then discussed. Two new IMEX-DIMSIM methods of orders four and five are presented at the end.

4.3.1 Stability considerations

A-stability, L-stability, and inherent Runge-Kutta stability The classical linear stability theory [51] considers the scalar test problem whose solution decays to zero

$$y' = \lambda y\,, \quad t \geq 0\,, \quad Re(\lambda) \leq 0\,. \tag{4.10}$$

A numerical method is stable if, when applied to solve the test problem (4.10) for one step of length h it generates a solution of non-increasing size. A GLM $(\mathbf{A}, \mathbf{B}, \mathbf{U}, \mathbf{V})$ (4.3) applied to the test problem gives a solution

$$y^{[n+1]} = \mathbf{M}(z)\, y^{[n]}\,, \quad \mathbf{M}(z) = \mathbf{V} + z\,\mathbf{B}\,\left(\mathbf{I}_{s\times s} - z\mathbf{A}\right)^{-1}\,\mathbf{U}\,. \tag{4.11}$$

Here $\mathbf{M}(z)$ is the stability matrix and has a corresponding stability function

$$p(w, z) = \det(w\mathbf{I}_{r\times r} - \mathbf{M}(z)), \tag{4.12}$$

where $w, z \in \mathbb{C}$ and $z = \lambda h$.

A-stability requires that the method is unconditionally stable independent of the size of the time step h, i.e., the spectral radius of the stability matrix $\rho(\mathbf{M}(z)) \leq 1$ for any z. L-stability

further requires that $\rho(\mathbf{M}(z)) \to 0$ when $z \to \infty$ [47]. L-stable methods damp components of high frequencies and are particularly useful for stiff problems. Since IMEX-GLM schemes are designed to treat stiff parts of a given problem implicitly, we want the implicit component to be L-stable, or at least A-stable. Imposing L-stability directly on the GLM coefficients leads to a difficult analysis, with complexity increasing dramatically as the order increases.

The inherent Runge-Kutta stability property [84, 85] provides a practical way to achieve L-stability. This property requires that the stability function (4.12) has the form

$$p(w, z) = w^{s-1}\left(w - R(z)\right), \tag{4.13}$$

where $R(z)$ is the stability function of a Runge Kutta method of order $p = s$. When (4.13) holds the existing L-stability theory for Runge Kutta methods can be applied to GLMs. Note that conditions (4.13) lead to additional nonlinear constraints on method coefficients; these constraints need to be solved accurately in practice.

Stability analysis for IMEX-GLMs To study the linear stability of IMEX-GLM schemes we consider the following generalized linear test equation [94]

$$y' = \xi y + \widehat{\xi} y\,, \quad t \geq 0, \quad Re(\xi), Re(\widehat{\xi}) \leq 0\,. \tag{4.14}$$

This test problem mimics the structure of (4.1). We consider ξy to be the nonstiff term and $\widehat{\xi} y$ the stiff term, and denote $w = h\xi$ and $\widehat{w} = h\widehat{\xi}$.

Applying (4.2) to the test equation (4.14) and assuming $\mathbf{I}_{s\times s} - w\mathbf{A} - \widehat{w}\widehat{\mathbf{A}}$ is nonsingular lead to

$$y^{[n]} = \mathbf{M}(w, \widehat{w})\, y^{[n-1]},$$

where the stability matrix is defined by [94]

$$\mathbf{M}(w, \widehat{w}) = \mathbf{V} + \left(w\,\mathbf{B} + \widehat{w}\,\widehat{\mathbf{B}}\right)\left(\mathbf{I}_{s\times s} - w\,\mathbf{A} - \widehat{w}\,\widehat{\mathbf{A}}\right)^{-1}\mathbf{U}\,. \tag{4.15}$$

Let $S \subset \mathbb{C}$ and $\widehat{S} \subset \mathbb{C}$ be the stability regions of the explicit GLM component and of the implicit GLM component, respectively. The *combined stability region* is defined by [94]

$$\mathcal{C} = \left\{ w \in S,\, \widehat{w} \in \widehat{S} \;:\; \rho\big(\mathbf{M}(w, \widehat{w})\big) < 1 \right\} \subset S \times \widehat{S} \subset \mathbb{C} \times \mathbb{C}\,. \tag{4.16}$$

For a practical analysis of stability we define a *desired stiff stability region*, e.g.,

$$\widehat{\mathcal{S}}_\alpha = \{\widehat{w} \in \widehat{S} \cap \mathbb{C}^- \;:\; |\mathrm{Im}(\widehat{w})| < \tan(\alpha)\,|\mathrm{Re}(\widehat{w})|\}\,,$$

and compute numerically the corresponding *constrained non-stiff stability region*:

$$\mathcal{S}_\alpha = \left\{ w \in S \;:\; \rho\big(\mathbf{M}(w, \widehat{w})\big) < 1\,, \quad \forall\, \widehat{w} \in \widehat{\mathcal{S}}_\alpha \right\}\,. \tag{4.17}$$

The IMEX-GLM method is stable if the constrained non-stiff stability region $\mathcal{S}_\alpha$ is non-trivial (has a non-empty interior) and is sufficiently large for a prescribed (problem-dependent) value of α, e.g., $\alpha = \pi/2$.

4.3.2 Finding high order IMEX-DIMSIMs with large stability regions

The implicit component of the IMEX-GLM is constructed first, and the desired L-stability property is imposed L-stable GLMs existing in the literature can also be used as implicit components in the combined IMEX scheme.

L-stability indicates that $\widehat{w}$ in the non-stiff stability definition (4.17) can be any value on the negative half-plane. So the constrained region with $\alpha = \pi/2$ is

$$\mathcal{S}_{\pi/2} = \Big\{ w \in S \ : \ \rho\big(\mathbf{M}(w, re^{i\theta})\big) < 1 \,, \ \ \forall\, \theta \in \Big[-\frac{\pi}{2}, \frac{\pi}{2}\Big] \,, \ \ \forall\, r \in [0, -\infty) \Big\} \,.$$

The corresponding explicit component is constructed next based on the following criteria: it shares the coefficients $\mathbf{c}, \widehat{\mathbf{U}}, \widehat{\mathbf{V}}$ with the implicit component; it satisfies the desired order conditions; and results in a large constrained stability region (4.17).

According to the order conditions in [94], $\mathbf{B}$ depends on $\mathbf{A}$ and $\mathbf{c}$. Thus the only free parameters in determining the explicit part are the $s(s-1)/2$ elements of matrix $\mathbf{A}$. The problem of finding IMEX-DIMSIMs can be regarded as a numerical optimization problem to find the entries of $\mathbf{A}$ such as to maximize the area of the constrained stability region $\mathcal{S}_{\pi/2}$.

We discretize the region $\mathcal{S}_{\pi/2}$ using finite sets of points in polar coordinates

$$\mathcal{S}_{\pi/2} \approx \Big\{ w \in S \ : \ \rho\big(\mathbf{M}(w, re^{i\theta})\big) < 1 \,, \ \ \forall\, \theta \in \Theta_{\mathrm{f}} \subset \Big[-\frac{\pi}{2}, \frac{\pi}{2}\Big] \,, \ \ \forall\, r \in R_{\mathrm{f}} \subset (-\infty, 0] \Big\} \,.$$

For example, $R_{\mathrm{f}} = [0, -10^{-3}, -10^{-2}, \ldots, -10^{3}]$ and Θ_{f} are a set of equally spaced points between $-\pi/2$ and $\pi/2$.

We next determine the boundary $\partial\mathcal{S}_{\pi/2}$ of the constrained stability region. For this we consider the points of intersection of the boundary with vertical lines on the negative half-plane with abscissae x_k. An intersection point $\widetilde{w}_k = (x_k, y_k)$ should satisfy

$$\max_{r \in R_{\mathrm{f}}, \theta \in \Theta_{\mathrm{f}}} \rho\big(\mathbf{M}(\widetilde{w}_k, re^{i\theta})\big) = 1. \tag{4.18}$$

Note that since the stability region is symmetric, we only need to consider the part above the real axis.

Starting with an initial point on the vertical line, e.g. $x_k + i\, y_*$ where y_* is large enough to make the point outside the stability region, we apply the bisection Algorithm 1 to find the first point $\widetilde{w} = x_k + i\, y$ along the vertical line such that

$$\max_{r \in R_{\mathrm{f}}, \theta \in \Theta_{\mathrm{f}}} \rho\big(\mathbf{M}(\widetilde{w}, re^{i\theta})\big) < 1. \tag{4.19}$$

A similar idea can be used to find the intersection of the stability region and the real axis, which is assumed to be the leftmost point of the stability region. Then we can determine

Algorithm 1 Bisection algorithm for finding the points of intersection

```
Initialize y_top ← y_0  y_bot ← 0
while  y_top − y_bot > tol  do
    y_mid = (y_top + y_bot)/2
    if  w̃ ← c + i y_mid satisfies the condition (4.19)  then
        y_bot = y_mid
    else
        y_top = y_mid
    end if
end while
return y_bot
```

Algorithm 2 Algorithm for computing the area of constrained stability regions

1: Find the point x_b of intersection of the stability region and the x axis using a bisection strategy similar to Algorithm 1
2: Generate m vertical lines with abscissae x_k linearly spaced between x_b and 0
3: Find the points of intersection of these lines and the stability region
4: Approximate the area of the stability region using the trapezoidal method

the boundary with the above-mentioned algorithm. Algorithm 2 summarizes the procedure to approximate the area of the stability region.

As we can see, the objective function that approximates the area of the stability region is highly nonlinear and computationally expensive, especially for the construction of high order methods. The optimization problem is in general difficult to solve numerically. First we transform the maximization problem to a minimization problem by minimizing the negative of the objective function. Then we use the combination of MATLAB genetic algorithm function, `ga` and MATLAB local minimizer `fminsearch`. We repeatedly apply the two optimization routines one after another using one's result as the starting point of the other. Each optimizer is run multiple times until the results converge; each run is initialized with the previous result. We terminate the procedure when the result does not change across multiple runs for both optimizers.

4.3.3 New IMEX general linear methods

The construction of DIMSIMs starts with choosing the abscissa vector $\mathbf{c}$ [88]. A natural choice is a vector of values equally spaced in the interval $[0, 1]$. For DIMSIMs of order p and stage order $q = p$, the last value $\mathbf{c}_s = 1$ allows to use the last stage value as the ODE solution at the next time step. This advantage also applies to IMEX-DIMSIM. Here we choose the common abscissae for the IMEX pairs equally spaced in $[0, 1]$, and including 0 and 1. There is no evidence so far that other choices would lead to better schemes.

A fourth-order IMEX-DIMSIM pair

We start with the construction of the implicit part of the IMEX pair. Butcher [88] reports a failed attempt to construct DIMSIMs with inherent Runge-Kutta stability, $p = q = r = s = 4$, and $c = [0, 1/3, 2/3, 1]$. Surprisingly we succeeded in solving the nonlinear system comes from the stability constraints by using the Mathematica software. For the detailed information on the nonlinear system, we refer to [88]. The coefficients of the type 2 (implicit) DIMSIM we found are given in Table 4.1. The choice of the diagonal element of $\widehat{A}$ equal to 0.572816062482135 ensures that the implicit method $L-$stable, following the classic theory of Runge-Kutta methods [51]. We remark that this new implicit DIMSIM method can be used by itself due to its favorable stability properties.

The optimization problem formulated in Section 4.3.2 for maximizing the constrained stability regions has six free variables, the lower triangular entries the coefficient matrix $\mathbf{A}$. The maximal area of the constrained stability region of the explicit method on the negative plane is approximately 1.34. Figure 4.1 shows the stability regions of the implicit component $\widehat{S}$, of the explicit component S, as well as the constrained stability regions $\widehat{S}_\alpha$ for $\alpha = \pi/2, \pi/3, \pi/4$.

We will refer to the resulting method as IMEX-DIMSIM4. The coefficients of the explicit method to 15 accurate digits are given in Table 4.1.

A fifth-order IMEX-DIMSIM pair

An L-stable fifth-order type 2 (implicit) DIMSIM with $p = q = r = s = 5$ and $c = [0, 1/4, 1/2, 3/4, 1]$ was constructed by Butcher [95]. We have obtained its coefficients with improved accuracy from 6 to 15 decimal digits by solving the nonlinear conditions using the Levenberg-Marquardt algorithm implemented by MATLAB's routine `fsolve`.

The corresponding explicit component is obtained by the numerical optimization procedure described in the Section 4.3.2. The maximal area of the constrained stability region of the explicit method on the negative plane is approximately 0.83, and is smaller than the area of the fourth order pair. Figure 4.2 shows the stability regions of the implicit component $\widehat{S}$, of the explicit component S, as well as the constrained stability regions $\widehat{S}_\alpha$ for $\alpha = \pi/2, \pi/3, \pi/4$.

We will refer to the resulting method as IMEX-DIMSIM5. The coefficients of the method to 15 accurate digits are given in Table 4.2 (compare the implicit coefficients to [95]).

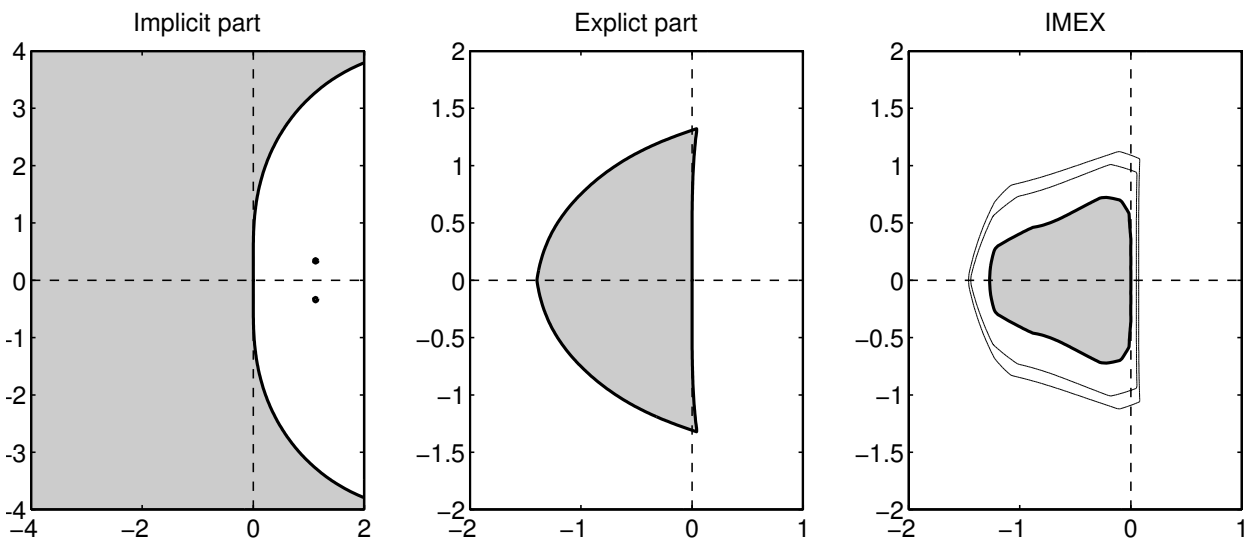


Figure 4.1: Stability regions for the fourth-order IMEX-DIMSIM pair with $p = q = r = s = 4$ and $c = [0, 1/3, 2/3, 1]$. From left to right are stability region $\widehat{S}$ of the implicit method, stability region S of the explicit method, and constrained stability regions $\widehat{S}_\alpha$ (with $\alpha = \pi/2, \pi/3, \pi/4$ from interior toward exterior, respectively).

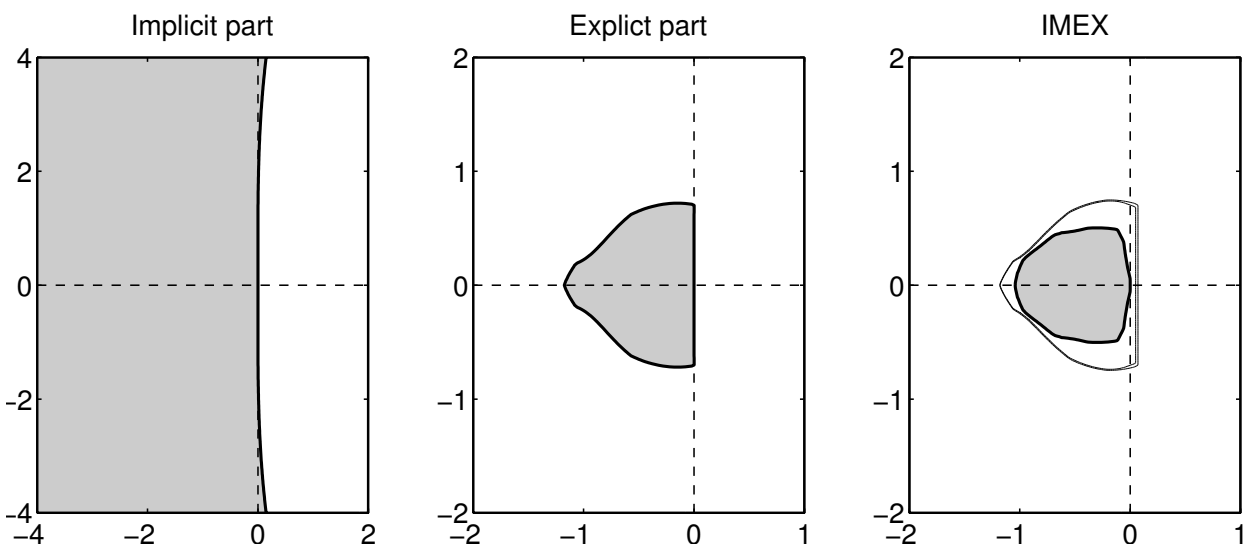


Figure 4.2: Stability regions for the fifth-order IMEX-DIMSIM pair with $p = q = r = s = 5$ and $c = [0, 1/4, 1/2, 3/4, 1]$. From left to right are stability region $\widehat{S}$ of the implicit method, stability region S of the explicit method, and constrained stability regions $\widehat{S}_\alpha$ (with $\alpha = \pi/2, \pi/3, \pi/4$ from interior toward exterior, respectively)

4.4 Numerical tests

We consider several test problems that are motivated by different application areas such as material science, fluid mechanics, and atmospheric modeling. All problems are governed by partial differential equations and contain both stiff components and nonstiff components. The first two test cases are implemented in MATLAB using finite difference schemes for space discretization. The time integration is performed with the two high order IMEX general linear methods IMEX-DIMSIM4 and IMEX-DIMSIM5. The performance of these methods is compared against two classic IMEX Runge-Kutta methods, ARK4(3)6L[2]SA and ARK5(4)8L[2]SA, from Kennedy and Carpenter [26]. We will refer to these methods as IMEX-RK4 and IMEX-RK5, respectively. Both IMEX Runge-Kutta methods have a stiffly-accurate implicit component and share the same abscissa $\mathbf{c} = \widehat{\mathbf{c}}$ as our IMEX-DIMSIMs do.

We have also implement the IMEX-DIMSIM schemes in the discontinuous Galerkin solver GMSH-DG [96] and applied them to the three-dimensional compressible Euler equations coming from multiscale nonhydrostatic atmospheric simulations.

All the experiments have been performed on a workstation with four Intel Xeon E5-2630 Processors. The goal is to assess the performance of the high order IMEX-DIMSIM and IMEX-RK methods on both two-dimensional and three-dimensional simulations.

4.4.1 Allen-Cahn equation

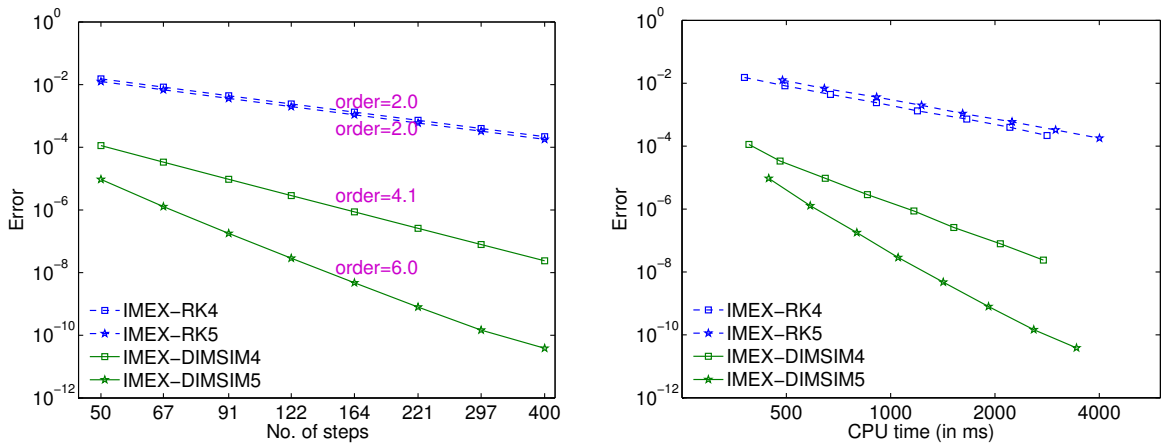
We consider the two-dimensional reaction-diffusion Allen-Cahn problem [97] which describes the process of phase transition in materials science.

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + \beta(u - u^3) + f, \quad 0 \le x, y \le 1\,, \quad 0 \le t \le 0.5, \tag{4.20}$$

where the parameters are $\alpha = 0.01$, $\beta = 3$, and $f(t, x, y)$ is a source term that is consistent with the exact solution $u(t, x, y) = 2 + \sin(2\pi(x - t))\cos(3\pi(y - t))$. Time varying Dirichlet boundary conditions (that represent the exact solution evaluated at the boundaries) are imposed. The spatial discretization is performed using a second-order central finite difference scheme on a uniform grid with $\Delta x = \Delta y = 1/40$.

Explicit time stepping methods have a maximal allowable time step $h \propto \Delta x^2$ due to the CFL condition related to diffusion. To overcome this limitation we treat the stiff diffusion term implicitly and the remaining terms explicitly. Since the discrete diffusion term is linear we perform a single LU factorization of the matrix $\mathbf{I} - h\gamma\mathbf{J}$ and reuse it throughout the simulation; here γ is a method coefficient and $\mathbf{J}$ is the Jacobian of the stiff diffusion.

The reference solution u_{ref} is obtained using MATLAB's routine `ode15s` with very tight tolerances $AbsTol = RelTol = 3 \times 10^{-14}$. The absolute solution error magnitude is measured



(a) Convergence diagram (b) Work-precision diagram

Figure 4.3: Comparison of high order IMEX-DIMSIM and IMEX-RK results for the 2D Allen-Cahn equation (4.20). Shown are the temporal discretization errors corresponding to the solution at the final time $t = 0.5$.

in the L_2 norm:

$$\mathbf{E} = \|u - u^{\text{ref}}\|_2. \tag{4.21}$$

Figure 4.3(a) shows the errors at the final time for solutions computed using different numbers of steps. The two IMEX-RK methods show a marked order reduction - to order two. There is no order reduction for the IMEX-DIMSIM schemes; IMEX-DIMSIM4 displays the theoretical order while IMEX-DIMSIM5 shows a higher convergence than the theoretical order. The IMEX-DIMSIMs give considerably more accurate results than the IMEX-RK methods for all step sizes tested. This is noteworthy since IMEX-DIMSIMs have fewer stages than the IMEX-RK methods of the same order and therefore require fewer function evaluations and linear solves per step. The corresponding work-precision diagrams of errors versus CPU time are shown in Figure 4.3(b) and reveal a sizable gap in efficiency between the two families of IMEX schemes. Figure 4.4 shows the spatial distribution of the absolute errors $|u_{\text{numerical}} - u_{\text{reference}}|$ at final time; this is only the temporal discretization error as we compare against a reference solution that uses the same spatial discretization. IMEX-RK methods give large errors near boundaries and relatively smaller errors in the interior of the domain are evenly distributed. The order reduction phenomenon of IMEX-RK methods originates with errors at the boundaries, but plague the whole domain as the time evolves. In contrast, IMEX-DIMSIMs handle the boundaries well and preserve the theoretical orders of convergence.

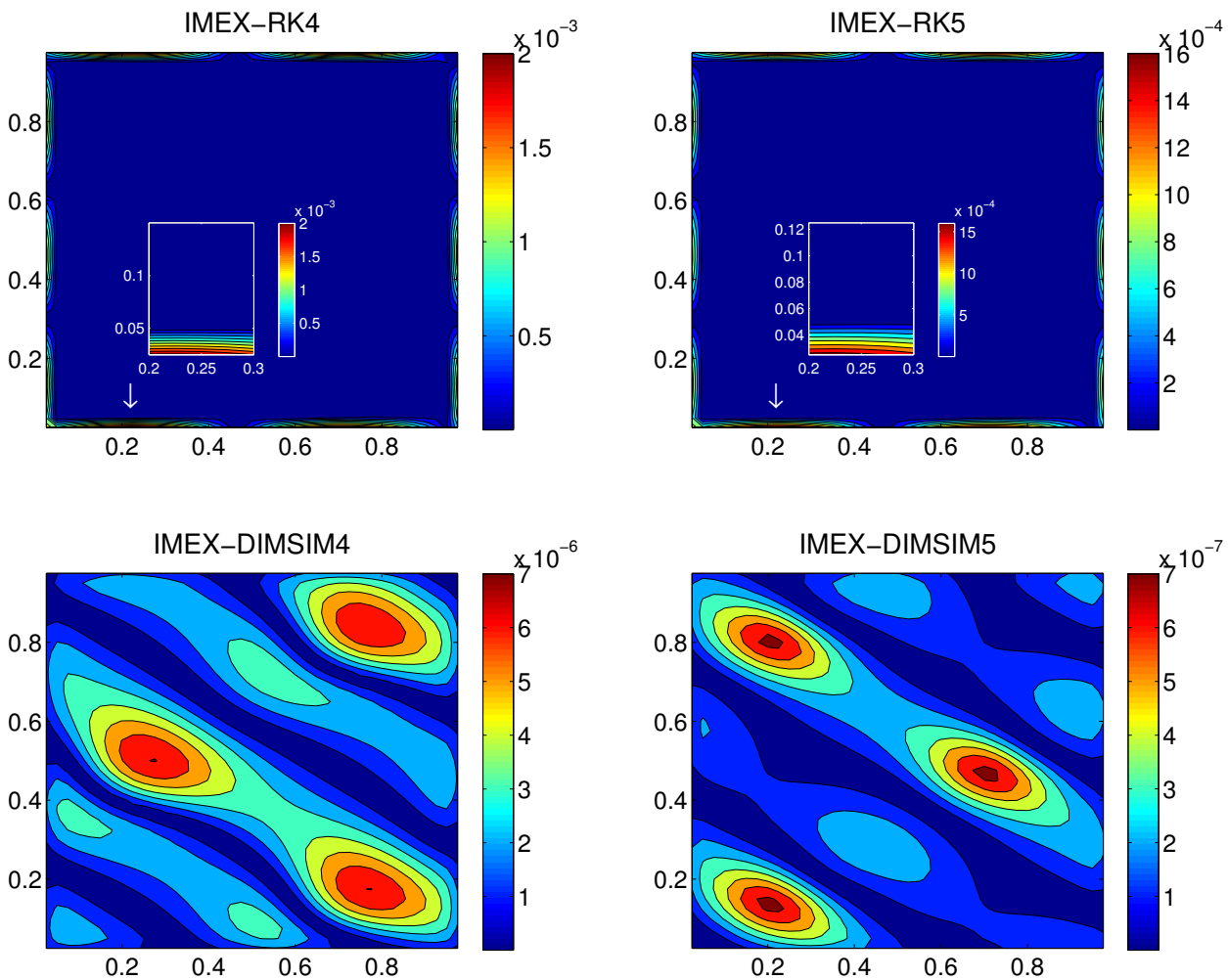


Figure 4.4: Absolute temporal errors at the final time $t = 0.5$ for various IMEX schemes on the 2D Allen-Cahn equation (4.20). A fixed time step of size $h = 1/50$ is used. IMEX-RK methods show large errors originating near boundaries. IMEX-DIMSIM methods have much smaller errors which are distributed over the entire domain.

4.4.2 Burgers' equation

The two-dimensional viscous Burgers equation [98]

$$\frac{\partial u}{\partial t} + \frac{1}{2}\nabla(u \cdot u) = \nu \nabla^2 u, \quad \nu = 0.1, \quad 0 \leq x, y \leq 1, \quad 0 \leq t \leq 1, \tag{4.22}$$

is a simplification of the 2D Navier-Stokes equations which admits the analytic solution

$$u^{\text{analytic}}(t, x, y) = \left(1 + e^{\frac{x+y-t}{2\nu}}\right)^{-1}.$$

The initial conditions and the Dirichlet boundary values correspond to the analytic solution. Spatial derivatives are discretized with second order central finite differences on a uniform grid with resolution $\Delta x = \Delta y = 1/50$.

The application of the IMEX integration treats the diffusion term implicitly and the convective term explicitly. We compare the numerical solutions against a reference solution computed with MATLAB routine $ode15s$ with tolerances $AbsTol = RelTol = 3 \times 10^{-14}$ that uses the same spatial discretization. Therefore the errors (4.21) reported here are only due to the temporal discretization.

Figure 4.5 compares the performance of the high order IMEX schemes. The convergence diagram in Figure 4.5(a) reveals that two IMEX-RK methods show order reduction to order two. The two IMEX-DIMSIMs converge with their theoretical orders. The efficiency diagram in Figure 4.5(b) illustrates again a gap in performance between the two families, with IMEX-DIMSIMs demonstrating a considerably better efficiency than IMEX-RK methods.

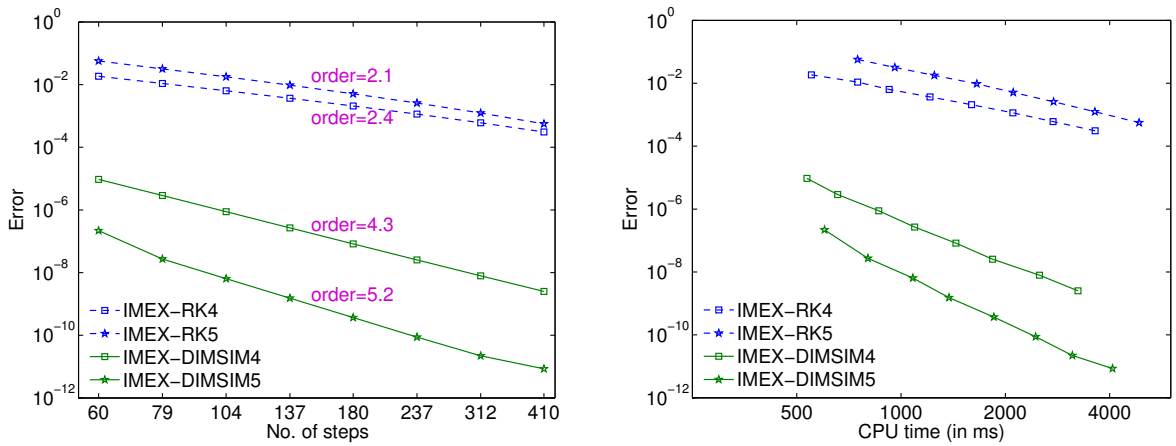
Figure 4.6 shows the spatial distribution of absolute errors at the final time. The boundary errors dominate the accuracy of the results for all schemes. The boundary conditions for this PDE may be more challenging than the previous one since they affects both spatial derivative terms in (4.22). Nevertheless, the error magnitude is much smaller for the IMEX-DIMSIM solutions.

4.4.3 Application to atmospheric simulations

Compressible Euler equations

The dynamics of non-hydrostatic atmospheric processes can be described by the compressible Euler equations [90]:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u} + p\mathbf{I}) &= -\rho g \widehat{\mathbf{e}}_{\mathbf{z}} \end{aligned} \tag{4.23a}$$



(a) Convergence diagram (b) Work-precision diagram

Figure 4.5: Comparison of high order IMEX-DIMSIM and IMEX-RK results for the 2D viscous Burgers equation (4.22). The integration time interval is $[0, 1]$. Shown are the temporal discretization errors corresponding to the solution at the final time $t = 1$.

$$\frac{\partial \rho\theta}{\partial t} + \nabla \cdot (\rho\theta\mathbf{u}) = 0\,,$$

where ρ is the density, $\mathbf{u} = (u, v, w)^T$ is the velocity vector, w being used in three-dimensional case, θ is the potential temperature, and $\mathbf{I}$ is the identity matrix. The gravitational acceleration is denoted by g while $\widehat{\mathbf{e}}_\mathbf{z}$ is a unit vector pointing upwards. The prognostic variables are ρ, $\rho\mathbf{u}$, and $\rho\theta$. The pressure p in the momentum equation is computed by the equation of state

$$p = p_0 \left(\frac{\rho\theta R_d}{p_0} \right)^{\frac{c_p}{c_v}}, \tag{4.23b}$$

where $p_0 = 10^5$ Pa is the surface pressure, R_d is the ideal gas constant, and c_p and c_v are the specific heat of the air for constant pressure and volume. To better maintain the hydrostatic state we follow the splitting introduced by Giraldo and Restelli [90]

$$\begin{aligned} \rho(\mathbf{x}, t) &= \bar{\rho}(z) + \rho'(\mathbf{x}, t) \\ (\rho\theta)(\mathbf{x}, t) &= \overline{(\rho\theta)}(z) + (\rho\theta)'(\mathbf{x}, t) \\ p(\mathbf{x}, t) &= \bar{p}(z) + p'(\mathbf{x}, t), \end{aligned}$$

where the overlined values are in hydrostatic balance. The governing equation (4.23) can then be rewritten as

$$\begin{aligned} \frac{\partial \rho'}{\partial t} &= -\nabla \cdot (\rho\mathbf{u}) \\ \frac{\partial \rho\mathbf{u}}{\partial t} &= -\nabla \cdot (\rho\mathbf{u}\mathbf{u} + p'\mathbf{I}) - \rho' g \widehat{\mathbf{e}}_\mathbf{z} \end{aligned} \tag{4.24a}$$

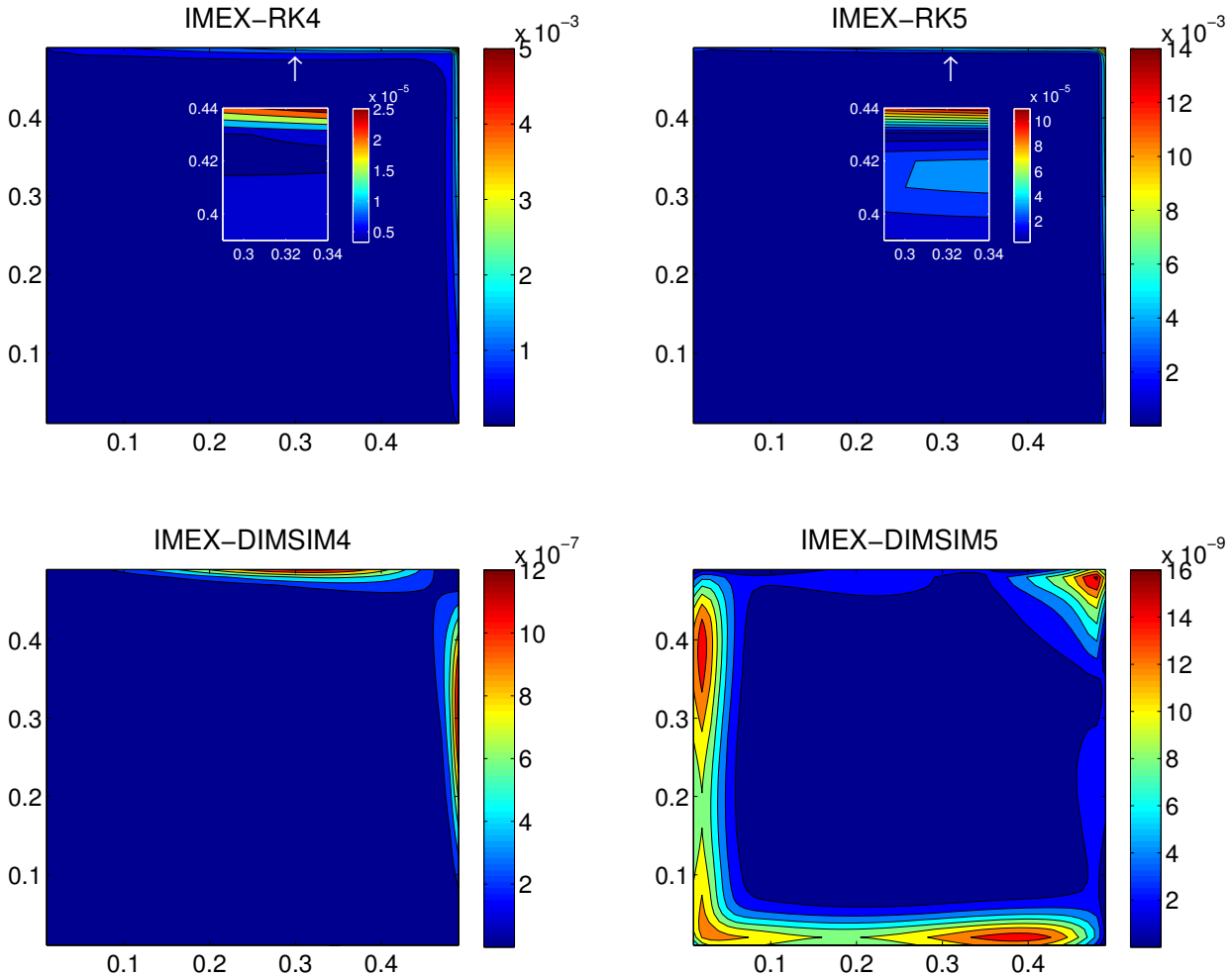


Figure 4.6: Absolute temporal errors at the final time $t = 1$ for various IMEX schemes on the 2D viscous Burgers equation (4.22). A fixed time step of size $h = 1/50$ is used. All methods show larger errors originating near boundaries. IMEX-DIMSIM methods have much smaller errors overall.

$$\frac{\partial(\rho\theta)'}{\partial t} = -\nabla\cdot(\rho\theta\mathbf{u}) \ ,$$

and closed with

$$p' = p_0\left(\frac{\rho\theta R_d}{p_0}\right)^{\frac{c_p}{c_v}} - \bar{p}. \tag{4.24b}$$

The equations are discretized in space using the discontinuous Galerkin method, whose usage for geophysical simulations is gaining popularity, e.g. [99, 45, 100, 101, 90]. The model, based upon the mesh database of the GMSH mesh generator code [46], has been used to solve several PDEs, either in the domain of geophysics [102, 103] and engineering [104, 105]. For more information about the space discretization, refer to [96].

The set of equations (4.24) applied to atmospheric flows is a good candidate for an IMEX time discretization, because of the different temporal scales involved. In usual atmospheric configurations, the acoustic waves are the fastest phenomena, with a propagation speed of about 340 ms^{-1}. This high celerity restricts the explicit time step to a small value due to the CFL stability condition. However, acoustic waves are generally not important for the modeler who is more interested by advective timescales. The IMEX method allows to circumvent the CFL condition by treating the linear acoustic waves implicitly, while the remaining terms are explicit. According to Giraldo et al. [9], the right-hand side of (4.24a) is additively split into a linear part responsible for the acoustic waves and a nonlinear part. The linear term

$$-\begin{bmatrix} \nabla\cdot(\rho\mathbf{u}) \\ \nabla\cdot(p'\mathbf{I}) + \rho' g\widehat{\mathbf{e}}_{\mathbf{z}} \\ \nabla\cdot\left(\rho\bar{\theta}\mathbf{u}\right) \end{bmatrix} \tag{4.25}$$

with the pressure linearized as

$$p' = \frac{c_p\bar{p}}{c_v\overline{\rho\theta}}\,(\rho\theta)'$$

is treated implicitly, while the remaining (nonlinear) terms are treated explicitly.

Test cases

In this chapter we consider two-dimensional and three-dimensional rising thermal bubble test cases slightly modified from the ones introduced in [90].

Two-dimensional case The motion of the air is driven by a time varying potential temperature perturbation from the bottom boundary

$$\theta' = \begin{cases} 0 & \text{for} \quad r > r_c, \\ \frac{\theta_c}{2}\left(1+\cos\left(\frac{\pi r}{r_c}\right)\right)\sin^2\left(\frac{\pi t}{50}\right) & \text{for} \quad r \le r_c, \end{cases} \tag{4.26}$$

where $\theta_c = 5°C$, $r = \sqrt{(x - x_c)^2}$, $r_c = 250$ m, and $(x, z) \in [0, 1000]^2$ with $t \in [0, 200]$ s and $x_c = 500$ m. No-flux boundaries are used for the other three boundaries. The computational domain is a 2D uniform mesh with actual resolution of about 66.7×66.7 m. Fourth-order polynomials are used on each element. The resulting ODE system contains $\sim 2.3 \times 10^4$ variables.

Three-dimensional case Diffusion terms

$$\begin{bmatrix} \nabla \cdot (\mu \nabla \rho') \\ \nabla \cdot (\mu \nabla (\rho \mathbf{u})) \\ \nabla \cdot (\mu \nabla (\rho \theta)') \end{bmatrix} \tag{4.27}$$

with $\mu = 6\ \mathrm{m^2 s^{-1}}$ are added to the right-hand side of (4.24a) to limit the oscillations resulting from a high order spatial discretization of a complex flow on a coarse grid.

The bottom boundary is also imposed as (4.26) with $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$, $r_c = 250$m, $(x, y, z) \in [200, 800]^2 \times [0, 600]$ and $(x_c, y_c) = (500, 500)$. No-flux boundaries are used for all the other boundaries. Considering the more expensive computational cost of the 3D test, we use a polynomial order of 3 for the DG scheme. A 3D uniform mesh grid with actual resolution of $100 \times 100 \times 100$ m is used. The resulting ODE system has $\sim 7 \times 10^4$ degrees of freedom.

Figure 4.7 shows the reference solutions at the final time for 2D and 3D cases.

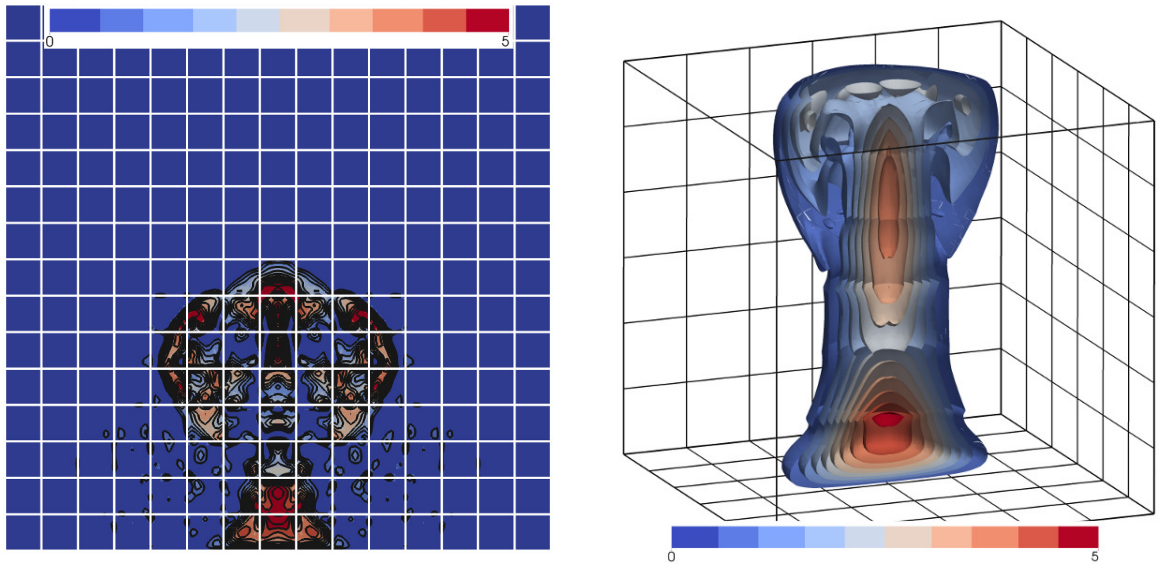
Numerical results

The relative L_2 errors for each of the prognostic variables

$$\mathbf{E}(q) = \sqrt{\frac{\int_\Omega (q^{\text{numerical}} - q^{\text{reference}})^2 d\Omega}{\int_\Omega (q^{\text{reference}})^2 d\Omega}}, \tag{4.28}$$

are measured against a reference solution obtained by applying the classic fourth-order explicit RK method to solve the original (non-split) model with a very small time step $h = 0.005s$. Since the time varying boundary conditions are imposed directly on the temperature term p' in the momentum equations of (4.24), we discuss the results for the variables $\rho\mathbf{u}$.

Figure 4.8 compares the convergence results and efficiency for the fourth-order IMEX-DIMSIM and IMEX-RK methods for the 2D simulations. As expected, the IMEX-DIMSIM reproduces the theoretical order of accuracy. But the IMEX-RK scheme shows an obvious order reduction, which translates into a loss of computational efficiency. The 3D results are given in Figure 4.9. The IMEX-RK method stills yields order reduction, less severely though. The error behavior of the IMEX-DIMSIM is somewhat irregular. It shows high



(a) 2D solution at final time $t = 200s$.

(b) 3D solution at final time $t = 300s$.

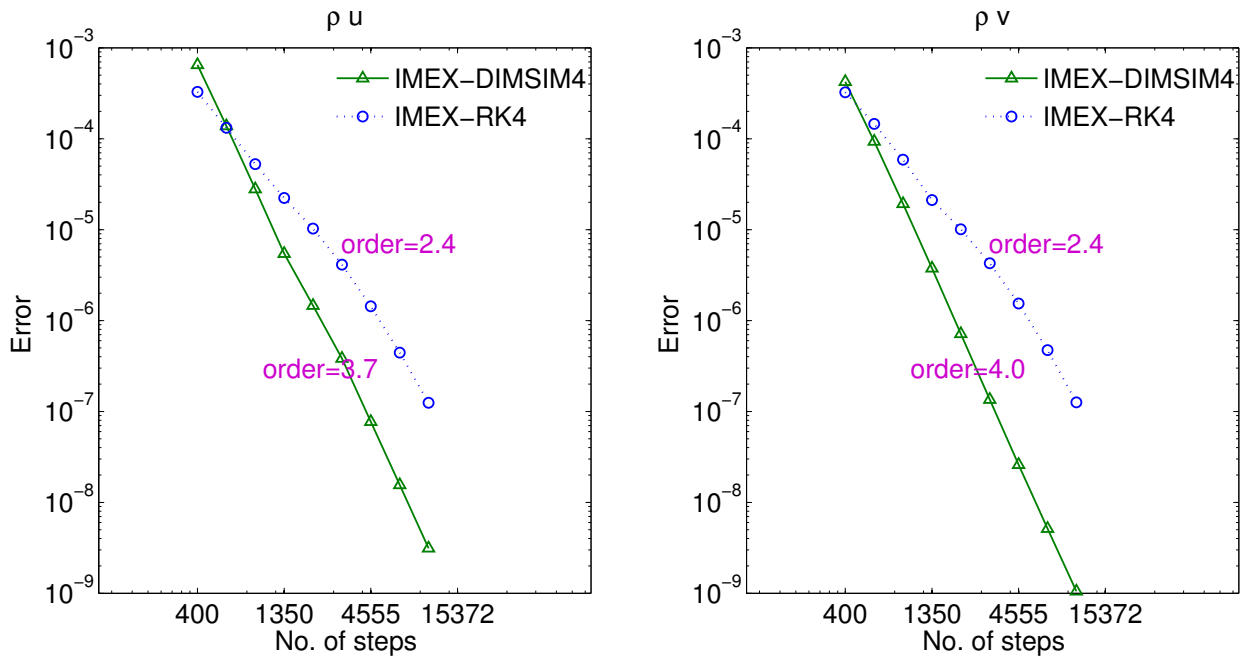
Figure 4.7: Perturbation of potential temperature (in $°C$) from the simulation of thermal rising bubble (4.24). The background mesh is displayed in wireframe.

order in the beginning, and then plateaus at the accuracy level 10^{-7} for a wide range of decreasing step sizes. The error plateau is likely due to the level of accuracy of the reference solution. However, even with this irregular behavior, the IMEX-DIMSIM is considerably more efficient than the IMEX-RK method. We have also tested large step sizes and found that the maximal allowable step size for IMEX-RK4 and IMEX-DIMSIM4 are both approximately equal to 1.0 sec. This agrees with the prediction of the stability analysis in section 4.3.1 which shows that the IMEX-DIMSIM has a good stability property. Furthermore, we notice that neither IMEX-RK5 nor IMEX-DIMSIM5 is suitable for this test problem because the maximal step sizes for them are restricted to values that are too small to make them competitive. Figure 4.10 shows that there are many eigenvalues of the Jacobian close to the imaginary axis, therefore a stability region covering a large part of imaginary axis is highly desirable.

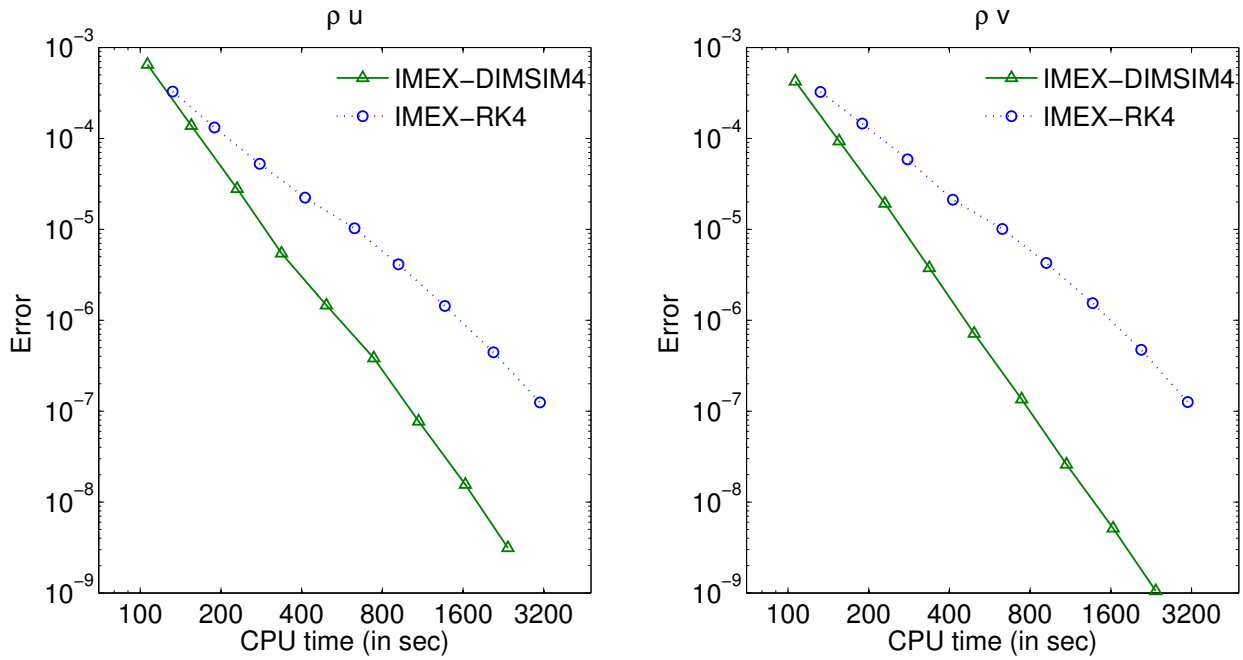
4.5 Conclusions

Multiscale problems in science and engineering are modeled by time-dependent systems of equations involving both stiff and nonstiff terms. Implicit-explicit time stepping schemes perform an implicit integration only for the stiff components of the system, and thus combine the low cost of explicit methods with the favorable stability properties of implicit methods.

Many modern PDE solvers use high order spatial discretization schemes, e.g., the discon-

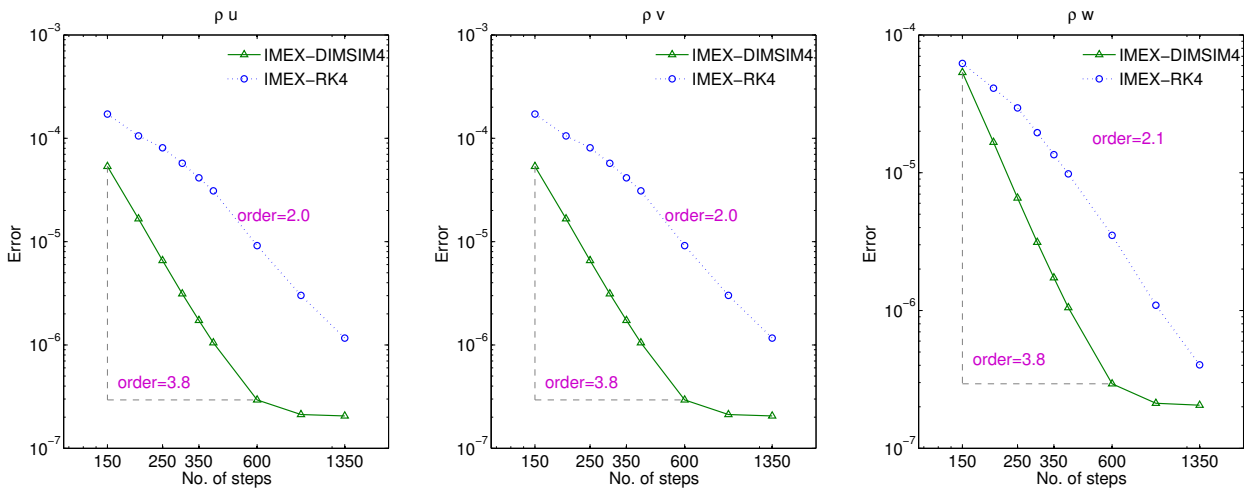


(a) Convergence diagram

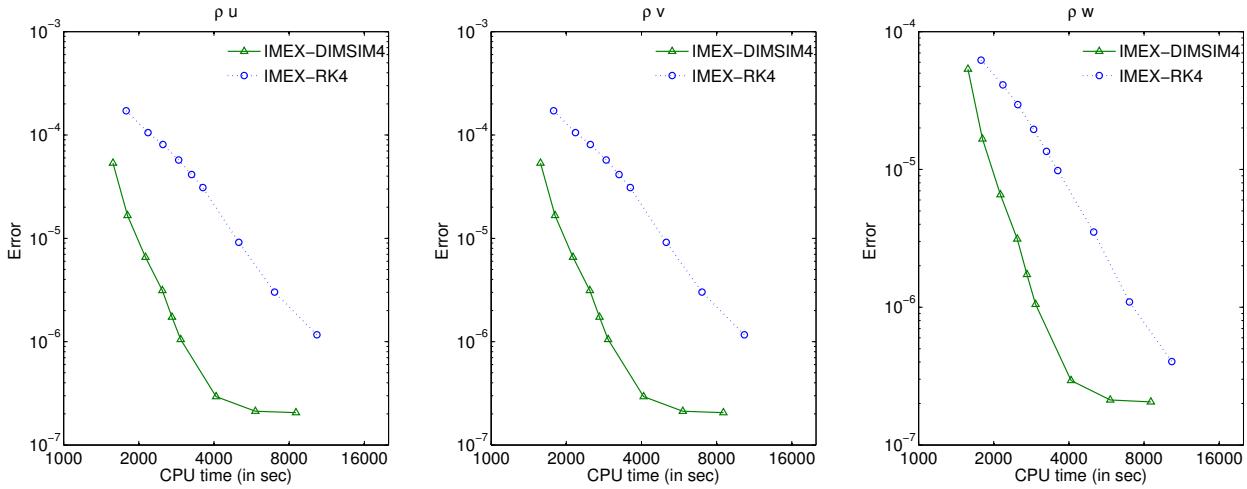


(b) Work-precision diagram

Figure 4.8: Comparison of high order IMEX-DIMSIM and IMEX-RK results for the 2D rising bubble simulation (4.24). The integration time interval is $[0, 200]$ sec. and is divided into 400,600,900,1350,2025,3037,4555,6832,10248 equal time steps to obtain the points in the diagrams. Temporal errors for all the variables (4.28) are computed for the solution at the final time.



(a) Convergence diagram



(b) Work-precision diagram

Figure 4.9: Comparison of high order IMEX-DIMSIM and IMEX-RK results for the 3D rising bubble (4.24). The integration time interval is $[0, 300]$ sec. and is divided into $150, 200, 250, 300, 350, 400, 600, 900, 1350$ equal time steps to obtain the points in the diagrams. Temporal errors for all the variables (4.28) are computed for the solution at the final time.

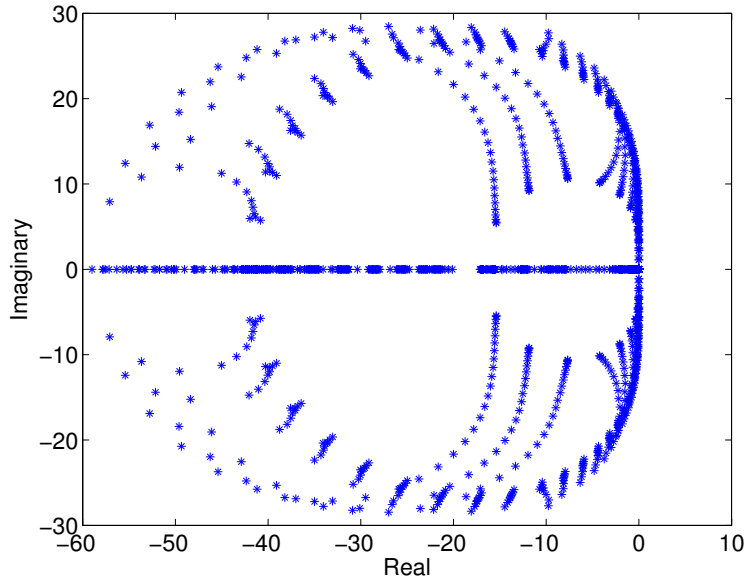


Figure 4.10: Plot of eigenvalues of the Jacobian for 2D rising bubble test problem.

tinuous Galerkin approach with high degree polynomials. Often the high order of spatial discretization is paired with a low order traditional time stepping scheme. It is therefore of considerable importance to develop high order time stepping algorithms that match the accuracy of the spatial discretization.

This chapter addresses the need for high order implicit-explicit temporal discretizations in large scale applications. We construct new fourth and fifth order IMEX DIMSIM schemes based on L-stable implicit components, and with the explicit components optimized such as to maximize the constrained stability regions. The new methods have good stability properties and can take large step sizes for stiff problems.

Several test problems from different application areas that can benefit from implicit-explicit integration are considered. These problems are the two-dimensional Allen-Cahn and Burgers' equations with finite difference spatial discretizations, and two- and three-dimensional compressible Euler equations with discontinuous Galerkin space discretizations. The performance of the new fourth and fifth order IMEX-DIMSIMs is compared against existing fourth and fifth order IMEX-RK methods. In all cases the IMEX-DIMSIMs can use large step sizes - similar to those taken by traditional implicit-explicit Runge-Kutta methods. However, the high stage order enables our methods to avoid the order reduction that plagues classic IMEX-RK methods when applied to stiff systems or to problems with complex boundary conditions. In all cases IMEX-DIMSIMs are considerably more efficient than traditional IMEX-RK methods of the same order.

Typically multiscale flow simulations are carried out using fixed, predefined time steps. This is the approach taken in this chapter as well.

The high order IMEX-GLM schemes proposed herein are not only of interest to multiscale nonhydrostatic atmospheric simulations, but also to many other fields where large-scale multiscale simulations are carried out with high order spatial discretizations. IMEX-GLMs can prove especially useful in situations where IMEX-RK methods suffer from order reduction; specific examples include stiff systems of singular perturbation type or problems with challenging time-dependent boundary conditions.

Table 4.1: Coefficients of the IMEX-DIMSIM-4.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.258897065974412 & 0 & 0 & 0 \\ 2.729801825357062 & -0.060004247312668 & 0 & 0 \\ 0.951308318232761 & 0.614160494289040 & 0.422498793609078 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 5.669708110906782 & -0.493235358869745 & 0.021475944586626 & 0.175951726795284 \\ 5.544708110906782 & 0.020653530019144 & -0.797968499857818 & 0.680943549709761 \\ 4.720814974705226 & 3.191226074825372 & -5.227438428178271 & 0.686166890688894 \\ 4.848863779632135 & 2.337640759837926 & -3.218585217497575 & 0.418013495315584 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0.074436267358921 & 0.055555555555556 & 0.006172839506173 & 0.000514403292181 \\ 1 & -2.003130911377728 & 0.242223637993112 & 0.052716285344531 & 0.008600849263247 \\ 1 & -0.987967606130879 & 0.013613972830935 & 0.038658018404147 & 0.017011414548385 \end{bmatrix}$$

$$\widehat{A} = \begin{bmatrix} 0.572816062482135 & 0 & 0 & 0 \\ 0.294478591621391 & 0.572816062482135 & 0 & 0 \\ 3.754531024312379 & -0.446626145372372 & 0.572816062482135 & 0 \\ 20.906355951077522 & -6.918033573971423 & 0.824272703722306 & 0.572816062482135 \end{bmatrix}$$

$$\widehat{B} = \begin{bmatrix} 2.818382755109841 & -0.107847984112942 & 1.213319973963157 & -0.548700992864529 \\ 3.266198817591976 & -1.885223345152593 & 3.830771904411522 & -1.797738883043436 \\ 3.774131970777119 & -3.469139895411032 & 5.100995462482731 & -4.672071998026633 \\ 1.800600620848989 & 6.203817506581311 & -13.407704583723200 & -5.034154872439978 \end{bmatrix}$$

$$\widehat{Q} = \begin{bmatrix} 1 & -0.572816062482135 & 0 & 0 & 0 \\ 1 & -0.533961320770192 & -0.135383131938489 & -0.025650275076168 & -0.003021498328079 \\ 1 & -3.214054274755475 & -0.010779770975077 & -0.053097178648182 & -0.017299808772539 \\ 1 & -14.385411143310540 & 1.683679993026802 & 0.081422122041277 & -0.051803591005091 \end{bmatrix}$$

$$v = [\ 0.281364340879037 \quad -1.282889560784121 \quad 2.266595749735792 \quad -0.265070529830707\ ]$$

$$c = [\ 0 \quad 1/3 \quad 2/3 \quad 1\ ]$$

Table 4.2: Coefficients of the IMEX-DIMSIM-5.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.380631951399918 & 0 & 0 & 0 & 0 \\ -0.723344119927179 & 0.934338548518619 & 0 & 0 & 0 \\ -0.292421654731536 & 1.489386717103117 & 0.229042913082062 & 0 & 0 \\ 10.333193352608074 & 0.200217292186561 & 0.841800685401247 & -0.148918889975160 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} -1.811278483713069 & 2.072219536433343 & 0.130011155311711 & 0.166279568600910 & 0.117403740739418 \\ -1.724125705935292 & 1.629858425322231 & 1.038344488645044 & -0.796914875843534 & 0.396841233783945 \\ -1.998394810009466 & 3.088356723470882 & -2.146707663207811 & 2.854109498231544 & -0.833722659704275 \\ -1.361504766226497 & 0.334933035918415 & 2.154212895587752 & 0.353113262914561 & -1.482126886275562 \\ 5.091061924499312 & -29.458910962376240 & 55.143920860593482 & -43.440447985319850 & 3.112719239754878 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -0.130631951399918 & 0.031250000000000 & 0.002604166666667 & 0.000162760416667 & 0.000008138020833 \\ 1 & 0.289005571408560 & -0.108584637129655 & -0.008364746307874 & 0.000170993363233 & 0.000108343335202 \\ 1 & -0.676007975453643 & -0.205618135816810 & -0.004861199044730 & 0.004533255151668 & 0.001138659940362 \\ 1 & -10.226292440220721 & 0.140734501734106 & 0.097068228416195 & 0.034078612640450 & 0.008071842745668 \end{bmatrix}$$

$$\widehat{A} = \begin{bmatrix} 0.278053841136452 & 0 & 0 & 0 & 0 \\ 0.220452276182580 & 0.278053841136452 & 0 & 0 & 0 \\ 2.294819895736366 & -0.602366708071285 & 0.278053841136452 & 0 & 0 \\ 5.054620901153854 & -1.529876218309763 & 0.097119141498823 & 0.278053841136452 & 0 \\ 9.345167780108133 & -1.412133513099773 & -1.883401998517870 & 0.782533955446870 & 0.278053841136452 \end{bmatrix}$$

$$\widehat{B} = \begin{bmatrix} 6.044855283302179 & -2.020000467205476 & 0.032934533641225 & 0.593578985923315 & -0.226664851205853 \\ 5.853954219943505 & -1.072092372634326 & -1.839270544389963 & 2.410922952843391 & -0.899263047489796 \\ 6.004175007913425 & -2.014097375842605 & 0.610845429880394 & -0.963490004887004 & -0.405182760273902 \\ 6.002703177071046 & -2.556003283230891 & 3.151551366098853 & -5.493514217893924 & 0.448102618067392 \\ 4.481882795290198 & 2.672564354868939 & -1.413660973235832 & -8.058154793746990 & 0.909905877341711 \end{bmatrix}$$

$$\widehat{Q} = \begin{bmatrix} 1 & -0.278053841136452 & 0 & 0 & 0 & 0 \\ 1 & -0.248506117319032 & -0.038263460284113 & -0.006085015868847 & -0.000561338127960 & -0.000037118138206 \\ 1 & -1.470507028801533 & 0.136564756449595 & 0.004900562818504 & -0.001619958388074 & -0.000365640421568 \\ 1 & -3.149917665479366 & 0.406619102975690 & 0.027778596315200 & -0.004406329750951 & -0.001692120959916 \\ 1 & -6.110220065073812 & 0.929780069812273 & 0.087106493228110 & -0.016782586272280 & -0.008434321001423 \end{bmatrix}$$

$$v = [\ -0.079385465132435 \quad 0.554317572910577 \quad -1.569589549144155 \quad 2.332074592443682 \quad -0.237417151077669\ ]$$

$$c = [\ 0 \quad 1/4 \quad 1/2 \quad 3/4 \quad 1\ ]$$

Chapter 5

IMEX two-step Runge-Kutta methods

5.1 Introduction

In this chapter we are still concerned with solving systems that involve both stiff and non-stiff processes. They are modeled by the following system of ordinary differential equations (ODEs)

$$y'(t) = F(y) = f(y) + g(y)\,, \quad t_0 \leq t \leq t_F\,, \quad y(t_0) = y_0\,; \quad y(t) \in \mathbb{R}^N\,, \tag{5.1}$$

where f and g represent the non-stiff and the stiff components, respectively. Without loss of generality, we skip the explicit dependence of f and g on the time argument (Compare 3.1). Problems of the form (5.1) often arise from the spatial discretization of partial differential equations (PDEs) in the method of lines approach; in this case y is the semi-discrete state. See Section 3.1 in Chapter 3 for more details.

General linear (GL) methods [106, 107, 108, 109, 110] represent a natural generalization of both RK and LM methods, as they use both internal stages and information from previous solution steps. Two step Runge Kutta (TSRK) methods are a subclass of general linear methods that uses the stage values from only one previous step [111, 112, 113, 114]. The added flexibility of using using both internal and external stage information allows to design algorithms with superior stability and accuracy properties.

This work proposes a new family of implicit-explicit methods based on pairs of TSRK schemes. The order conditions and the stability properties of the resulting discretization are investigated. Two practical IMEX TSRK methods are constructed, and are used in numerical tests to illustrate the theoretical findings.

The remainder of this chapter is organized as follows. Section 5.2 introduces the partitioned two-step Runge Kutta methods, and order conditions are derived in Section 5.2.3. Implicit-explicit TSRK methods are proposed in Section 5.3, and their stability properties

are discussed in Section 5.4. Practical IMEX TRSK methods are constructed in Section 5.6, and are used in numerical tests in Section 5.7. Conclusions are drawn in Section 5.8.

5.2 Partitioned two-step Runge Kutta methods

5.2.1 Two-step Runge Kutta (TSRK) methods

A TSRK method advances the numerical solution of (5.1) from t_n to $t_{n+1} = t_n + h$ as follows [24]:

$$\begin{aligned} Y_i^{[n]} &= (1-u_i)\, y_{n-1} + u_i\, y_{n-2} \\ &\quad + h \sum_{j=1}^{s} \left(a_{i,j}\, F\left(Y_j^{[n]}\right) + b_{i,j}\, F\left(Y_j^{[n-1]}\right)\right), \quad i = 1, \ldots, s\,, \end{aligned} \tag{5.2a}$$

$$\begin{aligned} y_n &= (1-\vartheta)\, y_{n-1} + \vartheta\, y_{n-2} \\ &\quad + h \sum_{j=1}^{s} \left(v_j\, F\left(Y_j^{[n]}\right) + w_j\, F\left(Y_j^{[n-1]}\right)\right). \end{aligned} \tag{5.2b}$$

The method (5.2) can be represented compactly by its tableau of coefficients [24]

$$\begin{array}{c|c|c} \mathbf{u} & \mathbf{A} & \mathbf{B} \\ \hline \vartheta & \mathbf{v}^T & \mathbf{w}^T \end{array}\,, \tag{5.3}$$

where $\mathbf{A} = (a_{i,j})_{1\le i,j\le s}$, $\mathbf{B} = (b_{i,j})_{1\le i,j\le s}$, $\mathbf{u} = (u_i)_{1\le i\le s}$, $\mathbf{v} = (v_i)_{1\le i\le s}$, $\mathbf{w} = (w_i)_{1\le i\le s}$, and ϑ is a scalar. In addition the abscissa vector $\mathbf{c} = (c_i)_{1\le i\le s}$ describes the time points where stage approximations are computed.

The TSRK method (5.2) has *stage order* q [112] if the stage vectors $Y_j^{[n]}$ are order q approximations of the exact solution at $t_{n-1} + c_j h$

$$Y_j^{[n]} = y(t_{n-1} + c_j h) + \mathcal{O}(h^{q+1})\,, \quad h \to 0\,, \quad j = 1, \ldots, s\,.$$

According to [112, 24], the necessary and sufficient condition for (5.2) to have stage order q is:

$$\frac{\mathbf{c}^\nu}{\nu!} - \frac{(-1)^\nu}{\nu!}\mathbf{u} - \frac{\mathbf{A}\mathbf{c}^{\nu-1}}{(\nu-1)!} - \frac{\mathbf{B}(\mathbf{c}-\mathbf{e})^{\nu-1}}{(\nu-1)!} = 0\,, \quad \nu = 1, \ldots, q\,, \tag{5.4}$$

where $\mathbf{e} = [1, \ldots, 1]^T \in \mathbb{R}^s$ and the power operator is applied component-wise. Note that the stage consistency condition ($\nu = 1$) determines the abscissa vector,

$$\mathbf{c} = (\mathbf{A} + \mathbf{B})\,\mathbf{e} - \mathbf{u}\,. \tag{5.5}$$

The TSRK method (5.2) has *order* p [24, 112] if the final approximation y_{n_F} to the solution $y(t_F)$ is of (global) order p, that is

$$y_{n_F} = y(t_F) + \mathcal{O}(h^p)\,, \quad h \to 0\,.$$

Consider a TSRK method (5.2) with stage order $q \geq p-1$. If the coefficients satisfy the additional conditions

$$\frac{1}{\nu!} - \frac{(-1)^\nu}{\nu!}\vartheta - \frac{\mathbf{v}^T\mathbf{c}^{\nu-1}}{(\nu-1)!} - \frac{\mathbf{w}^T(\mathbf{c}-\mathbf{e})^{\nu-1}}{(\nu-1)!} = 0\,, \quad \nu = 1,\ldots,p\,, \tag{5.6}$$

then the method has order p [24, 112].

5.2.2 Partitioned TSRK methods

Consider the partitioned system of ODEs

$$y' = \begin{bmatrix} y_{\{1\}} \\ \vdots \\ y_{\{N\}} \end{bmatrix}' = \begin{bmatrix} f_{\{1\}}(y) \\ \vdots \\ f_{\{N\}}(y) \end{bmatrix} = \sum_{k=1}^{N} \begin{bmatrix} \mathbf{0} \\ f_{\{k\}}(y) \\ \mathbf{0} \end{bmatrix} = f(y) \tag{5.7}$$

with scalar quantities $y_{\{1\}},\ldots,y_{\{N\}}$.

Each of the N ODEs for $y'_{\{k\}}$ is integrated using a different TSRK method

$$\begin{array}{c|c|c} \mathbf{u}^{\{k\}} & \mathbf{A}^{\{k\}} & \mathbf{B}^{\{k\}} \\ \hline \vartheta^{\{k\}} & \left(\mathbf{v}^{\{k\}}\right)^T & \left(\mathbf{w}^{\{k\}}\right)^T \end{array}\,, \quad k = 1,\ldots,N\,. \tag{5.8}$$

The coefficients of the methods (5.8) have to be coordinated such that the overall discretization of (5.7) has the desired accuracy and stability. We focus on families of methods (5.8) that have several particular characteristics which make them suitable for the integration of PDEs, discretized in the method of lines.

Remark 3. *The splitting of* (5.7), (5.8) *into scalar components does not restrict the generality of the discussion. All results in this section extend directly to systems of equations by selecting the component methods in* (5.8) *such that all components of a subsystem are discretized with the same scheme. For example, if the system* (5.7) *is partitioned into two vector subsystems with* $y_{[1]} = [y_{\{1\}},\ldots,y_{\{M\}}]^T$ *and* $y_{[2]} = [y_{\{M+1\}},\ldots,y_{\{N\}}]^T$*, then we apply a two way partitioned method by using* (5.8) *with* $\mathbf{A}^{\{1\}} = \cdots = \mathbf{A}^{\{M\}} = \mathbf{A}^{[1]}$, $\mathbf{A}^{\{M+1\}} = \cdots = \mathbf{A}^{\{N\}} = \mathbf{A}^{[2]}$, *and similarly for* $\mathbf{B}$, ϑ, $\mathbf{u}$, $\mathbf{v}$, *and* $\mathbf{w}$.

Preservation of linear invariants Preservation of linear invariants is an essential property for the integration of conservation laws [11, 115, 13]. For example, the numerical solution of the advection equation conserves the total mass of the system (to roundoff accuracy) if the space discretization is flux conservative, and the time discretization preserves linear invariants.

The family of TSRK methods (5.8) conserves linear invariants of the system if all the weights are equal to each other. Therefore of particular interest are methods which share the same theta

$$\vartheta^{\{k\}} = \vartheta\,, \quad k = 1, \ldots, N\,, \tag{5.9}$$

and the same weight vectors

$$\mathbf{v}^{\{k\}} = \mathbf{v}\,, \quad \mathbf{w}^{\{k\}} = \mathbf{w}\,. \quad k = 1, \ldots, N\,. \tag{5.10}$$

To be specific, consider a linear invariant of the ODE system (5.7)

$$\sum_{k=1}^{N} \mu_{\{k\}}\, f_{\{k\}}(y) = 0\,, \quad \forall\, y \quad \Rightarrow \quad \sum_{k=1}^{N} \mu_{\{k\}}\, y_{\{k\}}(t) = C = \text{constant}\,, \quad \forall\, t\,.$$

Assume that all the previous numerical solutions preserve this invariant

$$\sum_{k=1}^{N} \mu_{\{k\}}\, y_{\{k\},\ell} = C\,, \quad \ell = 0, \ldots, n-1\,.$$

From (5.2b), (5.9), and (5.10) it follows that the next step solution y_n also preserves the invariant

$$\begin{aligned}
\sum_{k=1}^{N} \mu_{\{k\}}\, y_{\{k\},n} \;=\; & (1-\vartheta) \sum_{k=1}^{N} \mu_{\{k\}}\, y_{\{k\},n-1} + \vartheta \sum_{k=1}^{N} \mu_{\{k\}}\, y_{\{k\},n-2} \\
& + h \sum_{j=1}^{s} v_j \sum_{k=1}^{N} \mu_{\{k\}}\, f_{\{k\}} \left(Y^{[n]}_{\{1\},j}, \ldots, Y^{[n]}_{\{N\},j} \right) \\
& + h \sum_{j=1}^{s} w_j \sum_{k=1}^{N} \mu_{\{k\}}\, f_{\{k\}} \left(Y^{[n-1]}_{\{1\},j}, \ldots, Y^{[n-1]}_{\{N\},j} \right) \quad = C\,.
\end{aligned}$$

Internal consistency We consider the case where that all individual methods (5.8) are stage consistent (5.5). Each method (5.8) computes stage values $Y^{[n]}_{\{k\}}$ that approximate the exact solution components $y_{\{k\}}(t_n + \mathbf{c}^{\{k\}}\, h)$ at abscissae $\mathbf{c}^{\{k\}} = (\mathbf{A}^{\{k\}} + \mathbf{B}^{\{k\}})\, \mathbf{e} - \mathbf{u}^{\{k\}}$. We call the method (5.8) internally consistent if all components $Y^{[n]}_{\{k\},i}$ of a stage vector approximate the exact solution at the same time moment, $t_n + c_i h$ for all k. Consequently, we focus on stage consistent methods which satisfy the simplifying assumption

$$\mathbf{u}^{\{k\}} \;=\; \mathbf{u}\,, \quad k = 1, \ldots, N\,, \tag{5.11}$$

and for which the abscissae of all N TSRK methods are equal

$$\mathbf{c}^{\{k\}} = \left(\mathbf{A}^{\{k\}} + \mathbf{B}^{\{k\}}\right) \cdot \mathbf{e} - \mathbf{u} \;=\; \mathbf{c}\,, \quad k = 1, \ldots, N\,. \tag{5.12}$$

The class of methods under consideration The class of partitioned TSRK (PTSRK) methods under consideration enjoys the properties of theta-consistency (5.9), u-consistency (5.11), linear conservation (5.10), and stage consistency (5.12). This class is characterized by the tableaux

$$\mathbf{c}^{\{k\}} = \mathbf{c}\,, \quad \begin{array}{c|c|c} \mathbf{u} & \mathbf{A}^{\{k\}} & \mathbf{B}^{\{k\}} \\ \hline \vartheta & \mathbf{v}^T & \mathbf{w}^T \end{array}\,, \quad k = 1, \ldots, N\,. \tag{5.13}$$

Partitioned methods in this family are written explicitly as follows:

$$\begin{aligned} Y^{[n]}_{\{k\},i} &= (1-u_i)\, y_{\{k\},n-1} + u_i\, y_{\{k\},n-2} && (5.14a)\\ &\quad + h \sum_{j=1}^{s} a^{\{k\}}_{i,j} f_{\{k\}}\left(Y^{[n]}_{\{1\},j}, \ldots, Y^{[n]}_{\{N\},j}\right) \\ &\quad + h \sum_{j=1}^{s} b^{\{k\}}_{i,j} f_{\{k\}}\left(Y^{[n-1]}_{\{1\},j}, \ldots, Y^{[n-1]}_{\{N\},j}\right) \\ y_{\{k\},n} &= (1-\vartheta)\, y_{\{k\},n-1} + \vartheta\, y_{\{k\},n-2} && (5.14b)\\ &\quad + h \sum_{j=1}^{s} v_j f_{\{k\}}\left(Y^{[n]}_{\{1\},j}, \ldots, Y^{[n]}_{\{N\},j}\right) \\ &\quad + h \sum_{j=1}^{s} w_j f_{\{k\}}\left(Y^{[n-1]}_{\{1\},j}, \ldots, Y^{[n-1]}_{\{N\},j}\right) \\ k &= 1, \ldots, N\,. \end{aligned}$$

5.2.3 Order conditions for partitioned TSRK methods

Theorem 5.2.1 (Convergence for the special class of PTSRK methods (5.13))**.** *Consider the class (5.13) of partitioned TSRK methods which satisfy the internal stage consistency condition (5.12). If*

- *each individual method is zero stable,*
- *the starting values are of order p,*
- *each component method has order p,*
- *each component method has stage order $q \geq p-1$,*

then the numerical solution converges with order p,

$$y_n - y(t_n) = \mathcal{O}(h^p)\,, \quad \forall\, n\,.$$

Remark 4. *In this case no "coupling" order conditions are needed. By "coupling" we mean order conditions in the form of nonlinear equations involving the coefficients of multiple methods. Such conditions are needed, for example, in the case of partitioned Runge Kutta methods [51], as well as in the case of general PTSRK methods.*

A proof based on P-trees is established by Sandu in [116]. Theorem 5.2.1 in Chapter 3.3.1 also implies this result since TSRK can be transformed into the general form of GLM (3.3) by combining y_{n-1}, y_{n-2} and $hF(Y^{n-1})$ (See (5.2)) into a column vector. A direct proof of Theorem 5.2.1 following the approach of Jackiewicz [24, 112] that is not based on P-trees is given in [42].

For PTSRK methods, this theorem leads to stage order conditions

$$c_i^\nu = u_i\,(-1)^\nu + \sum_{j=1}^{s} a_{i,j}^{\{k\}} \nu\, c_j^{\nu-1} + \sum_{j=1}^{s} b_{i,j}^{\{k\}} \nu\,(1-c_i)^{\nu-1}\,, \tag{5.15}$$
$$\text{for all } 1 \le \nu \le q \text{ and } k = 1,\ldots,N\,.$$

and order conditions

$$1 = \vartheta\,(-1)^\nu + \sum_{j=1}^{s} v_j^{\{k\}} \nu\, c_j^{\nu-1} + \sum_{j=1}^{s} w_j^{\{k\}} \nu\,(1-c_i)^{\nu-1}\,, \tag{5.16}$$
$$\text{for all } 1 \le \nu \le p \text{ and } k = 1,\ldots,N\,.$$

5.3 Implicit-explicit TSRK methods

Implicit-explicit (IMEX) TSRK methods are PTSRK methods (5.13) with a two-way splitting

$$\mathbf{A}^{\{1\}} = \mathbf{A}\,, \quad \mathbf{A}^{\{2\}} = \widehat{\mathbf{A}}\,, \quad \mathbf{B}^{\{1\}} = \mathbf{B}\,, \quad \mathbf{B}^{\{2\}} = \widehat{\mathbf{B}}\,,$$

where $\mathbf{A}$, $\mathbf{B}$ are the coefficients of an an explicit TSRK method, and $\widehat{\mathbf{A}}$, $\widehat{\mathbf{B}}$ are the coefficients of an implicit one. Moreover, $\widehat{\mathbf{A}}$ is lower triangular, with $\widehat{\mathbf{A}}_{i,i} = \gamma$ for all stages $i = 1,\ldots,s$. An IMEX TSRK method applied to (5.1) reads:

$$\begin{aligned} Y_i^{[n]} &= (1-u_i)\,y_{n-1} + u_i\,y_{n-2} \\ &\quad + h\sum_{j=1}^{i-1} a_{i,j}\, f\left(Y_j^{[n]}\right) + h\sum_{j=1}^{s} b_{i,j}\, f\left(Y_j^{[n-1]}\right) \end{aligned} \tag{5.17a}$$

$$
\begin{aligned}
& +h\sum_{j=1}^{i-1}\widehat{a}_{i,j}\,g\left(Y_j^{[n]}\right)+h\gamma\,g\left(Y_i^{[n]}\right)+h\sum_{j=1}^{s}\widehat{b}_{i,j}\,g\left(Y_j^{[n-1]}\right)\,, \\
y_n \;=\; & (1-\vartheta)\,y_{n-1}+\vartheta\,y_{n-2} \qquad\qquad (5.17b)\\
& +h\sum_{j=1}^{s}\left(v_j\,(f+g)\left(Y_j^{[n]}\right)+w_j\,(f+g)\left(Y_j^{[n-1]}\right)\right)\,.
\end{aligned}
$$

5.4 Stability aspects

Application of the TSRK method (5.3) to the linear scalar test equation

$$y' = \zeta\, y\,,$$

leads to a stability matrix of the form

$$\mathbf{M}(z)=\begin{bmatrix} 1-\vartheta+z\mathbf{v}^T\mathbf{S}(z)(\mathbf{e}-\mathbf{u}) & \vartheta+z\mathbf{v}^T\mathbf{S}(z)\mathbf{u} & \mathbf{w}^T+z\mathbf{v}^T\mathbf{S}(z)\mathbf{B} \\ 1 & 0 & \mathbf{0} \\ z\mathbf{S}(z)(\mathbf{e}-\mathbf{u}) & z\mathbf{S}(z)\mathbf{u} & z\mathbf{S}(z)\mathbf{B} \end{bmatrix}, \qquad (5.18)$$

with $z = h\,\zeta$ and $\mathbf{S}(z) = (\mathbf{I}_s - z\mathbf{A})^{-1}$ (see [24, p.95]). The method is linearly stable if the spectral radius of the stability matrix (5.18) is smaller than or equal to one. The stability region of the method is defined as

$$\mathcal{S} = \{z \in \mathbb{C} \;:\; \rho(M(z)) \le 1\}\,.$$

5.4.1 Linear stability of the partitioned method

To assess the stability of the IMEX method (5.17), i.e., to study how the stability properties of the two methods combine when used in tandem, we consider the linear scalar test problem

$$y' = \zeta\, y + \eta\, y\,, \quad \{\,\text{where } f(y) = \zeta\, y\,,\;\; g(y) = \eta\, y\,\}\,. \qquad (5.19)$$

We denote the non-stiff and the stiff complex variables by $z = h\,\zeta$ and $x = h\,\eta$, respectively. The method (5.17) applied to the scalar test equation (5.19) gives

$$
\begin{aligned}
Y^{[n]} &= (\mathbf{e}-\mathbf{u})\,y_{n-1}+\mathbf{u}\,y_{n-2}+\left(z\,\mathbf{A}+x\,\widehat{\mathbf{A}}\right)\,Y^{[n]}+\left(z\,\mathbf{B}+x\,\widehat{\mathbf{B}}\right)\,Y^{[n-1]} \\
y_n &= (1-\vartheta)\,y_{n-1}+\vartheta\,y_{n-2}+(z+x)\,\mathbf{v}^T\,Y^{[n]}+(z+x)\,\mathbf{w}^T\,Y^{[n-1]}\,.
\end{aligned}
$$

Solving this equation leads to

$$\begin{bmatrix} y_n \\ y_{n-1} \\ Y^{[n]} \end{bmatrix} = \mathbf{M}(x,z)\cdot\begin{bmatrix} y_{n-1} \\ y_{n-2} \\ Y^{[n-1]} \end{bmatrix}$$

with

$$\mathbf{M}(x,z) = \begin{bmatrix} \mathbf{M}_{1,1}(x,z) & \mathbf{M}_{1,2}(x,z) & \mathbf{M}_{1,3}(x,z) \\ 1 & 0 & \mathbf{0}_{1\times s} \\ \mathbf{S}(x,z)\,(\mathbf{e}-\mathbf{u}) & \mathbf{S}(x,z)\,\mathbf{u} & \mathbf{S}(x,z)\,\left(z\,\mathbf{B}+x\,\widehat{\mathbf{B}}\right) \end{bmatrix}, \tag{5.20}$$

where

$$\begin{aligned} \mathbf{S}(x,z) &= \left(\mathbf{I_s} - z\,\mathbf{A} - x\,\widehat{\mathbf{A}}\right)^{-1}, \\ \mathbf{M}_{1,1}(x,z) &= (1-\vartheta) + (z\,\mathbf{v} + x\,\widehat{\mathbf{v}})^T\,\mathbf{S}(x,z)\,(\mathbf{e}-\mathbf{u}), \\ \mathbf{M}_{1,2}(x,z) &= \vartheta + (z\,\mathbf{v} + x\,\widehat{\mathbf{v}})^T\,\mathbf{S}(x,z)\,\mathbf{u}, \\ \mathbf{M}_{1,3}(x,z) &= (z\,\mathbf{v} + x\,\widehat{\mathbf{v}})^T\,\mathbf{S}(x,z)\,\left(z\,\mathbf{B}+x\,\widehat{\mathbf{B}}\right) + z\,\mathbf{w}^T + x\,\widehat{\mathbf{w}}^T. \end{aligned}$$

Theoretical results about the spectral radius of the matrix (5.20) are difficult to obtain. Instead, the joint stability is analyzed numerically as follows. Define a desired stiff stability region $\mathcal{S} \subset \mathbb{C}$, for example $\mathcal{S}_\alpha = \{x \in \mathbb{C}^- \ : \ |\mathrm{Im}(x)| \le \tan(\alpha)\,|\mathrm{Re}(x)|\}$ for $A(\alpha)$-stability, and compute numerically the constrained non-stiff stability region:

$$\mathcal{N}_\alpha = \{z \in \mathbb{C} \ : \ \rho(M(x,z)) \le 1\,, \quad \forall\, x \in \mathcal{S}_\alpha\}\,. \tag{5.21}$$

If the $\mathcal{N}_\alpha$ is not degenerate (contains an open subset of the left half plane) then the IMEX combined stability is considered acceptable.

Nonlinear (algebraic) stability of the TSRK methods can be treated in the general linear method framework following [106, 117]. To the best of our knowledge no extension of the algebraic stability to partitioned or implicit-explicit methods is available at this time.

5.5 Convergence aspects

For completeness in this section we review the convergence and possible order reduction of IMEX-TSRK methods with the help of several standard test problems. The proofs of the theorems are given by Zharovsky and Sandu in [116].

We consider an infinite "region of interest" in the complex plane

$$\mathcal{R} = \{z \in \mathbb{C} \ : \ \mathrm{Re}(z) \le z_0 < 0\,, \quad |\mathrm{Im}(z)| \le \alpha\,|\mathrm{Re}(z)|\,, \quad \alpha \ge 0\}\,, \tag{5.22}$$

that contains all the eigenvalues of the stiff Jacobian $h\,g_y$.

5.5.1 Semilinear problems

Consider the split ODE

$$y' = \underbrace{\mu\,y}_{g(y)} + f(y)\,, \quad \mathrm{Re}(\mu) < 0\,, \quad y(t_0) = y_0\,, \tag{5.23}$$

where the stiff part $g(y) = \mu y$ is linear with $\mu \in \mathcal{R}$ (5.22), and the nonstiff part $f(y)$ is nonlinear. We assume that $f(y)$ is continuously differentiable and that its Jacobian is uniformly bounded in a vicinity of the solution, $\|\mathbf{f_y}(y)\| \leq L_f$ with $L_f = \mathcal{O}(1)$ of moderate size. This assumption is slightly stronger than Lipschitz continuity, and L_f plays the role of the Lipschitz constant.

Theorem 5.5.1. *[116][Convergence of IMEX TSRK methods applied to semi-linear test problem] Consider the IMEX TSRK method (5.17) of order p, stage order q for the explicit component, and stage order $\widehat{q}$ for the implicit component. Assume that the implicit component is linearly stable, and that the spectral radius of the implicit stability matrix (5.20) is bounded uniformly in the infinite "region of interest"* (5.22)

$$\rho\left(\widehat{\mathbf{M}}(z)\right) \leq \rho_0 < 1\,, \quad \forall\, z \in \mathcal{R}\,.$$

Then the IMEX method (5.17) is convergent with order $\min(p, q, \widehat{q})$ *for any μ. This convergence order holds uniformly for all levels of stiffness, i.e., for any $h\mu \in \mathcal{R}$.*

It is convenient to construct IMEX TSRK methods (5.17) with $\widehat{q} = q = p$ as such methods do not suffer from order reduction on the semi-linear problem (5.23).

5.5.2 Prothero-Robinson convergence

We consider the Prothero-Robinson (PR) [87] test problem written as a split system (5.1)

$$y' = \underbrace{\mu\,(y - \phi(t))}_{g(y)} + \underbrace{\phi'(t)}_{f(y)}\ , \quad \mu < 0\ , \quad y(0) = \phi(0)\ , \tag{5.24}$$

where the exact solution is $y(t) = \phi(t)$. A numerical method is said to be PR-convergent with order p if its application to (5.24) gives a solution whose the global error decreases as $\mathcal{O}(h^p)$ for $h \to 0$ and $h\mu \to -\infty$. In [42] a similar argument as in the proof of Theorem 5.5.1 was used to reveal that the IMEX method (5.17) is PR-convergent with order $\min(p, q)$.

This is shown as the following Theorem.

Theorem 5.5.2. *[116] Consider the IMEX TSRK method (5.17) of order p, stage order q for the explicit part, and stage order $\widehat{q}$ for the implicit part. Assume that the implicit part is linearly stable, and that the spectral radius of the implicit stability matrix (5.20) is bounded uniformly in the infinite "region of interest"* (5.22)

$$\rho\left(\widehat{\mathbf{M}}(z)\right) \leq \rho_0 < 1\,, \quad \forall\, z \in \mathcal{R}\,.$$

Then the IMEX method (5.17) is PR-convergent with order $\min(p, q)$.

In particular if the explicit stage order is $q = p$, then the PR order of convergence is p. It is convenient to construct IMEX TSRK methods (5.17) with explicit stage order $q = p$, even if $\widehat{q} = p - 1$, as such methods do not suffer from order reduction on the PR problem.

An important aspect of Theorem 5.5.2 is that the proof does not rely on the limit $h\mu \to \infty$. Consequently the order of convergence $\min(p, q)$ established there holds uniformly for all values of $h\mu$ in the "region of interest" in the complex plane. Therefore, this order of convergence is valid for a wide range of stiffness values of the $g(y)$ subsystem.

The discussion can be extended to IMEX-TSRK method applied to semi-discrete partial differential equations. It is well known that Runge Kutta methods can suffer severe order reductions in the presence of non-homogeneous boundary conditions or nonzero source terms [118]. Consider the system

$$y' = \mathbf{V}\, y + b(t)\,,$$

where $\mathbf{V}$ is a spatially discretized differential operator and $b(t)$ represents the non-homogeneous boundary or source terms. This system can be cast in the PR form (5.24) after a transformation of variables that diagonalizes $\mathbf{V}$, and after identifying $-\mu\phi(t) + \phi'(t)$ with the transformed $b(t)$. The analysis reveals that the IMEX-TSRK method applied to this linear PDE system converges with order $\min(p, q)$.

5.5.3 Singular perturbation analysis

Consider the singular perturbation test problem [47]:

$$\begin{bmatrix} y \\ z \end{bmatrix}' = \begin{bmatrix} f(y,z) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \varepsilon^{-1}\, g(y,z) \end{bmatrix}, \quad t_0 \le t \le t_F\,, \quad \begin{bmatrix} y \\ z \end{bmatrix}(t_0) = \begin{bmatrix} y_0 \\ z_0 \end{bmatrix}, \tag{5.25}$$

where the sub-jacobian g_z is invertible in a vicinity of the solution. Consequently, there is a a locally unique solution $z = G(y)$ that satisfies $g\,(y, G(y)) = 0$. For $\varepsilon \to 0$ the system (5.25) reduces to the index-1 differential algebraic equation

$$\begin{bmatrix} y' \\ 0 \end{bmatrix} = \begin{bmatrix} f(y,z) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ g(y,z) \end{bmatrix}, \quad \begin{bmatrix} y \\ z \end{bmatrix}(t_0) = \begin{bmatrix} y_0 \\ G(y_0) \end{bmatrix}, \tag{5.26}$$

and the differential variable evolves according to the non-stiff reduced ODE

$$y' = f\,(y, G(y))\,, \quad t_0 \le t \le t_F\,, \quad y(t_0) = y_0\,. \tag{5.27}$$

Of particular interest are IMEX-TSRK pairs where the implicit component is stiffly accurate

$$\mathbf{v} = \widehat{\mathbf{A}}^T\, \mathbf{e}_s\,, \quad \mathbf{w} = \widehat{\mathbf{B}}^T\, \mathbf{e}_s\,, \quad \vartheta = \mathbf{u}^T\, \mathbf{e}_s\,, \tag{5.28a}$$

and, in addition, it satisfies

$$\rho(\widehat{\mathbf{A}}^{-1}\, \widehat{\mathbf{B}}) < 1\,. \tag{5.28b}$$

For stiffly accurate methods $\mathbf{v}^T\,\widehat{\mathbf{A}}^{-1} = \mathbf{e}_s^T$, c_s=1, and

$$\begin{aligned} \mathbf{w}^T &= \mathbf{v}^T\,\widehat{\mathbf{A}}^{-1}\,\widehat{\mathbf{B}}, && (5.28c)\\ \vartheta &= \mathbf{v}^T\,\widehat{\mathbf{A}}^{-1}\mathbf{u}, \quad 1-\vartheta = \mathbf{v}^T\,\widehat{\mathbf{A}}^{-1}(\mathbf{1}-\mathbf{u})\,. && (5.28d) \end{aligned}$$

From (5.18) the stiff stability matrix at infinity is

$$\widehat{\mathbf{M}}(\infty) = \begin{bmatrix} 1-\vartheta-\mathbf{v}^T\widehat{\mathbf{A}}^{-1}(\mathbf{1}-\mathbf{u}) & \vartheta-\mathbf{v}^T\widehat{\mathbf{A}}^{-1}\mathbf{u} & \mathbf{w}^T-\mathbf{v}^T\,\widehat{\mathbf{A}}^{-1}\,\widehat{\mathbf{B}} \\ 1 & 0 & 0 \\ -\widehat{\mathbf{A}}^{-1}(\mathbf{1}-\mathbf{u}) & -\widehat{\mathbf{A}}^{-1}\mathbf{u} & \widehat{\mathbf{A}}^{-1}\,\widehat{\mathbf{B}} \end{bmatrix}. \qquad (5.29)$$

For stiffly accurate methods we have

$$\widehat{\mathbf{M}}(\infty) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ -\widehat{\mathbf{A}}^{-1}(\mathbf{1}-\mathbf{u}) & -\widehat{\mathbf{A}}^{-1}\mathbf{u} & \widehat{\mathbf{A}}^{-1}\,\widehat{\mathbf{B}} \end{bmatrix}.$$

For stiffly accurate schemes (5.28b), (5.28c) imply that $\rho(\widehat{\mathbf{M}}(\infty)) = \rho(\widehat{\mathbf{A}}^{-1}\,\widehat{\mathbf{B}}) < 1$.

The IMEX TSRK method (5.17) applied to (5.25) reads:

$$\begin{aligned} Y^{[n]} &= (\mathfrak{1}-\mathfrak{u})\,y_{n-1} + \mathfrak{u}\,y_{n-2} + h\,\mathfrak{A}\,f^{[n]} + h\,\mathfrak{B}\,f^{[n-1]}, && (5.30a)\\ Z_i^{[n]} &= (\mathfrak{1}-\mathfrak{u})\,z_{n-1} + \mathfrak{u}\,z_{n-2} + h\,\varepsilon^{-1}\,\widehat{\mathfrak{A}}\,g^{[n]} + h\,\varepsilon^{-1}\,\widehat{\mathfrak{B}}\,g^{[n-1]}\,, && (5.30b)\\ y_n &= (1-\vartheta)\,y_{n-1} + \vartheta\,y_{n-2} + h\,\mathfrak{v}^T f^{[n]} + h\,\mathfrak{w}^T f^{[n-1]}\,, && (5.30c)\\ z_n &= (1-\vartheta)\,z_{n-1} + \vartheta\,z_{n-2} + h\,\varepsilon^{-1}\,\mathfrak{v}^T g^{[n]} + h\,\varepsilon^{-1}\,\mathfrak{w}^T g^{[n-1]}\,. && (5.30d) \end{aligned}$$

We denote the Kronecker products of coefficients by fractional letters, e.g., $\mathfrak{A} = \mathbf{A}\otimes\mathbf{I}$ for an appropriately sized identity matrix in (5.30a), and so on. We use the notation $f^{[n]} = [f^T(Y_1^{[n]}, Z_1^{[n]})\ldots f^T(Y_s^{[n]}, Z_s^{[n]})]^T$; $g^{[n]}$ is defined similarly.

Taking the limit $\varepsilon\to 0$ in (5.30) gives

$$\begin{aligned} Y^{[n]} &= (\mathfrak{1}-\mathfrak{u})\,y_{n-1} + \mathfrak{u}\,y_{n-2} + h\,\mathfrak{A}\,f^{[n]} + h\,\mathfrak{B}\,f^{[n-1]}, && (5.31a)\\ g^{[n]} &= -\left(\widehat{\mathbf{A}}^{-1}\,\widehat{\mathbf{B}}\otimes\mathbf{I}\right)\,g^{[n-1]}\,, && (5.31b)\\ y_n &= (1-\vartheta)\,y_{n-1} + \vartheta\,y_{n-2} + h\,\mathfrak{v}^T f^{[n]} + h\,\mathfrak{w}^T f^{[n-1]}\,, && (5.31c)\\ 0 &= \mathfrak{v}^T g^{[n]} + \mathfrak{w}^T g^{[n-1]} = \left((\mathbf{w}^T - \mathbf{v}^T\,\widehat{\mathbf{A}}^{-1}\,\widehat{\mathbf{B}})\otimes\mathbf{I}\right)\,g^{[n-1]}\,. && (5.31d) \end{aligned}$$

Theorem 5.5.3. *[116] [IMEX TSRK on index-1 DAEs] Consider an IMEX TSRK method* (5.17) *of order p, explicit stage order q, implicit stage order $\widehat{q}$, and whose implicit part satisfies* (5.28c) *and $\rho(\widehat{\mathbf{M}}(\infty)) < 1$. The application of this IMEX-TSRK to the reduced problem* (5.31) *gives solutions with global errors*

$$y_n - y(t_n) = \mathcal{O}\left(h^p\right)\,, \quad z_n - z(t_n) = \mathcal{O}\left(h^{\min(p-1,q-1,\widehat{q})}\right)\,.$$

In addition, if (5.28d) *holds, then*

$$z_n - z(t_n) = \mathcal{O}\left(h^{\min(p,q,\widehat{q}+1)}\right) .$$

The last equation holds for IMEX-TSRK methods with a stiffly accurate implicit component.

In particular it is advantageous to construct IMEX TSRK methods with a stiffly accurate implicit part, $q = p$, and $\widehat{q} \in \{p-1, p\}$.

5.6 Construction of practical IMEX TRSK methods

In this section we construct practical implicit-explicit TSRK schemes with order $p = s + 1$ and stage order $\widehat{q} = q = s$. The relatively large number of stages is necessary to be able to enforce the desired stability properties.

We consider two strategies to construct IMEX pairs. The first one starts with an existing implicit TSRK method with appropriate stability properties (e.g., A or L-stability), and develops the explicit component. This simplifies the construction procedure, but places significant restrictions on the IMEX pairs one can obtain; for example, most A-stable TSRK methods currently available in the literature impose $\mathbf{u} = 0$ for simplicity, which limits the stability of the IMEX pair. A sixth order IMEX TSRK pair was developed in [42] by using this approach. When the implicit component is predetermined, the only free parameters for the explicit component the entries of the $\mathbf{A}$ matrix. The coefficients $\mathbf{B}$ of the nonstiff method result from the explicit stage order conditions (5.4) and the internal consistency conditions (5.12). The $\mathbf{A}$ coefficients are computed using a numerical optimization procedure related to [113], where we optimize the explicit stability region such as to contain the longest possible interval along the imaginary axis. We have also considered maximizing the area of a half ellipse included in the stability region, and with one semi-axis overlapping the imaginary axis; the results are similar to the ones discussed below and are not reported here. The optimization process is done in two steps. First, we explore the parameter space using the genetic algorithm function `ga` in MATLAB optimization toolbox. The best member of this process is then taken as the starting point for MATLAB's `fminsearch` routine, which locally refines the solution and provides a sufficient number of accurate digits.

The second strategy is build simultaneously both the implicit and the explicit components of the IMEX TSRK pair. We construct pairs of orders three and four with $\mathbf{u} \neq 0$, and with implicit components that are both stiffly accurate and $A(\alpha)$ stable. The parameters are found using the Differential Evolution `DE` optimization package [119] by maximizing the length of the imaginary axis segment included in the stability region of the explicit component. The stiff accuracy condition (5.28b) and $A(\alpha)$ stability of the implicit component are included as optimization constraints. For good quality of solutions the genetic optimizer `DE` is run multiple times; each run is initialized with the previous result.

5.6.1 A third order, two stage IMEX pair

Third order IMEX-TSRK methods with two stages are specified by the abscissa vector $\mathbf{c} = [c_1, c_2]$ and the tableaux of coefficients

$$\begin{array}{c|cc|cc} u_1 & \lambda & & \widehat{b}_{11} & \widehat{b}_{12} \\ u_2 & \widehat{a}_{21} & \lambda & \widehat{b}_{21} & \widehat{b}_{22} \\ \hline \vartheta & v_1 & v_2 & w_1 & w_2 \end{array}, \quad \begin{array}{c|cc|cc} u_1 & & & b_{11} & b_{12} \\ u_2 & a_{21} & & b_{21} & b_{22} \\ \hline \vartheta & v_1 & v_2 & w_1 & w_2 \end{array}, \tag{5.32}$$

where all the variables are real parameters. We impose the abscissa values $\mathbf{c} = [0, 1]^T$, the order conditions (5.16), the stage order conditions (5.15), and the stiff accuracy condition (5.28a) to constrain the coefficients. The resulting family of third order IMEX TSRK pairs depends on four free parameters $u_1, u_2, \widehat{a}_{21}, a_{21}$. The implicit component is

$$\begin{array}{c|cc|cc} u_1 & \frac{5-u_2}{12} & & \frac{u_1}{2} & \frac{u_2+6u_1-5}{12} \\ u_2 & \widehat{a}_{21} & \frac{5-u_2}{12} & \frac{5u_2-1}{12} & \frac{2u_2-2\widehat{a}_{21}+2}{3} \\ \hline u_2 & \widehat{a}_{21} & \frac{5-u_2}{12} & \frac{5u_2-1}{12} & \frac{2u_2-2\widehat{a}_{21}+2}{3} \end{array},$$

and the explicit component is

$$\begin{array}{c|cc|cc} u_1 & & & \frac{u_1}{2} & \frac{u_1}{2} \\ u_2 & a_{21} & & \frac{u_2-1}{2} & \frac{3+u_2-2a_{21}}{2} \\ \hline u_2 & \widehat{a}_{21} & \frac{5-u_2}{12} & \frac{5u_2-1}{12} & \frac{2u_2-2\widehat{a}_{21}+2}{3} \end{array}.$$

The optimization procedure could not find any feasible points for A-stability, so we relaxed this restriction by requiring $A(\alpha)$ stability with $\alpha = 75°$. We also restricted the range of parameters to $a_{21}, \widehat{a}_{21} \in [-5, 5]$ and $u_1, u_2 \in [-1, 1]$ to avoid having large coefficients which may cause large roundoff errors. The resulting method is called IMEX-TSRK(2,3), and its coefficients are given in Table 5.8.

The stability regions for each component and for the combined method are shown in Figures 5.1 (a,c,e). The regions of combined stability for all choices of the stiff stability region $\mathcal{S}$ include an open subset of the left complex half plane (thus allowing conditional stability for positive step sizes), as well as a nontrivial part of the imaginary axis (which is desirable for certain PDEs such as advection equations where the Jacobian eigenvalues after spatial discretization could have large imaginary parts [118]).

5.6.2 A fourth order, three stage IMEX pair

We impose the abscissa values $\mathbf{c} = [0, 1/2, 1]^T$, the order conditions (5.16), the stage order conditions (5.15), and the stiff accuracy condition (5.28a) to constrain the coefficients. A

family of fourth order, third stage order IMEX-TSRK pairs is obtained with nine degrees of freedom. We choose $u_1, u_2, u_3, \lambda, a_{21}, a_{31}, a_{32}, \widehat{a}_{21}, \widehat{a}_{31}$ as the free parameters. The resulting IMEX pair has the implicit component

$$\begin{array}{c|ccc|ccc}
u_1 & \lambda & & & \frac{u_1}{6} & \frac{2u_2}{3} & \frac{u_1}{6} - \lambda \\
u_2 & \widehat{a}_{21} & \lambda & & \frac{5+4u_2-24\lambda}{24} & -\frac{2}{3} + \frac{2u_2}{3} + 3\lambda & \frac{23}{24} - \widehat{a}_{21} + \frac{u_2}{6} - 3\lambda \\
u_3 & \widehat{a}_{31} & \frac{4}{3} - 4\lambda & \lambda & \frac{\vartheta+6\lambda-1}{6} & \frac{2}{3} + \frac{2\vartheta}{3} - 4\lambda & -\frac{5}{6} - \widehat{a}_{31} + \frac{\vartheta}{6} + 6\lambda \\
\hline
u_3 & \widehat{a}_{31} & \frac{4}{3} - 4\lambda & \lambda & \frac{\vartheta+6\lambda-1}{6} & \frac{2}{3} + \frac{2\vartheta}{3} - 4\lambda & -\frac{5}{6} - \widehat{a}_{31} + \frac{\vartheta}{6} + 6\lambda
\end{array},$$

and the explicit component

$$\begin{array}{c|ccc|ccc}
u_1 & & & & \frac{u_1}{6} & \frac{2u_2}{3} & \frac{u_1}{6} \\
u_2 & a_{21} & & & \frac{5+4u_2}{24} & -\frac{2}{3} + \frac{2u_2}{3} & \frac{23}{24} - a_{21} + \frac{u_2}{6} \\
u_3 & a_{31} & a_{32} & & \frac{\vartheta-6a_{32}+7}{6} & -\frac{10}{3} + 3a_{32} + \frac{2\vartheta}{3} & \frac{19}{6} - a_{31} + 3a_{32} + \frac{\vartheta}{6} \\
\hline
u_3 & \widehat{a}_{31} & \frac{4}{3} - 4\lambda & \lambda & \frac{\vartheta+6\lambda-1}{6} & \frac{2}{3} + \frac{2\vartheta}{3} - 4\lambda & -\frac{5}{6} - \widehat{a}_{31} + \frac{\vartheta}{6} + 6\lambda
\end{array}.$$

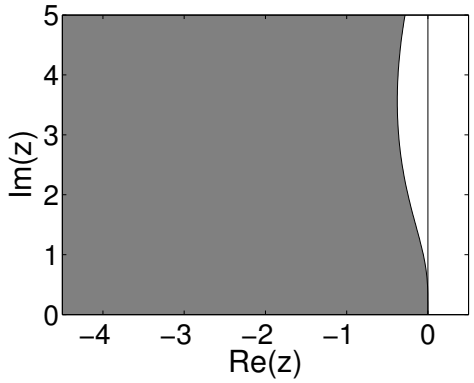
We impose $A(\alpha)$ stability with $\alpha = 75°$. The optimized method is called IMEX-TSRK(3,4) and its coefficients are given in Table 5.8. The stability regions for each component and for the combined method are shown in Figures 5.1 (b,d,f).

5.7 Numerical tests

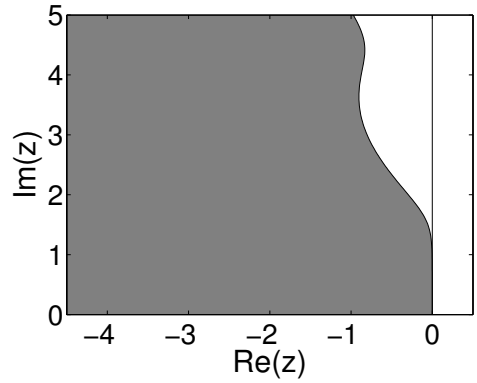
We now illustrate the convergence properties of the proposed IMEX TSRK schemes with the help of several test problems including ODEs (van der Pol equation) and discretized PDEs (a linear advection-reaction system and shallow water equations). The advection-reaction PDE displays stiffness in one additive component of the right hand side, while the van der Pol ODE displays stiffness in one component; they help investigate the convergence behavior and the possible order reduction of various methods. The larger shallow water model is used in coastal and environmental engineering and helps compare the accuracy and relative efficiency of different implicit-explicit time integration schemes.

The advantage of IMEX Runge-Kutta and IMEX linear multistep schemes over fully explicit or fully implicit schemes applied to separable problems (5.1) has been well established [77, 120, 30]. These results, however, do not show a clear favorite: different IMEX schemes perform best on different problems. We compare the new IMEX TSRK schemes with the following Runge Kutta and multistep IMEX methods from literature:

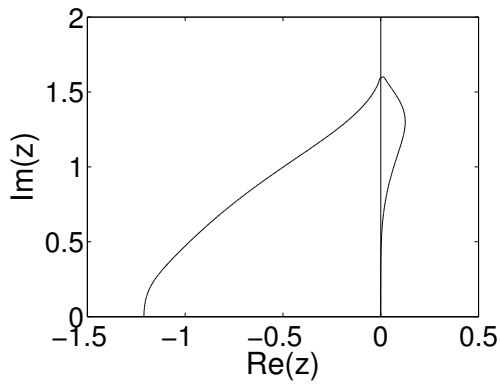
- the second and third order IMEX Runge-Kutta methods $ARS(2,2,2)$ and $ARS(3,4,3)$, respectively, from Ascher, Ruuth and Spiteri in [77];



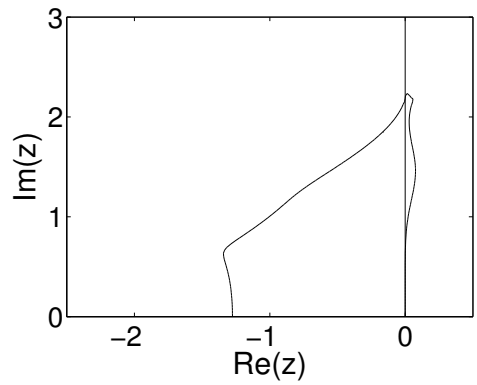
(a) The third order method. Implicit stability region.



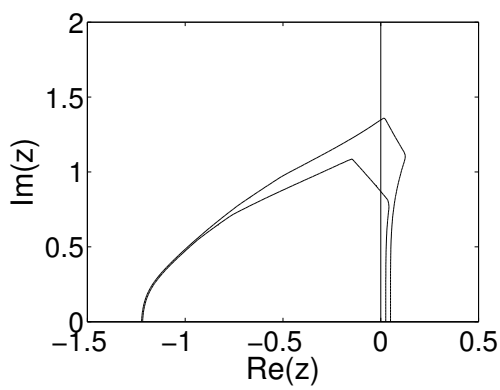
(b) The fourth order method. Implicit stability region.



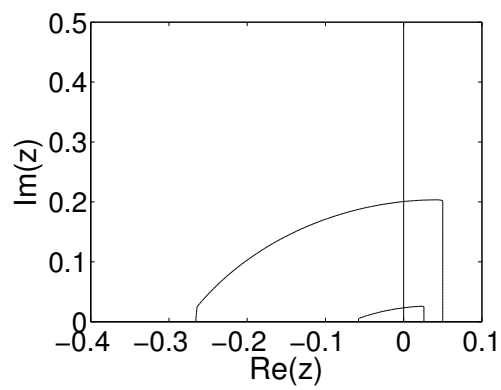
(c) The third order method. Explicit stability region.



(d) The fourth order method. Explicit stability region .



(e) The third order method. Constrained explicit stability regions $\mathcal{N}_\alpha$ for $\alpha \in \{60°, 75°\}$.



(f) The fourth order method. Constrained explicit stability regions $\mathcal{N}_\alpha$ for $\alpha \in \{60°, 75°\}$.

Figure 5.1: (a),(b) Stability regions for the implicit parts of the proposed IMEX-TSRK methods. (c),(d) Stability regions for the explicit parts. (e),(f) Explicit stability regions $\mathcal{N}_\alpha$ in (5.21) are constrained by the $A(\alpha)$ stability of the implicit part. The explicit stability regions $\mathcal{N}_\alpha$ shown here correspond to $\alpha = 60°$ (outer contours) and $\alpha = 75°$ (inner contours).

- the fourth and fifth order methods $ARK4(3)6L[2]SA$ and $ARK5(4)8L[2]SA$, named here $KC(5,6,4)$ $KC(7,8,5)$, from Kennedy and Carpenter [120];
- the IMEX linear multistep methods of orders 2–5 from Hundsdorfer and Ruuth [30].

Here we use the naming of these methods proposed in [30].

For each of the test cases below a reference solution was computed with MATLAB's `ode15s` routine with the very tight tolerances atol = rtol = $2.22045 \cdot 10^{-14}$. The initial steps needed by multistep methods and TSRK methods are also calculated by `ode15s` with the same tolerance settings. All numerical errors are measured at the final simulation times against the corresponding reference solutions. All the experiments are performed on a workstation with 4 Intel Xeon E5-2630 processors running Linux Kernel 3.8.4.

5.7.1 Advection-reaction system

This test case is borrowed from [30] and is described by the following PDE system:

$$\partial_t \begin{bmatrix} y \\ z \end{bmatrix} = -\partial_x \begin{bmatrix} \alpha_1 y \\ \alpha_2 z \end{bmatrix} + \begin{bmatrix} -k_1 & k_2 \\ k_1 & -k_2 \end{bmatrix} \cdot \begin{bmatrix} y \\ z \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}, \quad (t,x) \in [0,1] \times [0,1], \tag{5.33}$$

with parameters

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = 10^6 \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{5.34}$$

and with the following initial and boundary values

$$y(x,0) = 1 + s_2 x, \quad z(x,0) = \frac{k_1}{k_2} y(x,0) + \frac{1}{k_2} s_2, \quad y(0,t) = 1 - \sin(12t)^4. \tag{5.35}$$

The space discretization is done with fourth order finite differences in the interior and third order upwind biased finite differences at the borders of the spatial domain. The space grid consists of 400 uniformly distributed nodes. We treat the nonstiff advection term explicitly, and the stiff reaction term implicitly.

For all the schemes tested, the solutions are computed with different fixed time steps. They are compared against MATLAB's reference solution at the final time. The errors in L_1-norms are plotted against the time step in Figure 5.2. We observe that $ARS(3,4,3)$ shows the same second order behavior $ARS(2,2,2)$. Both KC schemes suffer from significant order reduction. However, the IMEX-BDF schemes, IMEX-TSRK(2,3), and IMEX-TSRK(3,4) do not exhibit order reduction. For the same step size IMEX-TSRK schemes yield much smaller errors than IMEX-BDF schemes of the same order. The sixth order scheme IMEX-TSRK(5,6) does not show the theoretical order, but has the advantage of high accuracy compared to other schemes under the same step size. In Figure 5.3 the errors are plotted as functions of CPU time (in seconds) with the same sequence of time step sizes used in the convergence

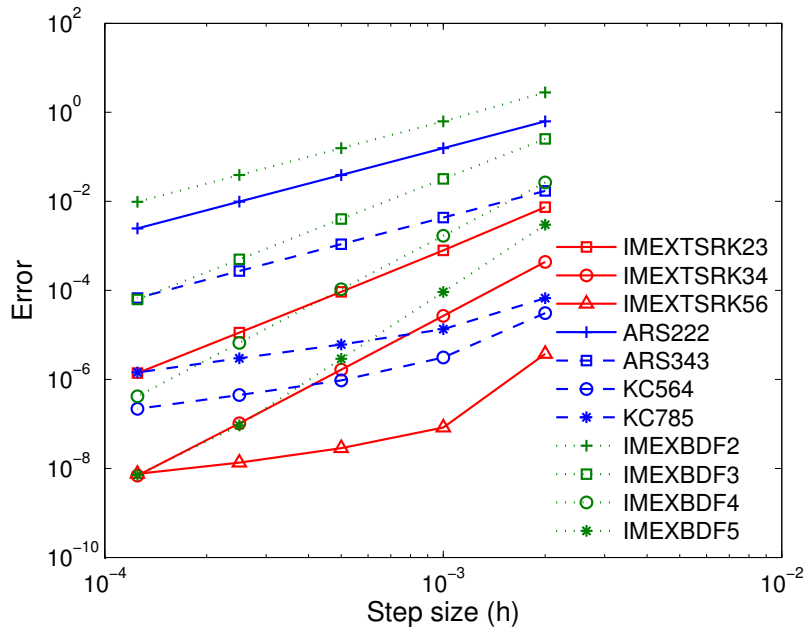


Figure 5.2: Convergence of IMEX-TSRK, IMEX-RK, and IMEX-BDF schemes for the advection-reaction problem (5.33). The solution errors (measured at the final time in L_1 norm) are plotted against the simulation time step h.

results. Among all the third order schemes, IMEX-TSRK(2,3) shows the highest efficiency. For fourth order ones, IMEX-TSRK(3,4) and IMEX-BDF4 almost coincide; KC(5,6,4) leads for the low accuracy part and lags behind for higher accuracy (errors smaller than 10^{-6}).

5.7.2 Van der Pol equation

The first non-linear test is the van der Pol equation

$$\frac{d}{dt}\begin{bmatrix} y \\ z \end{bmatrix} = f(y,z) + g(y,z) = \begin{bmatrix} z \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ ((1-y^2)z - y)/\epsilon \end{bmatrix}, \quad 0 \le t \le 0.55139, \tag{5.36}$$

with parameter values taken from [33]

$$\epsilon = 10^{-6}, \quad y(0) = 2, \quad z(0) = -\frac{2}{3} + \frac{10}{81}\epsilon - \frac{292}{2187}\epsilon^2 - \frac{1814}{19683}\epsilon^3 + \mathcal{O}(\epsilon^4)\,. \tag{5.37}$$

We integrate equation (5.36) treating the first term on the right explicitly and the other term implicitly with different fixed step sizes. The results at the final time are compared against the MATLAB reference solution. The errors for the stiff component z are plotted against step sizes in Figure 5.4. All IMEX-RK methods except ARS(2,2,2) exhibit order reduction. All

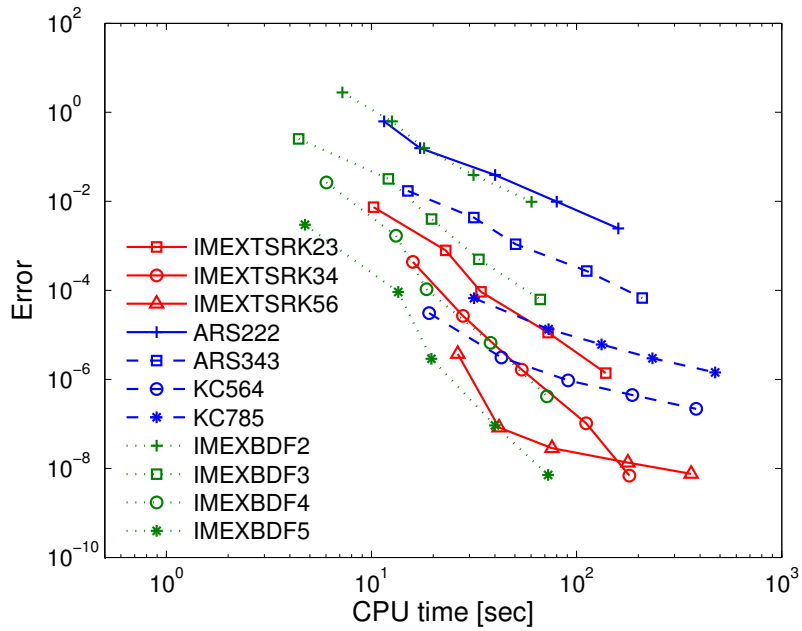


Figure 5.3: Global errors versus CPU time for the advection-reaction problem (5.33) solved with IMEX-TSRK, IMEX-RK, and IMEX-BDF schemes.

IMEX-BDF methods display the expected orders. IMEX-TSRK(2,3) and IMEX-TSRK(3,4) also show satisfactory convergence behavior and still retain the advantage of higher accuracy compared to the corresponding IMEX-BDF methods of the same order. IMEX-TSRK(5,6) suffers from order reduction in the high accuracy regime. This is also the case for another high order method $KC785$. The IMEX-RK methods use more right hand side evaluations per step than the IMEX-TSRK methods, and exhibit order reduction. This test case illustrates the benefit of the proposed IMEX-TSRK schemes.

5.7.3 Shallow water equations

Semi-implicit time integration has become popular in the numerical weather and climate prediction communities, e.g., as an effective way to treat gravity waves. Semi-implicit integration uses an IMEX scheme, with the implicit method applied to the linearized right hand side operator, and the explicit method to the remaining nonlinear part. This approach goes back to [121].

In this test we solve the two dimensional shallow water equations using a semi-implicit integration approach. The shallow water system under consideration is

$$\partial_t H = -\partial_x(uH) - \partial_y(vH)$$

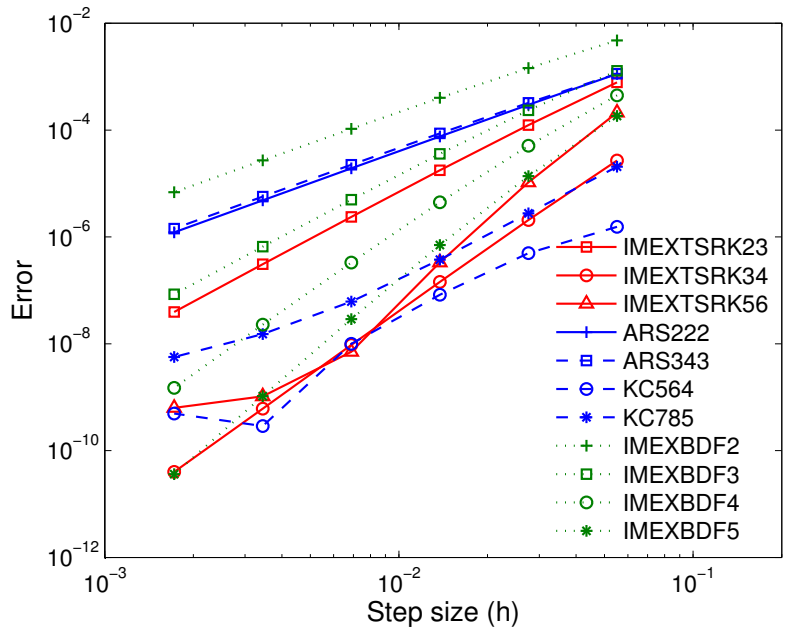


Figure 5.4: Convergence of IMEX-TSRK, IMEX-RK, and IMEX-BDF schemes for the stiff variable z of the van der Pol equation (5.36) with $\epsilon = 10^{-6}$. The solution errors (measured at the final time) are plotted against the simulation time step h.

$$\begin{aligned} \partial_t(uH) &= -\partial_x(u^2H + \frac{1}{2}gH^2) - \partial_y(uvH) \\ \partial_t(vH) &= -\partial_x(uvH) - \partial_y(v^2H + \frac{1}{2}gH^2) \end{aligned} \tag{5.38}$$

on the unit square domain $(x, y) \in [0, 1] \times [0, 1]$. Here H is the fluid height, and u and v are the flow velocity components. The initial conditions at $t_0 = 0$ are

$$u(t_0, x, y) = 0\,, \quad v(t_0, x, y) = 0\,, \quad H(t_0, x, y) = 1 + \exp(-\|(x, y) - (c_1, c_2)\|_2^2)\,, \tag{5.39}$$

where the Gaussian height profile is described by $c_1 = 1/3$ and $c_2 = 2/3$. Furthermore, the local gravity constant is $g = 9.81$ $[m/sec^2]$. Reflective boundary conditions are used for velocity terms and free-slip boundary conditions are used for height. The initial condition produces traveling waves that suffer multiple reflections at the boundaries.

This test follows the shallow water example in [122]. A second order finite difference scheme is used for space discretization. The resulting semi-discrete ODE system is

$$\frac{\mathrm{d}}{\mathrm{d}t}U(t) = F_{\text{swe}}\left(U(t)\right)\,, \tag{5.40}$$

where $U(t)$ is the discrete counterpart of the vector of unknowns (H, uH, vH). Denote by $J_{\text{swe}} = \partial F_{\text{swe}}/\partial U$ the Jacobian of the discretized shallow water operator. We split the right

hand side of equation (5.40) into a stiff part

$$g\left(U(t)\right) = J_{\text{swe}}\big(U(t)\big) \cdot U(t)\,, \tag{5.41}$$

and a non-stiff part

$$f(U(t)) = F_{\text{swe}}(U(t)) - g\left(U(t)\right)\,. \tag{5.42}$$

Here g is a non-linear function in $U(t)$. The numerical solution of the implicit part of the method uses a simplified Newton iteration that requires only a single factorization of $\mathbf{I} - h\lambda J_{swe}(U_n)$ per step. Here U_n is the numerical approximation to $U(t_n)$. Note that a semi-implicit approach can be obtained similarly by choosing a linear stiff part, i.e., a splitting of the form $g(U) = J_{\text{swe}}\left(U_n\right) \cdot (U(t) - U_n)$ on each step $t \in [t_n, t_{n+1}]$.

The system is integrated from $t_0 = 0$ to $t_F = 1$ [second] with a sequence of step sizes that starts with $h = 0.05$ [seconds]; the step size is halved for each subsequent run. The Jacobian is updated only once in each time step for all methods tested. The errors in L_1-norms are plotted against the time step size in Figure 5.5 and against the CPU time in Figure 5.6. Figure 5.5 reveals that all the schemes display the theoretical orders of convergence. IMEX-RK methods yield the smallest errors for a given time step. However, IMEX-TSRK are slightly superior to both IMEX-RK and IMEX-BDF in terms of efficiency. To achieve the same order, IMEX-TSRK requires fewer stages than IMEX-RK, and the performance gap between the two families gets larger as the order increases. IMEX-BDF methods have the lowest cost per time step, however, the larger number of time steps attenuates this advantage. In general, high order IMEX schemes are more efficient than low order ones. But for a specific order, the selection of one IMEX family is not a clear cut choice. The efficiency is highly dependent on implementation, and the relative performance is likely to vary for different problems. Intrinsic method properties such as stability, accuracy, and lack of order reduction, become even more important for in the context of complex multiphysics systems, for which IMEX schemes are designed.

5.8 Conclusions

This paper develops a new family of implicit-explicit time integrators based on pairs of two-step Runge-Kutta methods. The class of schemes of interest is characterized by stage consistency (same abscissae) and linear invariant preservation (same weights). The study of order conditions for partitioned TSRK methods reveals that, in case of high stage orders, no additional coupling conditions need to be satisfied. Therefore this framework offers extreme flexibility in pairing implicit and explicit methods. We construct two practical IMEX TSRK methods, of orders three and four, respectively. In both cases the implicit parts are stiffly-accurate. The corresponding explicit parts have been constructed such as to maximize their stability properties; their coefficients were found via a numerical optimization approach. A convergence analysis for the Prothero-Robinson problem shows that the effective order of

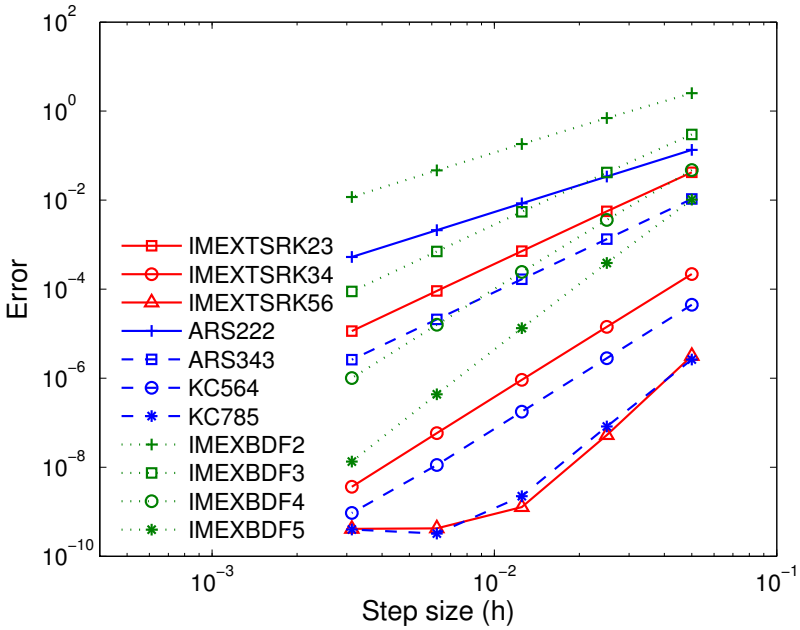


Figure 5.5: Convergence of IMEX-TSRK, IMEX-RK, and IMEX-BDF schemes for the shallow water equations (5.38). The solution errors (measured at the final time in L_1 norm) are plotted against the simulation time step h.

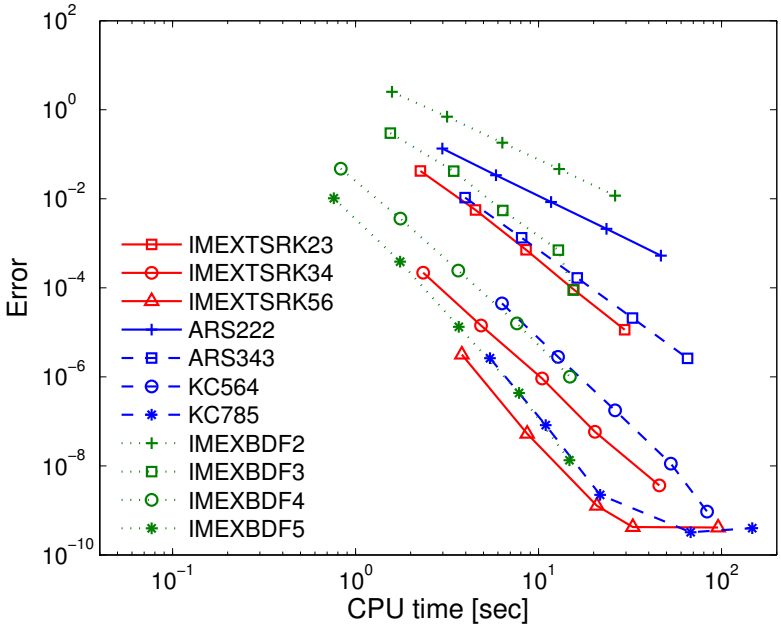


Figure 5.6: The global error versus CPU time for IMEX-TSRK, IMEX-RK, and IMEX-BDF schemes applied to dove the shallow water equations (5.38).

IMEX schemes in case of stiffness equals the stage order of the explicit part; a slight order reduction (from p to $q = p-1$) is expected in the general case, but can be avoided in principle by using explicit methods with $q = p$.

Numerical examples include an advection-reaction system, the Van der Pol equation, and a semi-implicit integration of shallow water equations. Representative existing IMEX schemes of Runge Kutta and linear multistep types are included for comparison. The results show that IMEX TSRK methods have favorable properties for stiff problems where IMEX Runge-Kutta methods suffer from sever order reduction. The larger shallow water test shows that the IMEX-TSRK methods are competitive in terms of accuracy and efficiency with the currently available families of methods.

The new framework allows to obtain IMEX schemes of order p and stage order $p-1$ using only $s = p - 1$ stages. For $p \geq 4$ this is an advantage over existing IMEX Runge-Kutta schemes. The proposed framework offers extreme flexibility in the construction of new partitioned methods, since no coupling conditions are necessary.

Table 5.1: Coefficients of the third order IMEX TSRK method

$$
\begin{aligned}
\mathbf{c} &= \begin{bmatrix} 0 & 1 \end{bmatrix}^T \\
\mathbf{u} &= \begin{bmatrix} 1 & 0 \end{bmatrix}^T \\
\mathbf{A} &= \begin{bmatrix} 0 & 0 \\ 1.825912381819746 & 0 \end{bmatrix} \\
\mathbf{B} &= \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -0.325912381819746 \end{bmatrix} \\
\widehat{\mathbf{A}} &= \begin{bmatrix} 5/12 & 0 \\ 0.815364469611720 & 5/12 \end{bmatrix} \\
\widehat{\mathbf{B}} &= \begin{bmatrix} 1/2 & 1/12 \\ -1/12 & -0.148697802945053 \end{bmatrix} \\
\vartheta &= 0 \\
\mathbf{v} &= \begin{bmatrix} 0.81536446961172 & 5/12 \end{bmatrix}^T \\
\mathbf{w} &= \begin{bmatrix} -1/12 & -0.148697802945053 \end{bmatrix}^T
\end{aligned}
$$

Table 5.2: Coefficients of the fourth order IMEX TSRK method

$$\mathbf{c} = \begin{bmatrix} 0 & 0.5 & 1 \end{bmatrix}^T$$

$$\mathbf{u} = \begin{bmatrix} 0.237173125722858 & 0.528507103963907 & 0.278367184188801 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ 1.215905100969091 & 0 & 0 \\ -1.012523027539376 & 1.143656957186445 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.039528854287143 & 0.158115417148572 & 0.039528854287143 \\ 0.296417850660651 & -0.314328597357395 & -0.169487250308440 \\ 0.069404240178355 & 0.283215661018536 & 0.794613353344841 \end{bmatrix}$$

$$\widehat{\mathbf{A}} = \begin{bmatrix} 0.235880190910947 & 0 & 0 \\ 1.018461419211348 & 0.235880190910947 & 0 \\ 0.290136315264242 & 0.389812569689545 & 0.235880190910947 \end{bmatrix}$$

$$\widehat{\mathbf{B}} = \begin{bmatrix} 0.039528854287143 & 0.158115417148572 & -0.196351336623804 \\ 0.060537659749704 & 0.393311975375446 & -0.679684141283538 \\ 0.115608054942414 & -0.091275974184587 & 0.338206027566240 \end{bmatrix}$$

$$\vartheta = 0.278367184188801$$

$$\mathbf{v} = \begin{bmatrix} 0.290136315264242 & 0.389812569689545 & 0.235880190910947 \end{bmatrix}^T$$

$$\mathbf{w} = \begin{bmatrix} 0.115608054942414 & -0.091275974184587 & 0.338206027566240 \end{bmatrix}^T$$

Chapter 6

Application of AMF to high order LIRK methods

6.1 Introduction

A frequently used approach to solve partial differential equations (PDEs) is the method of lines, where the spatial derivative terms are discretized first using techniques such as finite differences, finite volumes or finite elements, and then integrating the resulting system of ordinary differential equations (ODEs) in time. Discretization of PDEs with linear terms in space leads to a semi-linear ODE system of the form

$$y' = F(y,t) = \mathbf{L}\, y + f(y,t)\,, \quad y \in \mathbb{R}^N \tag{6.1}$$

where $\mathbf{L}$ is a spatial linear operator and $f(y,t)$ is nonlinear. We consider the case where the linear term has a fast characteristic time scale and the nonlinear term has a slow characteristic time scale. Due to the Courant-Friedrichs-Levy (CFL) condition, the step size of an explicit time integrator is restricted by the fastest time scale. To alleviate the time step constraint imposed by the stiff linear term with a reasonably low computational cost, linearly implicit methods (particular cases of implicit-explicit methods) treat the stiff linear term implicitly while the nonlinear term explicitly.

Various families of linearly implicit integration methods have been proposed and successfully applied to solve PDEs with linear dispersion and dissipation [123, 93, 124, 125]. Fully implicit schemes solve at each step a linear system where the matrix involves the Jacobian of the right hand side function. Efficient schemes specialized in the solution of (6.1) such as linearly implicit methods use only the part of the Jacobian associated with the stiff linear term, resulting in linear systems of the form

$$(\mathbf{I} - h\,\gamma\,\mathbf{L})\; x = \ell, \quad x, \ell \in \mathbb{R}^N, \tag{6.2}$$

where h is the step size, γ is a parameter of the integration scheme, and the right hand side ℓ is determined by the method.

The matrix $\mathbf{L}$ is usually sparse but has a large bandwidth, especially when high order spatial discretization schemes are applied. Consequently the LU factorization of the matrix in (6.2) can be very costly in large scale problems. One approach to increase efficiency is to apply iterative solvers to (6.2), as in [126], but there are associated challenges related to preconditioning and convergence.

An alternative approach to increase the computational efficiency is approximate matrix factorization (AMF). Assuming that the matrix is a sum of simpler matrices

$$\mathbf{L} = \sum_{r=1}^{R} \mathbf{L}_r \tag{6.3}$$

AMF replaces the system matrix (6.2) with a product of simpler, and easier to factorize, matrices

$$\mathbf{I} - h\,\gamma\,\mathbf{L} \approx \mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}} = \prod_{r=1}^{R} \left(\mathbf{I} - h\,\gamma\,\mathbf{L}_r\right). \tag{6.4}$$

The approximation formula (6.4) defines implicitly the matrix $\widetilde{\mathbf{L}}$ as

$$\widetilde{\mathbf{L}} = \mathbf{L} + \sum_{k=2}^{R} (-h\gamma)^{k-1} \sum_{1 \le i_1 < i_2 < \cdots < i_k \le R} \mathbf{L}_{i_1}\,\mathbf{L}_{i_2} \ldots \mathbf{L}_{i_k}. \tag{6.5}$$

For example, consider $\mathbf{L}$ to be the discrete two-dimensional Laplace operator. It can be written in the form (6.3) where $\mathbf{L}_1$ and $\mathbf{L}_2$ correspond to the derivatives along the x direction and y direction, respectively. The AMF approximation corresponds to the alternating directions factorization [127, 128]

$$\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}} := \left(\mathbf{I} - h\,\gamma\mathbf{L}_1\right)\,\left(\mathbf{I} - h\,\gamma\mathbf{L}_2\right)$$

where

$$\widetilde{\mathbf{L}} = \mathbf{L} - h\,\gamma\,\mathbf{L}_1\,\mathbf{L}_2. \tag{6.6}$$

The idea of using AMF to speed up calculations in implicit time integration has appeared multiple times in the literature. Sandu [129] discussed a family of methods named ELADI, which are Rosenbrock-W schemes that make use of AMF to speed up calculations. The order of the resulting discretization remains unchanged since Rosenbrock-W schemes can accommodate any Jacobian approximation. Houwen et al. provided a survey for AMF methods applied in the context of several different linear integration schemes, and provide stability results for such schemes [130]. The discussion is limited to methods of low order, two and three, presumably due to the inaccurate nature of the AMF approximation. Gonzalez [38] proposed a way to apply AMF-refinements to second-order and third-order Rosenbrock-type

methods for solving advection-diffusion-reaction PDEs. But their methods are limited to the low or medium precision level, and no generalization to higher order is supplied. In [39] Beck et al. compared the efficiency of AMF versus Krylov based approaches to the solution of linear systems in the context of Newton iterations arising in Radau [47] and Peer [131] integration methods. These methods avoid the issue of order degradation, through the use of integration schemes in which the Jacobian of the spatial discretization does not appear explicitly. They conclude that AMF methods are extremely efficient, particularly when low accuracy solutions are sought. Berzins et al. [132, 133] presented a method for solving the linear system in a Newton iteration arising from several classes of time integration methods including theta methods, backward differential formulas, and implicit-explicit (IMEX) multistep methods. They performed and analysis of the error arising from operator splitting and provided a method to control timesteps such as to guarantee Newton convergence when using AMF. Since AMF is only used to speed up the solution of the nonlinear equations, the error does not affect the order of accuracy at the time stepping level. Calvo and Gerisch [134] applied AMF to a form of linearly implicit Runge-Kutta (LIRK) methods that avoids the computation of matrix-vector products. First order of convergence is obtained by using a third-order LIRK method, and improved to second order by adding a correction to the solution at each time step.

None of the existing methods that take advantage of AMF can provide highly accurate results and there is still room for improvement in efficiency. The goal of this work is to achieve high accuracy while maintaining a low computational cost. The focus is on using AMF with linearly implicit Runge-Kutta methods of high order. The main contribution of this paper is to account for the inherent inaccuracy of AMF through low cost refinements of the stage values and to recover the accuracy of the underlying time discretization.

The remainder of this paper is organized as follows. Section 6.2 introduces LIRK methods and the existing approaches to apply AMF. Section 6.3 presents a new strategy to incorporate AMF by using an inexpensive stage refinement procedure. An error analysis explains how this strategy solves the accuracy degradation issue that affects existing approaches. Stability issues are also investigated. Section 6.4 reports numerical rests for a variety of test problems of different dimensions and different degrees of stiffness, and illustrates the convergence behavior and efficiency of the approach. Conclusions are drawn in Section 6.5.

6.2 Linearly implicit Runge-Kutta methods

A general linearly implicit Runge-Kutta (LIRK) scheme proposed by Calvo, de Frutos, and Novo [93] is obtained by applying the IMEX Runge-Kutta methods [32, 34]

$$Y_i = y_n + h \sum_{j=1}^{i-1} a_{i,j}\, f(Y_j) + h \sum_{j=1}^{i} \widehat{a}_{i,j}\, g(Y_j), \tag{6.7a}$$

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j\, f(Y_j) + h \sum_{j=1}^{s} \widehat{b}_j\, g(Y_j), \tag{6.7b}$$

to solve (6.1) where the stiff component is linear, $g(y) := \mathbf{L}y$:

$$(\mathbf{I} - h\,\widehat{a}_{i,i}\,\mathbf{L})\; Y_i = \ell_i := y_n + h \sum_{j=1}^{i-1} a_{i,j}\, f(Y_j) + h \sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\mathbf{L}\, Y_j, \tag{6.8a}$$

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j\, f(Y_j) + h \sum_{j=1}^{s} \widehat{b}_j \mathbf{L}\, Y_j. \tag{6.8b}$$

Order conditions for LIRK methods are derived in [93], and simplifications of the IMEX Runge-Kutta conditions are possible due to the special form of the nonlinear term.

The coefficients $\widehat{a}_{i,i}$ in practical methods are chosen to be equal to the same value γ for computational efficiency, as this allows to reuse the same LU factorization in all stages. A linear transformation of variables allows for a reformulation of the stage equations (6.8a) in a form that avoids explicit multiplications by the matrix $\mathbf{L}$

$$(\mathbf{I} - h\,\gamma\,\mathbf{L})\; U_i = y_n + \sum_{j=1}^{i-1} \frac{\widehat{a}_{i,j} - a_{i,j}}{\gamma}\; Y_j + h \sum_{j=1}^{i-1} a_{i,j}\, F(Y_j), \tag{6.9a}$$

$$Y_i = U_i - \sum_{j=1}^{i-1} \frac{\widehat{a}_{i,j} - a_{i,j}}{\gamma}\; Y_j. \tag{6.9b}$$

Moreover, it is convenient to choose pairs of methods with the same weights $b_j = \widehat{b}_j$ in which case the next step solution (6.8b) is

$$y_{n+1} = y_n + h \sum_{j=1}^{s} b_j\, F(Y_j). \tag{6.9c}$$

Calvo and Gerisch [134] studied the use of LIRK methods with AMF. The approximation is obtained by replacing $\mathbf{I} - h\gamma\mathbf{L}$ with $\mathbf{I} - h\gamma\widetilde{\mathbf{L}}$ in (6.9a), or equivalently, by using the matrix $\widetilde{\mathbf{L}}$ instead of $\mathbf{L}$ in (6.8)

$$\left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)\; \widetilde{Y}_i = y_n + h \sum_{j=1}^{i-1} a_{i,j}\, f(\widetilde{Y}_j) + h \sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\widetilde{\mathbf{L}}\,\widetilde{Y}_j,$$

$$\widetilde{y}_{n+1} = y_n + h \sum_{j=1}^{s} b_j\, f(\widetilde{Y}_j) + h \sum_{j=1}^{s} \widehat{b}_j\,\widetilde{\mathbf{L}}\,\widetilde{Y}_j.$$

It can also be regarded as a direct application of the LIRK method (6.8) to solve the perturbed ODE system

$$\widetilde{y}' = \widetilde{\mathbf{L}}\,\widetilde{y} + f(\widetilde{y}) = F(\widetilde{y}) - h\,\gamma\,\left(\mathbf{L} - \widetilde{\mathbf{L}}\right)\,\widetilde{y} \tag{6.10}$$

instead of the original system (6.1). The first-order behavior of this approach has been explained in [134] by the fact that the perturbation added in (6.10) changes the solution over one time step by $\mathcal{O}(h^2)$.

To recover second order Calvo and Gerisch [134] apply corrections to the numerical solution obtained by LIRK with AMF. One such correction has the form

$$y_{n+1} = \widetilde{y}_{n+1} + h^2\,\gamma\,\left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1}\,\left(\mathbf{L} - \widetilde{\mathbf{L}}\right)\,y_n\,. \tag{6.11}$$

The matrix inverse in the correction term uses the same LU factorization as the solution. The presence of the matrix inverse in the correction term ensures the linear stability of the new solution (6.11). It is also noted in [134] that "regaining order three using corrections similar to (6.11) is not feasible with a computational cost comparable with the cost of recovering order two".

6.3 LIRK-AMF methods with stage refinement

We consider a different way to incorporate AMF into LIRK methods, which forms the basis of all approaches presented in this paper. In order to keep the right-hand side of the original ODE system (6.1), we only approximate $\mathbf{I} - h\,\gamma\mathbf{L}$ with $\mathbf{I} - h\,\gamma\widetilde{\mathbf{L}}$ when computing the Runge-Kutta stages:

$$\left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)\,\widetilde{Y}_i \;=\; \widetilde{\ell}_i := y_n + h\sum_{j=1}^{i-1} a_{i,j}\,f(\widetilde{Y}_j) + h\sum_{j=1}^{i-1}\widehat{a}_{i,j}\,\mathbf{L}\,\widetilde{Y}_j, \tag{6.12a}$$

$$\widetilde{y}_{n+1} \;=\; y_n + h\sum_{j=1}^{s} b_j\,f(\widetilde{Y}_j) + h\sum_{j=1}^{s}\widehat{b}_j\mathbf{L}\,\widetilde{Y}_j. \tag{6.12b}$$

Thus the new method uses an inexact Jacobian for the implicit part in the LIRK scheme. The change of the left hand side in (6.12a) will affect the solution, and a correction is needed to restore accuracy.

Consider the solution of the original stage equation (6.8a)

$$(\mathbf{I} - h\,\gamma\,\mathbf{L})\;Y_i - \ell_i = 0$$

by simplified Newton iterations of the form:

$$Y_i^{(k)} = Y_i^{(k-1)} - \left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1}\cdot\left((\mathbf{I} - h\,\gamma\,\mathbf{L})\;Y_i^{(k-1)} - \ell_i\right). \tag{6.13}$$

For example, the direct solution is:

$$Y_i^{(0)} \;=\; \left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1}\,\ell_i,$$

and the solution after one refinement iteration is:

$$\begin{aligned} Y_i^{(1)} &= Y_i^{(0)} - \left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1} \cdot \left((\mathbf{I} - h\,\gamma\,\mathbf{L})\; Y_i^{(0)} - \ell_i\right) \\ &= Y_i^{(0)} - \left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1} \cdot \left(h\,\gamma\,\widetilde{\mathbf{L}} - h\,\gamma\,\mathbf{L}\right)\; Y_i^{(0)}. \end{aligned} \tag{6.14}$$

Next we analyze the linear system solution errors and investigate how this errors propagate to affect the solution at the next step.

6.3.1 Error analysis

Consider the exact stage solution

$$Y_i = (\mathbf{I} - h\,\gamma\,\mathbf{L})^{-1}\;\ell_i.$$

The linear system solution error after k iterations is defined as

$$\varepsilon_i^{(k)} = Y_i^{(k)} - Y_i.$$

From (6.13) we obtain

$$\varepsilon_i^{(k)} = \varepsilon_i^{(k-1)} - \left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1} \cdot (\mathbf{I} - h\,\gamma\,\mathbf{L})\;\varepsilon_i^{(k-1)} \tag{6.15a}$$

$$= \left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1} \cdot \left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}} - (\mathbf{I} - h\,\gamma\,\mathbf{L})\right)\;\varepsilon_i^{(k-1)} \tag{6.15b}$$

$$= -\left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1} \cdot \left(h\,\gamma\,\widetilde{\mathbf{L}} - h\,\gamma\,\mathbf{L}\right)\;\varepsilon_i^{(k-1)}. \tag{6.15c}$$

Nonstiff or moderately stiff case. In the nonstiff or moderately stiff case we have $\|h\mathbf{L}_i\| = \mathcal{O}(h)$, therefore $\|\widetilde{\mathbf{L}} - \mathbf{L}\| = \mathcal{O}(h)$ and

$$\left\|\left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1} \cdot \left(h\,\gamma\,\widetilde{\mathbf{L}} - h\,\gamma\,\mathbf{L}\right)\right\| = \mathcal{O}\left(h^2\right).$$

Consequently from (6.15c) the error decrease is

$$\left\|\varepsilon_i^{(k+1)}\right\| = \mathcal{O}(h^2)\;\left\|\varepsilon_i^{(k)}\right\| \quad\Rightarrow\quad \left\|\varepsilon_i^{(k)}\right\| = \mathcal{O}\left(h^{2k+2}\right).$$

Highly stiff case. For the highly stiff case $\|h\mathbf{L}_i\| \gg 1$. We make the assumption that, for any $\overline{h}$ there exists $0 < \rho(\overline{h}) < 1$ such that the following matrix norm is uniformly bounded for any step size smaller than $\overline{h}$:

$$\left\|\left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1} \cdot \left(h\,\gamma\,\widetilde{\mathbf{L}} - h\,\gamma\,\mathbf{L}\right)\right\| \le \rho(\overline{h}) < 1\,, \quad \forall h:\; 0 \le h \le \overline{h}.$$

In the highly stiff case the error decrease equation (6.15c) leads to

$$\left\|\varepsilon_i^{(k)}\right\| \;=\; \rho\,\left\|\varepsilon_i^{(k-1)}\right\| \quad\Rightarrow\quad \left\|\varepsilon_i^{(k)}\right\| = \rho^k\,\varepsilon_i^{(0)}.$$

We expect that the convergence rate will decrease with increasing step sizes, i.e., $\rho(\overline{h}) \to 1$ when $\overline{h} \to \infty$.

Example 1 (Dimensional splitting of the discrete diffusion operator on a Cartesian grid)**.** *Consider the two-dimensional diffusion operator with periodic boundary conditions on a domain of size $L_X \times L_Y$. It is discretized on an $M \times N$ grid of size Δx, Δy. We perform a dimensional splitting. The error equation* (6.15c) *can be written as*

$$(\mathbf{I} - h\,\gamma\,\mathbf{L}_1)\;(\mathbf{I} - h\,\gamma\,\mathbf{L}_2)\;\varepsilon_i^{(k)} = (h\,\gamma\,\mathbf{L}_1)\;(h\,\gamma\,\mathbf{L}_2)\;\varepsilon_i^{(k-1)}. \tag{6.16}$$

Consider the discrete frequencies

$$-\frac{M}{2} \le m \le \frac{M}{2} - 1, \quad -\frac{N}{2} \le n \le \frac{N}{2} - 1, \quad \tilde{m} = \frac{2\pi m}{L_X}, \quad \tilde{n} = \frac{2\pi n}{L_Y}.$$

A discrete Fourier transform applied to (6.16) *gives the following error equation for each spatial mode (m, n) of the error:*

$$(1 + h\,\gamma\,\tilde{m}^2)(1 + h\,\gamma\,\tilde{n}^2)\,\hat{\varepsilon}_{m,n}^{(k)} = (-h\,\gamma\,\tilde{m}^2)(-h\,\gamma\,\tilde{n}^2)\,\hat{\varepsilon}_{m,n}^{(k-1)}$$

Let

$$z_1 = \frac{h}{\Delta x^2}\left(\frac{2\pi m}{M}\right)^2, \quad z_2 = \frac{h}{\Delta y^2}\left(\frac{2\pi n}{N}\right)^2.$$

The evolution of the mode (m, n) of the error is

$$\hat{\varepsilon}_{m,n}^{(k+1)} = \frac{(\gamma z_1)(\gamma z_2)}{(1 + \gamma z_1)(1 + \gamma z_2)}\hat{\varepsilon}_{m,n}^{(k)}$$

The error amplification factor for the (m, n) mode is

$$R_{m,n} = \frac{(\gamma z_1)(\gamma z_2)}{(1 + \gamma z_1)(1 + \gamma z_2)}, \quad |R_{m,n}| < 1, \quad |R_{m,n}| \overset{z_1, z_2 \to \infty}{\longrightarrow} 1.$$

Therefore we expect that more iterations will be required for stiff problems. The AMF with correction will work well for mildly stiff problems. It will work well for stiff problems only when the solution is smooth, and the high order modes are approaching zero. Similar conclusions are drawn for the three-way splitting of a three dimensional diffusion problem

$$R_{m,n.k} = \frac{(\gamma z_1)(\gamma z_2) + (\gamma z_1)(\gamma z_3) + (\gamma z_2)(\gamma z_3) + (\gamma z_1)(\gamma z_2)(\gamma z_3)}{(1 + \gamma z_1)(1 + \gamma z_2)(1 + \gamma z_3)}.$$

Remark 5. *The accuracy analysis in Calvo and Gerisch's paper [134] considers the non-stiff case. For very stiff systems the correction term* (6.11) *is*

$$h\,\left(\mathbf{I} - h\,\gamma\,\widetilde{\mathbf{L}}\right)^{-1}\,\left(h\,\gamma\,\mathbf{L} - h\,\gamma\,\widetilde{\mathbf{L}}\right)\,y_n = \mathcal{O}(h)$$

and the remaining error term is $\mathcal{O}(h^2)$, therefore the corrected solution is first order.

6.3.2 Propagation of linear system errors

The computation of stage values via (6.12) and (6.13) propagates the linear system errors from one stage to another. To account for the total error consider the methods (6.8) and (6.12) and let

$$\delta Y_i = \widetilde{Y}_i - Y_i. \tag{6.17}$$

We assume that these errors are small. The exact stage equations (6.8a) read

$$(\mathbf{I} - h\,\gamma\,\mathbf{L})\; Y_i \;=\; y_n + h\sum_{j=1}^{i-1} a_{i,j}\, f(Y_j) + h\sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\mathbf{L}\,Y_j.$$

The AMF stage equations are solved inexactly and read

$$(\mathbf{I} - h\,\gamma\,\mathbf{L})\;\left(\widetilde{Y}_i - \varepsilon_i\right) \;=\; y_n + h\sum_{j=1}^{i-1} a_{i,j}\, f(\widetilde{Y}_j) + h\sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\mathbf{L}\,\widetilde{Y}_j,$$

where ε_i is the error due to the simplified Newton approximation of the system solution. Using (6.17) we express this in terms of the solution of the exact stages

$$\begin{aligned}
(\mathbf{I} - h\,\gamma\,\mathbf{L})\;(Y_i + \delta Y_i - \varepsilon_i) \;&=\; y_n + h\sum_{j=1}^{i-1} a_{i,j}\, f(Y_j + \delta Y_j) \\
&\quad + h\sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\mathbf{L}\,(Y_j + \delta Y_j) \\
&=\; y_n + h\sum_{j=1}^{i-1} a_{i,j}\, f(Y_j) + h\sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\mathbf{L}\,Y_j \\
&\quad + h\sum_{j=1}^{i-1} a_{i,j}\, f'_j\,\delta Y_j + h\sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\mathbf{L}\,\delta Y_j,
\end{aligned}$$

where we use the mean value theorem

$$f(Y_j + \delta Y_j) - f(Y_j) = f'_j \cdot \delta Y_j\,, \quad f'_j = \int_0^1 f_y(Y_j + s\,\delta Y_j)\,ds.$$

The nonstiff/moderately stiff assumption about the nonlinear terms f implies that these average Jacobians are of moderate size,

$$\|f'_j\| = \mathcal{O}(1)\,, \quad \forall\, j\,. \tag{6.18}$$

After subtracting the exact stage equations we are left with the error relation

$$\delta Y = \left(\mathbf{I} - h\,\widehat{\mathbf{A}} \otimes \mathbf{L} - h\,\mathbf{A} \odot F'\,\right)^{-1}\,\left(\mathbf{I} - h\,\widehat{\mathbf{\Gamma}} \otimes \mathbf{L}\right)\,\varepsilon \tag{6.19}$$

where $\widehat{\mathbf{\Gamma}}$=diag($\widehat{\mathbf{A}}$) and $(\mathbf{A} \odot F')_{i,j} = a_{i,j} f'_j$. For nonstiff or moderately stiff problems $\|h\mathbf{L}\| = \mathcal{O}(h)$, $\|hF'\| = \mathcal{O}(h)$, and for small step sizes we have

$$\|\delta Y\| = (1 + \mathcal{O}(h))\ \|\varepsilon\|.$$

For highly stiff systems $\|h\mathbf{L}\| \to \infty$ and we have

$$\delta Y = \left(\widehat{\mathbf{A}} \otimes (\mathbf{L}/\|\mathbf{L}\|)\right)^{-1} \left(\widehat{\mathbf{\Gamma}} \otimes (\mathbf{L}/\|\mathbf{L}\|)\right) \varepsilon$$

therefore

$$\|\delta Y\| = \mathcal{O}(1)\, \|\varepsilon\|.$$

In both the nonstiff and the stiff cases the stage error is of the size of the linear system solution error.

From the exact step equation the error in the solution is

$$\delta y_{n+1} \quad = \quad h \sum_{j=1}^{s} b_j\, f'_j\, \delta Y_j + h \sum_{j=1}^{s} \widehat{b}_j \mathbf{L}\, \delta Y_j.$$

Non-stiff or moderately stiff problems. In the non-stiff or moderately stiff case where $\|\mathbf{L}\| = \mathcal{O}(1)$ we have

$$\varepsilon \sim \mathcal{O}\left(h^{2k+2}\right) \quad \Rightarrow \quad \delta Y \sim \mathcal{O}\left(h^{2k+2}\right) \quad \Rightarrow \quad \|\widetilde{y}_{n+1} - y_{n+1}\| \sim \mathcal{O}\left(h^{2k+3}\right).$$

For $k = 0$ and $k = 1$ correction iterations we have the following results.

Theorem 6.3.1. *If a LIRK method of order higher than* 2 *is applied to a nonstiff or moderately stiff case of* (6.1) *with the AMF technique according to* (6.12)*, then the order of the method will reduce to second order.*

Theorem 6.3.2. *If a LIRK method of order* 3 *or* 4 *is applied to a nonstiff or moderately stiff case of* (6.1) *with the AMF technique according to* (6.12)*, and one correction iteration* (6.14) *is applied to each stage value* Y_i*, the order of the method is the same as that of the original method.*

Remark 6. *Taking more iterations in the correction procedure may further reduce the linear system solution errors.*

Remark 7. *Correspondingly two iterations of the correction procedure are needed for a LIRK method of order* 5 *or* 6 *since* $k = 2$ *yields an error in the solution of magnitude* $\mathcal{O}\left(h^7\right)$*. The idea can be extended to arbitrarily high order methods.*

Very stiff problems. In the highly stiff case a more complex analysis based on (6.19) is called for since $\|h\mathbf{L}\|$ can be large and $\|h\mathbf{L}\cdot\delta Y_j\|$ can also become large. We consider LIRK methods with a stiffly accurate implicit component $\widehat{b}_i = \widehat{a}_{s,i}$. We have that

$$\begin{aligned} Y_s &= y_n + h\sum_{j=1}^{i-1} a_{s,j}\, f(Y_j) + h\sum_{j=1}^{s} \widehat{a}_{s,j}\,\mathbf{L}\,Y_j, \\ y_{n+1} &= y_n + h\sum_{j=1}^{s} b_j\, f(Y_j) + h\sum_{j=1}^{s} \widehat{b}_j\mathbf{L}\,Y_j \\ &= Y_s + h\sum_{j=1}^{s} (b_j - a_{s,j})\, f(Y_j). \end{aligned}$$

The corresponding error equation

$$\delta y_{n+1} = \delta Y_s + h\sum_{j=1}^{s} (b_j - a_{s,j})\, f_j'\,\delta Y_j$$

and the non-stiff condition (6.18) reveal that the error in the solution is of the same size as the error in the linear solvers

$$\|\widetilde{y}_{n+1} - y_{n+1}\| \sim \|\delta Y\| \sim \|\varepsilon\|.$$

6.3.3 Stability considerations

Following Calvo and Gerisch [134] we perform stability analysis using the following scalar test problem:

$$hf(y) = iwy, \quad h\mathbf{L} = z = z_1 + z_2, \quad h\widetilde{\mathbf{L}} = z_1 + z_2 - \gamma z_1 z_2. \tag{6.20}$$

The LIRK method (6.8) applied to the test problem (6.20) gives:

$$\begin{aligned} y_{n+1} &= R(z, iw)\, y_n, \\ R(z, iw) &= 1 + (iw\, b + z\widehat{b})^T \left(\mathbf{I} - z\widehat{A} - iwA\right)^{-1} \mathbf{1}, \\ R(\infty, iw) &= 1 - \widehat{b}^T\, \widehat{A}^{-1}\, \mathbf{1}. \end{aligned}$$

The LIRK+AMF method (6.12) applied to the test problem (6.20) gives:

$$\begin{aligned} (1-\gamma z_1)(1-\gamma z_2)\,\widetilde{Y}_i &= y_n + iw\sum_{j=1}^{i-1} a_{i,j}\,\widetilde{Y}_j + (z_1+z_2)\sum_{j=1}^{i-1} \widehat{a}_{i,j}\,\widetilde{Y}_j, \\ (1+\gamma^2 z_1 z_2)\,\widetilde{Y}_i &= y_n + iw\sum_{j=1}^{i-1} a_{i,j}\,\widetilde{Y}_j + (z_1+z_2)\sum_{j=1}^{i} \widehat{a}_{i,j}\,\widetilde{Y}_j, \end{aligned}$$

and therefore

$$\begin{aligned}
\widetilde{Y}_i &= \Big((1+\gamma^2 z_1 z_2)\mathbf{I} - z\widehat{A} - iwA\Big)^{-1}\,\mathbf{1}\,y_n, \\
\widetilde{y}_{n+1} &= y_n + iw\sum_{j=1}^{s} b_j\,\widetilde{Y}_j + (z_1+z_2)\sum_{j=1}^{s}\widehat{b}_j\,\widetilde{Y}_j.
\end{aligned}$$

Consequently, for very stiff linear components the overall scheme is weakly stable:

$$\begin{aligned}
y_{n+1} &= R(z_1,z_2,iw)\,y_n, \\
R(z_1,z_2,iw) &= 1 + (iw\,b + z\widehat{b})^T\,\Big((1+\gamma^2 z_1 z_2)\mathbf{I} - z\widehat{A} - iwA\Big)^{-1}\,\mathbf{1}, \\
R(\infty,\infty,iw) &= 1.
\end{aligned}$$

For the scheme with one refinement step we have:

$$\begin{aligned}
(1-\gamma z_1)\,(1-\gamma z_2)\;Y_i^{(1)} &= 1 + iw\sum_{j=1}^{i-1} a_{i,j}\,\widetilde{Y}_j + z\sum_{j=1}^{i-1}\widehat{a}_{i,j}\;\widetilde{Y}_j, \\
\big(1+\gamma^2 z_1 z_2\big)\;Y_i^{(1)} &= 1 + iw\sum_{j=1}^{i-1} a_{i,j}\,\widetilde{Y}_j + z\sum_{j=1}^{i}\widehat{a}_{i,j}\;\widetilde{Y}_j, \\
(1-\gamma z_1)\,(1-\gamma z_2)\;\widetilde{Y}_i &= \big(1-\gamma\,(z_1+z_2) + 2\,\gamma^2 z_1 z_2\big)\;Y_i^{(1)},
\end{aligned}$$

and therefore

$$\begin{aligned}
\tau &= \frac{1-\gamma\,(z_1+z_2)+2\,\gamma^2 z_1 z_2}{(1-\gamma z_1)\,(1-\gamma z_2)\,(1+\gamma^2 z_1 z_2)}, \\
\widetilde{Y} &= \Big(\tau^{-1}\mathbf{I} - iwA - (z_1+z_2)\,\widehat{A}\Big)^{-1}\,\mathbf{1}\,y_n, \\
\widetilde{y}_{n+1} &= y_n + \Big(iwb + z\widehat{b}\Big)^T\cdot\Big(\tau^{-1}\mathbf{I} - iwA - (z_1+z_2)\,\widehat{A}\Big)^{-1}\,\mathbf{1}\,y_n.
\end{aligned}$$

The corresponding stability function is

$$\begin{aligned}
y_{n+1} &= R(z_1,z_2,iw)\,y_n, \\
R(z_1,z_2,iw) &= 1 + \Big(iwb + z\widehat{b}\Big)^T\cdot\Big(\tau^{-1}\mathbf{I} - iwA - z\,\widehat{A}\Big)^{-1}\,\mathbf{1}, \\
R(\infty,\infty,iw) &= 1.
\end{aligned}$$

We have that for very stiff components the scheme with one level of refinement is weakly stable. The refinement does not improve the overall stability properties of the scheme, only the accuracy. The correction step of Calvo and Gerisch [134] leads to L-stable, first order methods for very stiff problems.

6.4 Numerical experiments

We perform numerical experiments with the following methods:

- **LIRK3(4):** the original LIRK methods of orders three and four, respectively, proposed by Calvo, de Frutos, and Novo [93]. The implicit parts are L-stable and stiffly accurate;
- **LIRK3(4)AMF:** the LIRK methods using AMF as (6.12);
- **LIRK3(4)AMFR1:** the LIRK methods using AMF together with one iteration refinement;
- **LIRK3(4)AMFR2:** the LIRK methods using AMF together with two iterations refinement.

In all the experiments, the error is computed in the relative L_2 norm as follows

$$E = \frac{\|u - u_{\text{ref}}\|_2}{\|u_{\text{ref}}\|_2}, \tag{6.21}$$

where u is the numerical solution at the final time, and u_{ref} is the reference solution at the same point obtained using Matlab's `ode15s` solver with very tight tolerances (AbsTol=RelTol=3×10^{-14}).

6.4.1 An Allen-Cahn type problem

We use the PDE test problem of Allen-Cahn type from [134]:

$$u_t = \Delta u + u - u^3 + f, \tag{6.22}$$

where f is chosen to make the exact solution of the equation be

$$u(t, x, y) = e^t \sin(\pi x) \sin(\pi y).$$

The spatial domain is $(x, y) \in [0, 1] \times [0, 1]$ and the time interval is $t \in [0, 1]$ (units). The initial conditions and Dirichlet boundary conditions are calculated from the exact solution.

The spatial discretization uses second order central finite differences of the Laplacian on a uniform grid of size $M \times M$

$$(x_i, y_i) = \left(\frac{i}{M+1}, \frac{j}{M+1}\right), \quad i, j = 1, \ldots, M. \tag{6.23}$$

In our tests we consider the case $M = 59$. The discrete solution elements $U_{ij}(t) \approx u(t, x_i, y_j)$ are ordered into a vector

$$z = (U_{11}, U_{12}, \ldots, U_{1M}, \ldots, U_{M1}, U_{M2}, \ldots, U_{MM})^T.$$

The resulting ODE system can then be written into the form (6.1) with the discrete diffusion term being the linear part. The largest magnitude of the eigenvalues of the Jacobian for the diffusion term is approximately 2.88E4. The discrete Laplacian operator $\mathbf{L} = \mathbf{L}_x + \mathbf{L}_y$ is split into two matrices corresponding to derivatives along x and y directions, respectively:

$$\mathbf{L}_x = \mathbf{D}_M \otimes \mathbf{I}_M,\ \mathbf{L}_y = \mathbf{I}_M \otimes \mathbf{D}_M,\ \mathbf{D}_M = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}, \tag{6.24}$$

where the symbol $\otimes$ denotes the tensor product and $\mathbf{I}_M$ is an identity matrix of dimension $M \times M$.

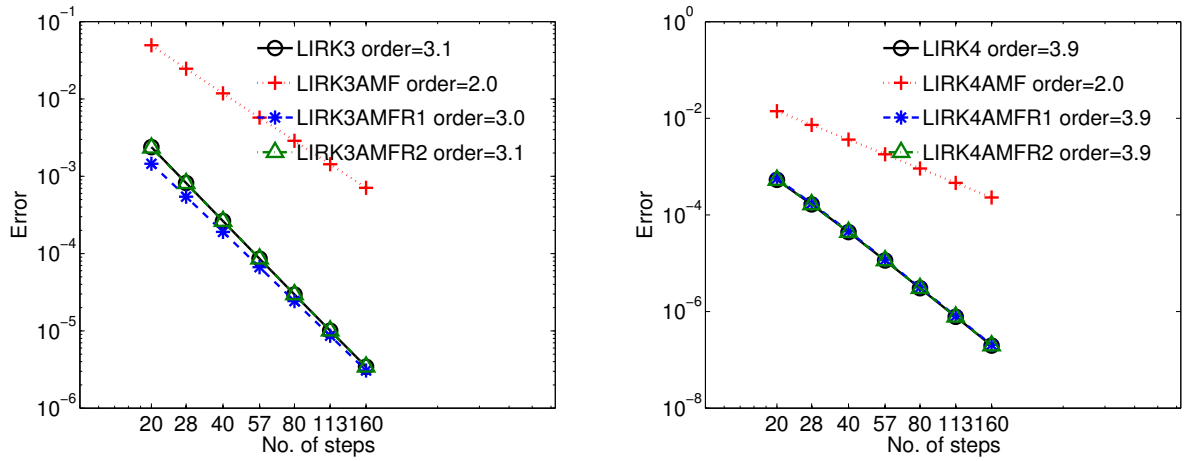
Figure 6.1(a) plots the convergence results for all the methods tested. As expected, both LIRK3AMF and LIRK4AMF show second order, and give less accurate results for the same time step than the underlying LIRK methods. All the LIRK methods with AMF and refinement perform equally well as the original LIRK methods; LIRK3AMFR1 produces slightly better results, and the full order of the underlying LIRK methods has been recovered.

Figure 6.1(b) shows the corresponding work-precision diagrams. LIRK methods with AMF are not very competitive in terms of efficiency. One refinement iteration improves LIRK3AMF and LIRK4AMF significantly. LIRK3AMFR1 and LIRK4AMFR1 are clearly the most efficient methods among the methods of the same order. A second iteration does not improve accuracy, but spends compute time, and makes LIRK3(4)AMFR2 schemes slightly less efficient. Calvo and Gerisch's approach [134] can achieve second order with AMF, and may only be attractive for low accuracy requirements. The approach with stage refinement proposed herein is competitive at all accuracy levels due to the recovery of the full order and the addition of the relatively cheap refinement procedure. It should be noted that the separation of $\mathbf{L}$ into $\mathbf{L}_x$ and $\mathbf{L}_y$ in (6.24) allows for a reduction of the large linear systems with $2M$ small ones of dimension $M \times M$, which can lead to considerable savings in the computational cost. The trick is not used with this example simply for the sake of enabling a fair comparison with the results in [134].

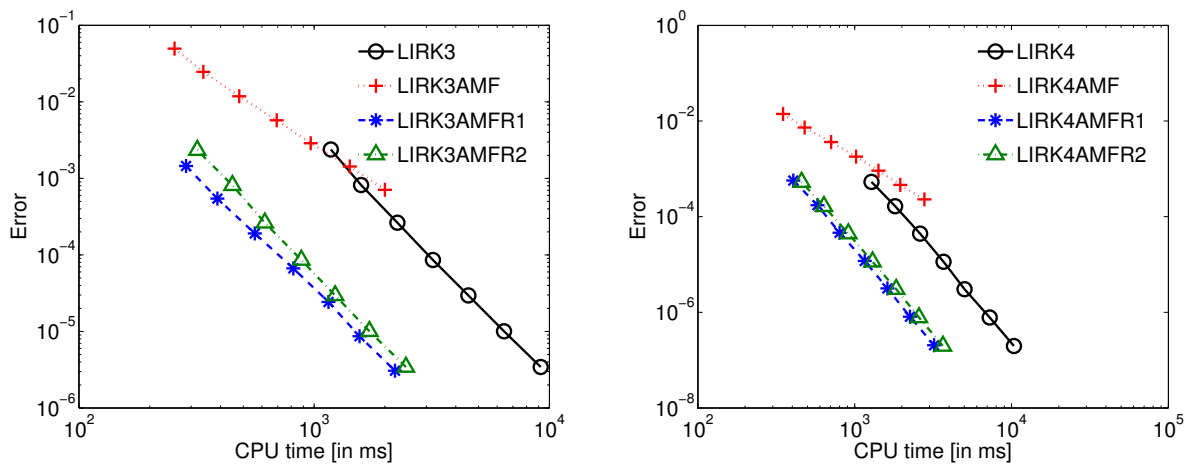
6.4.2 Brusselator problem

We next consider the two-dimensional Brusselator reaction-diffusion equation [51, Sec. IV.10]

$$u_t = 1 + u^2 v - (B+1)\, u + \alpha \Delta u, \tag{6.25a}$$
$$v_t = B\, u - u^2 v + \alpha \Delta v, \tag{6.25b}$$



(a) Temporal errors vs. number of steps



(b) Temporal errors vs. CPU time

Figure 6.1: Results for the 2D Allen-Cahn type problem (6.22).

where $(x, y) \in [0, 1]^2$, $t \in [0, 1.5]$, with the Neumann boundary conditions

$$\frac{\partial u}{\partial \mathbf{n}} = 0, \quad \frac{\partial v}{\partial \mathbf{n}} = 0.$$

The problem is discretized with a second order central finite difference scheme on a uniform mesh (6.23). The stiffness of this problem increases with the value of α and number of grid points M. We test LIRK+AMF methods with or without iterative corrections on two different cases.

Case 1. A nonstiff stiff case described by $\alpha = 0.001$, $B = 3$ and the initial conditions

$$u(x, y, 0) = 0.5 + y, \quad v(x, y, 0) = 1 + 5x.$$

with $M = 39$ grid points used in each dimension. This gives an ODE system is of dimension $N = 3,042$.

Case 2. A stiff case, in which the settings follow [51, Sec. IV.10] where $\alpha = 0.1$, $B = 3.4$, and the initial conditions are defined as

$$u(x, y, 0) = 22y(1 - y)^{3/2}, \quad v(x, y, 0) = 22x(1 - x)^{3/2}.$$

We choose $M = 127$ for the three-way splitting test and $M = 199$ for the two-way splitting test so that the resulting ODE systems are of size $N = 31,752$ and $N = 79,202$ respectively. The time varying Jacobian for the reaction term used in the three-way splitting test makes the linearization challenging especially when the problem is very stiff. For a large M, e.g. $M = 150$, the error caused by the linearization leads to failure in solving the linear systems with direct methods after a few time steps. So we select a relatively smaller value of M for the three-way splitting test. The details on the splitting setup are given later in this section.

Table 6.1 shows the dominant eigenvalues for these test cases which shed light on the degree of stiffness. Note that the Jacobian matrix for the reaction term has complex eigenvalues which makes this test problem more challenging than the previous one.

After spatial discretization the PDE is turned into a semi-linear ODE system of the form

$$z' = \underbrace{\mathbf{L}_{dif}\, z}_{\text{diffusion}} + \underbrace{\mathbf{L}_{rea}(t)\, z}_{\text{reaction}} + R,$$

where z is the combined vector for the variables u and v, $\mathbf{L}_{dif}$ and $\mathbf{L}_{rea}(t)$ are the Jacobian of the diffusion term and reaction term respectively, and R is the rest of the terms such as boundary treatment. The diffusion term is stiff while the reaction term is nonstiff. We first apply LIRK methods with the diffusion treated implicitly and the other terms explicitly. Next we include the Jacobian of the reaction terms in the linear part and apply a three-way splitting strategy. The LU decompositions are performed per time step using sparse Gaussian elimination in MATLAB.

Table 6.1: The dominant eigenvalues (largest in magnitude) of each component.

Case	L_x	L_y	$L_x + L_y$	$L_{rea}(t_0)$
1 ($M = 39$)	1.28E1	1.28E1	2.56E1	1.05E1
2 ($M = 127$)	6.55E3	6.55E3	1.31E4	2.01E1
3 ($M = 199$)	1.60E4	1.60E4	3.20E4	2.01E1

Two-way splitting. A directional splitting is applied to the diffusion term which is written as the sum of derivatives in the x-direction and y-direction, $\mathbf{L}_{dif} = \mathbf{L}_x + \mathbf{L}_y$. See (6.24) for

the structure of $\mathbf{L}_x$ and L_y. This splitting allows to reduce the linear algebra effort to solving $2M$ tridiagonal systems of dimension M, all of which share the same matrix $\mathbf{I} - h\gamma\,\mathbf{D_M}$, and use reordered right-hand sides.

Figure 6.2(a) shows the convergence plots of different methods for Case 1. Both LIRK3AMF and LIRK4AMF give second order. With one refinement iteration the results become as accurate as those of the original LIRK methods. The largest allowable time steps are almost the same for all methods, implying good stability properties of LIRK methods with AMF. Figure 6.2(b) presents the corresponding work-precision diagrams. It can be seen that LIRK methods with one refinement iteration yield the best efficiency. To achieve the same accuracy level, LIRK3AMFR1 is about 2.2 times faster than LIRK3 and LIRK4AMFR1 is about 1.6 times faster than LIRK4.

Figure 6.3 shows the convergence and work-precision diagrams for different methods for the large scale stiff case with $M = 199$. Generally LIRK methods with AMF and two refinement iterations give more accurate results than those with AMF and one refinement iteration. The refinement works well and recovers the theoretical orders of the corresponding LIRK methods. The errors for methods with two refinement iterations approach the LIRK results at a faster rate than methods with just one refinement iteration. This differs from the results of the first case, but is in line with the theoretical prediction. Another notable advantage of the AMF technique is the gain in term of stability. If no refinement is employed, the maximal time step size can be at least two times larger than that allowed by the original LIRK methods for both third-order and fourth-order schemes. However, the gain disappears or shrinks when the refinement procedure is added.

In the efficiency comparison, LIRK methods with AMF and two refinement are the most effective for solutions more accurate than approximately 10^{-4}. LIRK methods with AMF provide a good compromise between accuracy and speed since they can use a maximal allowable time step size that is at least two times larger than LIRK methods, and run significantly faster than LIRK methods.

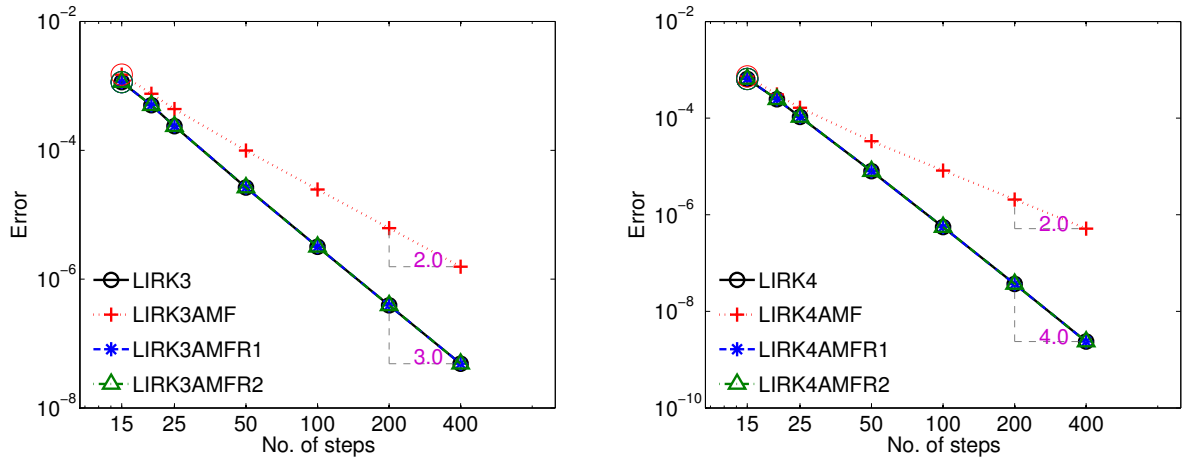
Three-way splitting. We use a three-way splitting of the linear part

$$\mathbf{L} = \mathbf{L}_x + \mathbf{L}_y + \mathbf{L}_{rea}(t)$$

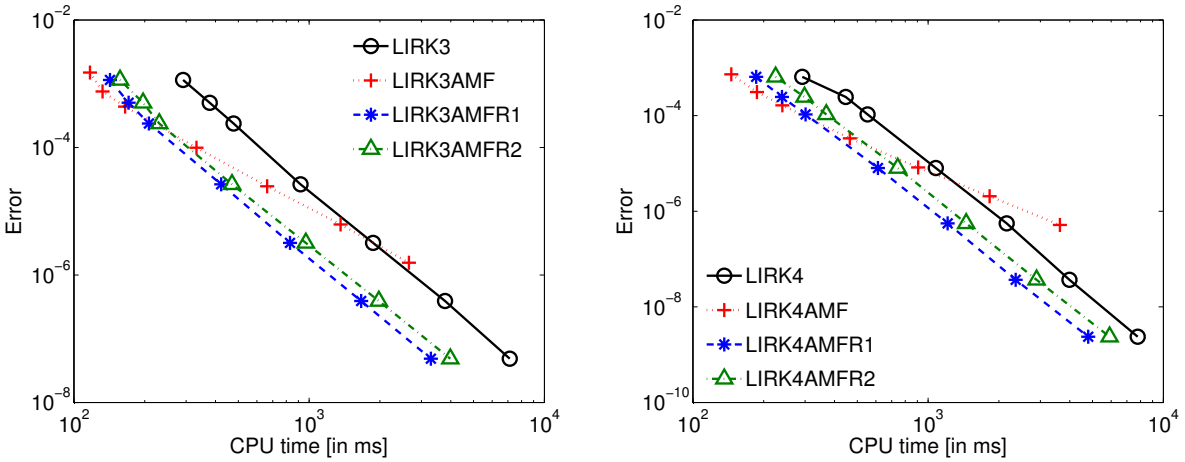
where the Jacobian of the reaction terms is treated implicitly. $\mathbf{L}_{rea}(t)$ contains four blocks, each of which is a diagonal matrix. It is updated at each time step.

The linear system associated with $\mathbf{I} - h\gamma\,\mathbf{L}_{rea}$ can be reduced to M^2 smaller systems of dimension two which can be solved separately at each grid point. We choose to solve the large sparse system directly as an explicit decoupling does not lead to a clear performance gain for a serial implementation.

The results are shown in Figure 6.4, and 6.5. For Case 1, the refinement procedure can successfully improve the order from 2 to the theoretical order. AMF without refinement is



(a) Temporal error vs. number of steps

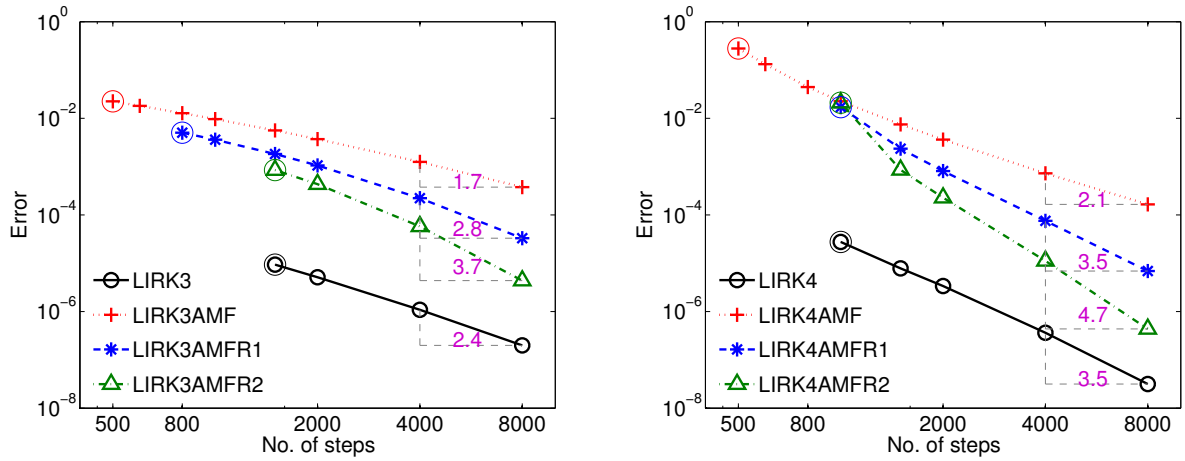


(b) Temporal error vs. CPU time

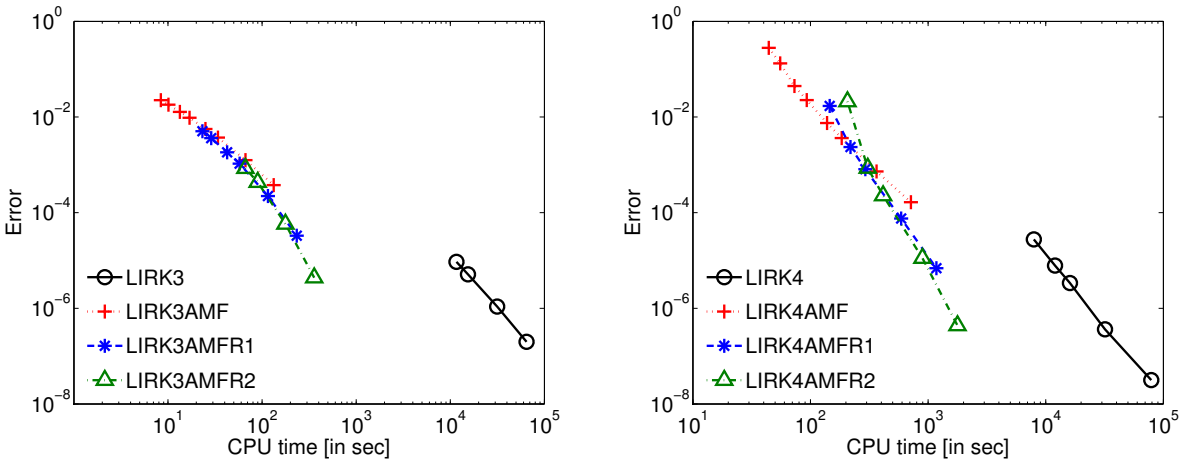
Figure 6.2: Results for the 2D Brusselator system(6.25), Case 1, with $M = 39$. AMF is applied with a two-way splitting of the Jacobian. $15, 20, 25, 50, 100, 200, 400$ equal steps are used for the time integration of the system on the interval $[0, 1]$. The left-most points (highlighted by a circle) on each curve indicates the maximal allowable time steps.

not competitive in terms of efficiency, but AMF with refinement yields some performance gain for third-order schemes and comparable results with LIRK methods for fourth-order schemes. This is due to the fact that the system is relatively small and additional cost is brought in to solve the linear system associated with $\mathbf{L}_{rea}(t)$.

For the stiff case, the Jacobian for the implicit part $\mathbf{L}$ makes the linear system difficult to solve with direct methods. One LU decomposition of the system may take over 5000 seconds. To improve the performance of LIRK methods, we make use of the reordering algorithm `symamd` in MATLAB before solving the linear systems. The reordering can also help reduce the



(a) Temporal error vs. number of steps

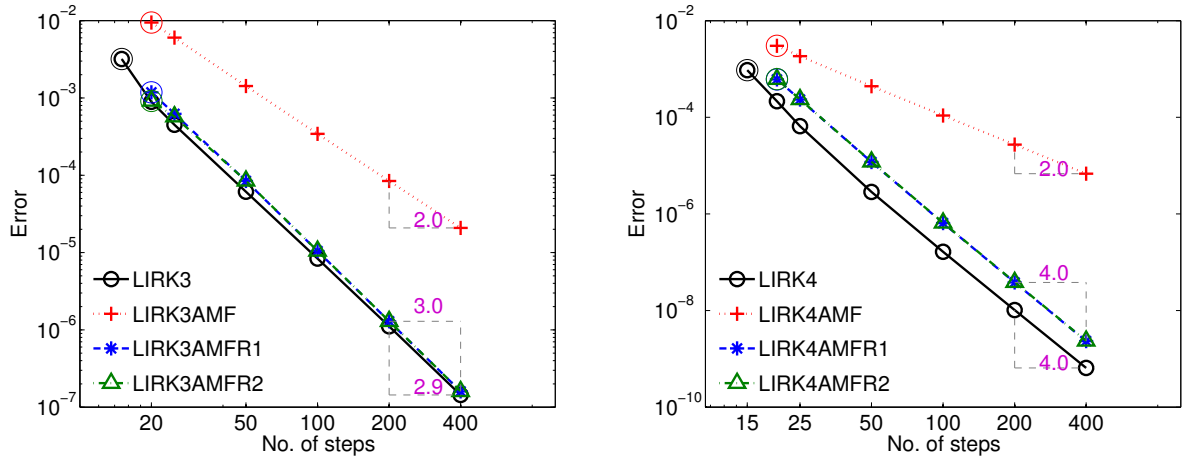


(b) Temporal error vs CPU time

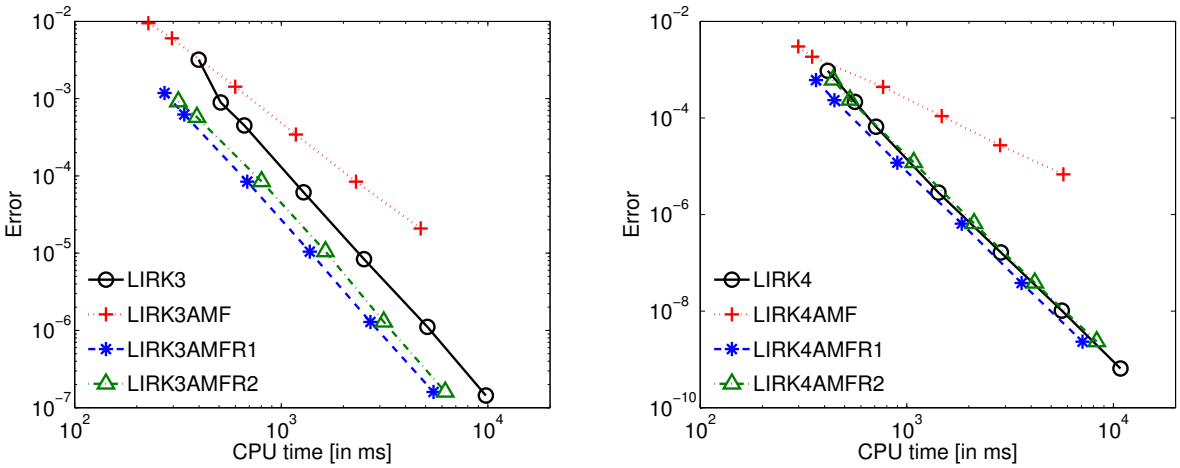
Figure 6.3: Results for the 2D Brusselator system (6.25), Case 2, with $M = 199$. AMF is applied with a two-way splitting of the Jacobian. 400, 500, 600, 800, 1000, 1500, 2000, 4000, 8000 equal steps are used for the time integration of the system on the interval $[0, 1]$. The left-most points (highlighted by a circle) on each curve indicate the maximal allowable time steps.

bandwidth of the sparse matrices as similar to purpose of the splitting schemes we used.

The convergence orders for the stiff case are very close to the two-way splitting test results. In terms of efficiency, there is still considerable performance gain for AMF with refinement, especially for AMF with two refinement iterations. The savings in CPU time may mainly come from reducing the big system into multiple small systems, which is another advantage of the application of AMF.



(a) Temporal error vs. number of steps

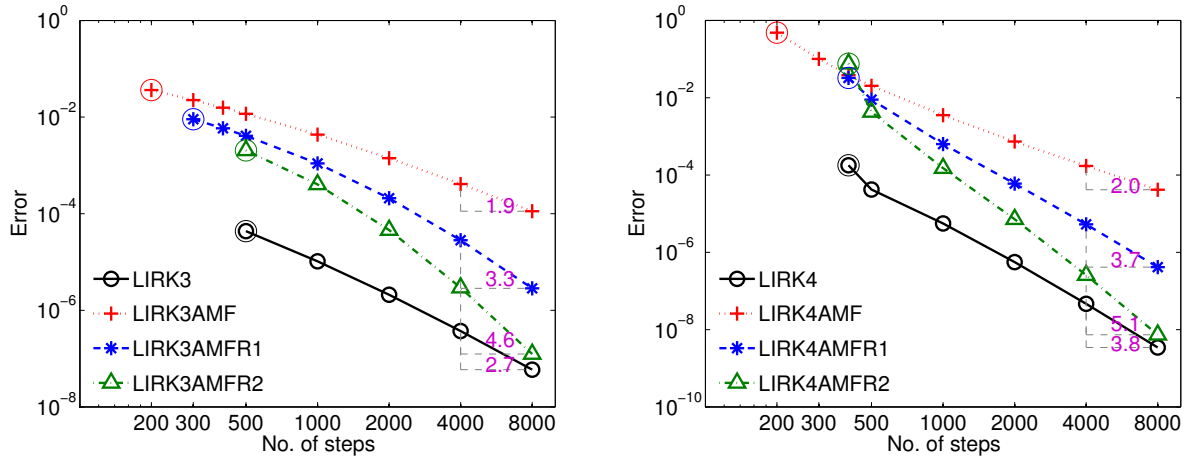


(b) Temporal error vs. CPU time

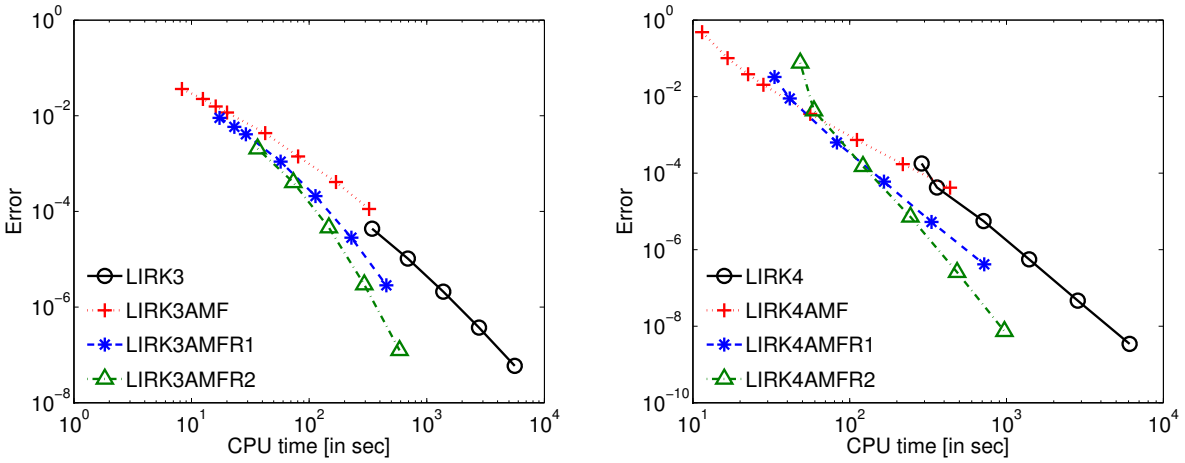
Figure 6.4: Results for the 2D Brusselator system (6.25), Case 1, with $M = 39$. AMF is applied with a *three-way* splitting of the Jacobian. 15, 20, 25, 50, 100, 200, 400 equal steps are used for the time integration of the system on the interval $[0, 1]$. The left-most points (highlighted by a circle) on each curve indicates the maximal allowable time steps. The numbers inside the triangle give the convergence order.

6.5 Conclusions

We have applied approximate matrix factorization to high order linearly implicit Runge-Kutta methods for solving semi-linear systems of differential equations. The factorization (splitting) error brought by AMF leads to severe order degradation, especially for high order Runge-Kutta methods. The existing approach to recover second order is based on correction applied to the next step solution [134]. In this work the full order of the underlying methods



(a) Temporal error vs. number of steps



(b) Temporal error vs. CPU time

Figure 6.5: Results for the 2D Brusselator system (6.25), Case 2, with $M = 127$. AMF is applied with a three-way splitting of the Jacobian. 100, 200, 300, 400, 500, 1000, 2000, 4000, 8000 equal steps are used for the time integration of the system on the interval $[0, 1]$. The left-most points (highlighted by a circle) on each curve indicates the maximal allowable time steps.

is recovered by correcting stage values via a refinement procedure based on the idea of simplified Newton iterations.

We have performed error analysis for the linear system solutions with AMF, and investigated how this errors affect the next step solution. In the non-stiff and mildly stiff case the full order of the underlying method can be recovered using a fixed, small number of refinement iterations. In the very stiff case the number of iterations can be large since the convergence can deteriorate with increasing stiffness. A stability analysis reveals that the stage refinement procedure does not improve the overall stability of the LIRK+AMF method. When AMF is used the resulting schemes are only weakly stable for very stiff problems. Consequently, this application of AMF is attractive for mildly stiff problems, but may not work well for very stiff systems.

Numerical experiments on a variety of test problems of different sizes and different degrees of stiffness validate the theoretical findings on the accuracy and stability of high order linearly implicit Runge-Kutta methods when AMF is used. The results also show that the proposed approach can improve the efficiency of high order linearly implicit Runge-Kutta methods significantly and thus is attractive for solving large scale mildly stiff systems such as diffusion-reaction equations. Furthermore, our tests on the three-way splitting demonstrate that our methods can also efficiently deal with problems where stiffness comes from both diffusion and reaction terms. Though we considered LIRK schemes up to order four, the general framework developed herein can be applied to higher order LIRK methods, and could be extended to study the use of AMF with implicit-explicit multistep methods and implicit-explicit general linear methods.

Chapter 7

Summary and future work

Numerical simulations of multiphysics and multiscale systems pose great challenges to both algorithms and software. Partitioned time stepping schemes are essential for providing an efficient and time-accurate solution to these problems. Tools that can efficiently perform sensitivity analysis of large scale simulations are also very important. This study addresses novel partitioned time stepping methods for large scale systems of PDEs and ODEs and high performance software for ODE simulations with sensitivity analysis capabilities.

FATODE is a general purpose software for time integration of nonstiff and stiff ODE systems, and direct and discrete adjoint sensitivity analyses. This library is potentially useful for a variety of applications including uncertainty quantification, inverse problems such as parameter estimation and data assimilation, and optimization of systems governed by ODEs. FATODE is currently being used by other groups for:

- sensitivity analysis of multibody dynamic systems to tackle the optimization of the response of very complex systems such as vehicles [135, 136, 137, 138, 139],
- computation of gradient or Hessian information for optimization-based time-parallel algorithms and data assimilation [140, 141], and for
- educational use for the undergraduate course in mathematics "Applied Mathematical Modeling" [142].

As far as we know, FATODE is the first publicly available general purpose software that implements a *discrete adjoint sensitivity analysis* approach. This gives gradients that are exact, within roundoff error, and therefore are highly suitable for numerical optimization problems. Since FATODE is designed to be easily incorporated into other simulation codes, more applications are expected to benefit from its capabilities in the future without considerable adaptation effort.

Partitioned time integration schemes are becoming increasingly popular for solving multi-physics and multiscale problems, which arise in many application areas such as mechanical and chemical engineering, astrophysics, meteorology and oceanography, and environmental science. We propose a new family of partitioned time integration methods based on high stage order general linear methods. The high stage order guarantees that no coupling conditions are needed to ensure the order of accuracy of the partitioned GLM. We prove that the general linear framework is well suited for the construction of multi-methods (composite methods). We apply the partitioned general linear framework to construct new implicit-explicit (IMEX) DIMSIM pairs with different orders of accuracy ranging from two to five, based on L-stable implicit components, and with the explicit components optimized such as to maximize the constrained stability regions. The stability and convergence analysis indicate that IMEX GLMs are particularly attractive for solving stiff problems, where other multistage methods may suffer from serious order reduction.

High order spatial discretization schemes are gaining popularity in modern PDE solvers. Time discretization schemes that match the high order of the spatial discretization are highly desirable, but usually not available. The high order implicit-explicit methods we constructed demonstrate good stability and high efficiency for the two-dimensional Allen-Cahn and Burgers' equations with finite difference spatial discretizations, and two- and three-dimensional compressible Euler equations with discontinuous Galerkin space discretizations. We conclude from both the theoretical results and numerical results that IMEX GLMs are useful in situations where IMEX Runge-Kutta methods suffer from order reduction; specific examples include stiff systems of singular perturbation type or problems with challenging time-dependent boundary conditions. We are currently exploring the application of the new methods to large aerodynamic simulations.

The approach to construct IMEX schemes via numerical optimization has been extended to another family of promising GLMs, namely, two-step Runge-Kutta methods. Two practical IMEX TSRK methods, of orders three and four respectively, have been constructed. They consist of a stiffly-accurate implicit part and an optimized explicit part. Numerical examples including an advection-reaction system, the Van der Pol equation, and a semi-implicit integration of shallow water equations confirm the theoretical predictions and show competitive the IMEX-TSRK methods are competitive in terms of accuracy and efficiency with classic IMEX schemes of Runge-Kutta and linear multistep types.

The novel schemes we have developed are suitable for large scale systems, especially ones with challenging boundary conditions, source terms, or stiff components. They are highly competitive alternatives to classic IMEX methods.

The success of an IMEX scheme depends on the ability to find a good additive splitting of the system. An appropriate splitting should be determined primarily by considering the stiffness of different parts in the right-hand side of the system. Another aspect to consider is whether the stiff part is linear or can be linearized since this may affect the computational cost of the scheme significantly. In practice, one often has basic knowledge of the system,

e.g. the dynamics of the physics, otherwise a strategy to justify the stiffness of different components is needed. A common methodology for the measurement of stiffness is to look into the eigenvalues of the Jacobian. To serve as a guideline for dealing with the splitting, the approaches we used with IMEX GLMs in all the numerical examples are summarized below:

- Straightforward splitting between different terms. Examples include the advection-reaction system in Section 5.7.1, the Allen-Cahn equation and Burgers' equation in Section 4.4, where the diffusion terms are known to be stiff and thus treated implicitly.
- Use of zero-padding to transform component partitioning into additive partitioning. In the van der Pol equation in Section 3.6.1, only one of the two components is stiff. The right-hand side is split into two zero-padded vectors corresponding to the two components respectively.
- Splitting involving linearization. To linearize a nonlinear stiff part, some terms on the right-hand side of the system may have to be modified. For example, in the compressible Euler equations in Section 3.6.2 and 4.4.3, a linear operator is built from the advective terms to account for acoustic waves which has the fastest time scale. It is also possible that the whole right-hand side is reformulated completely by techniques like the Jacobian linearization used for the splitting of the shallow water equations in Section 5.7.3.

Other approaches may also be feasible for various problems, but are not further discussed here.

Moreover, the partitioned GLM framework opens up many opportunities to find new methods with favorable properties. This work opens up new possibilities for constructing useful time integration methods. An important ingredient of efficient implementations for large scale simulations is step size adaptivity. In order to realize the full potential of IMEX GLMs we will focus on the construction of schemes in this family with error estimation and adaptive time steps.

Linearly implicit Runge-Kutta (LIRK) methods with approximate matrix factorization (AMF) can solve large ODE systems resulting from semidiscretization of PDEs that have a stiff linear part, e.g., reaction-diffusion systems. However, the use of AMF usually leads to severe order reduction and loss of accuracy. Consequently AMF is attractive only for low order time integration methods and low-to-medium accuracy requirements. By using an inexpensive correction procedure similar to the inexact Newton's iteration on each stage computation, the accuracy of high order LIRK methods is preserved. Error analysis and error propagation are studied. The effectiveness of our approach for both two-way and three-way splittings have been demonstrated on several reaction-diffusion type problems of different degrees of stiffness. The application of AMF to IMEX GLMs deserves further investigation.

Bibliography

[1] R. Rosner. The opportunities and challenges of exas-cale computing. *Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee on Exascale Computing, Office of Science, U.S. Department of Energy*, 2010.

[2] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, and et al. Multiphysics simulations: Challenges and opportunities. *International Journal of High Performance Computing Applications*, 2012.

[3] L. Gebhardt, D. Fokin, Th. Lutz, and S. Wagner. An implicit-explicit Dirichlet-based field panel method for transonic aircraft design. In *AIAA Applied Aerodynamics Conference*, June 2002.

[4] S. J. Ruuth. Implicit-explicit methods for reaction-diffusion problems in pattern formation. *Journal of Mathematical Biology*, 34:148–176, 1995.

[5] J.G. Verwer, J.G. Blom, and W. Hundsdorfer. An implicit-explicit approach for atmospheric transport-chemistry problems. *Applied Numerical Mathematics*, 20:191–209, 1996.

[6] F.X. Giraldo, J.B. Perot, and P.F. Fischer. A spectral element semi-lagrangian (SESL) method for the spherical shallow water equations. *Journal of Computational Physics*, 190(2):623–650, September 2003.

[7] D. Estep, V. Ginting, D. Ropp, J. Shadid, and S. Tavener. An a posteriorica priori analysis of multiscale operator splitting. *SIAM Journal on Numerical Analysis*, 46(3):1116–1146, 2008.

[8] F. Giraldo and M. Restelli. High-order semi-implicit time-integrators for a triangular discontinuous Galerkin oceanic shallow water model. *International Journal for Numerical Methods in Fluids*, 63(9):1077–1102, 2009.

[9] F. Giraldo, M. Restelli, and M. Läuter. Semi-implicit formulations of the Navier–Stokes equations: Application to nonhydrostatic atmospheric modeling. *SIAM Journal on Scientific Computing*, 32(6):3394–3425, 2010.

[10] D.R. Durran and P.N. Blossey. Implicit-explicit multistep methods for fast-wave-slow-wave problems. *Monthly Weather Review*, 140:1307–1325, April 2012.

[11] E.M. Constantinescu and A. Sandu. Multirate timestepping methods for hyperbolic conservation laws. *Journal on Scientific Computing*, 33(3):239–278, 2007.

[12] E.M. Constantinescu, A. Sandu, and G.R. Carmichael. Modeling atmospheric chemistry and transport with dynamic adaptive resolution. *Computational Geosciences*, 12(2):133–151, 2008.

[13] A. Sandu and E.M. Constantinescu. Multirate Adams methods for hyperbolic equations. *Journal of Scientific Computing*, 38(2):229–249, 2009.

[14] T.J.R. Hughes and W.K. Liu. Implicit-explicit finite elements in transient analysis: Implementation and numerical examples. *Journal of Applied Mechanics*, 45:375, 1978.

[15] T.J.R. Hughes and W.K. Liu. Implicit-explicit finite elements in transient analysis: Stability theory. *Journal of Applied Mechanics*, 45:371, 1978.

[16] A. Kanevsky, M.H. Carpenter, D. Gottlieb, and J.S. Hesthaven. Application of IMEX high order RK methods to discontinuous Galerkin schemes. *Journal of Computational Physics*, 225:1753–1781, 2007.

[17] J.R. Leis and M.A. Kramer. ODESSA - an ordinary differential equation solver with explicit simultaneous sensitivity analysis. *ACM Transactions on Mathematical Software*, 14(1):61–75, 1986.

[18] R. Serban and A.C. Hindmarsh. CVODES, the sensitivity-enabled ODE solver in SUNDIALS. Technical Report UCRL-PROC-210300, Lawrence Livermore National Laboratory, 2003.

[19] M. Alexe and A. Sandu. Forward and adjoint sensitivity analysis with continuous explicit Runge-Kutta schemes. *Applied Mathematics and Computation*, 208(2):328–34, 2009.

[20] A. Sandu and D. Daescu and G.R. Carmichael. Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: II – Numerical validation and applications. *Atmospheric Environment*, 37:5097–5114, 2003.

[21] A. Sandu, D. Daescu, and G.R. Carmichael. Direct and Adjoint Sensitivity Analysis of Chemical Kinetic Systems with KPP: I – Theory and Software Tools. *Atmospheric Environment*, 37:5083–5096, 2003.

[22] J.C. Butcher and Z. Jackiewicz. Diagonally implicit general linear methods for ordinary differential equations. *BIT*, 33(3):452–472, 1993.

[23] J.C. Butcher. Diagonally-implicit multi-stage integration methods. *Applied Numerical Mathematics*, 11(5):347 – 363, 1993.

[24] Z. Jackiewicz. *General Linear Methods for Ordinary Differential Equations*. Wiley, Hoboken, New Jersey, 2009.

[25] M. Crouzeix. Une methode multipas implicite-explicite pour l'approximation des equations d'evolution parabolique. *Numerische Mathematik*, 35:257–276, 1980.

[26] C.A. Kennedy and M.H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44(1-2):139–181, Jan 2003.

[27] E.M. Constantinescu and A. Sandu. Extrapolated implicit-explicit time stepping. *SIAM Journal on Scientific Computing*, 31(6):4452–4477, 2010.

[28] F. Filbet and S. Jin. A class of asymptotic-preserving schemes for kinetic equations and related problems with stiff sources. *Journal of Computational Physics*, 229(20):7625 – 7648, 2010.

[29] J. Frank, W. Hundsdorfer, and J.G. Verwer. On the stability of IMEX linear multistep methods. *Applied Numerical Mathematics*, 25:193–205, 1997.

[30] W. Hundsdorfer and S.J. Ruuth. IMEX extensions of linear multistep methods with general monotonicity and boundedness properties. *Journal of Computational Physics*, 225(2):2016–2042, 2007.

[31] U. Ascher, S. Ruuth, and B. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):pp. 797–823, 1995.

[32] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25:151–167, 1997.

[33] S. Boscarino. Error analysis of IMEX Runge–Kutta methods derived from differential-algebraic systems. *SIAM Journal on Numerical Analysis*, 45(4):1600–1621, 2007.

[34] L. Pareschi and G. Russo. Implicit-explicit Runge-Kutta schemes for stiff systems of differential equations. In *Recent trends in numerical analysis*, pages 269–288. Nova Science Publishers, Inc., 2000.

[35] L. Pareschi and G. Russo. Implicit-explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxation. *Journal of Scientific Computation*, 2004.

[36] J.G. Verwer, B.P. Sommeijer, and W. Hundsdorfer. RKC time-stepping for advection–diffusion–reaction problems. *Journal of Computational Physics*, 201(1):61–79, 2004.

[37] M.P. Calvo and A. Gerisch. Linearly implicit Runge-Kutta methods and approximate matrix factorization. *Applied Numerical Mathematics*, 53(2-4):183–200, 2005.

[38] S. González-Pinto, D. Hernández-Abreu, and S. Pérez-Rodríguez. Rosenbrock-type methods with inexact AMF for the time integration of advection-diffusion-reaction PDEs. *Journal of Computational and Applied Mathematics*, 262:304–321, may 2014.

[39] S. Beck, S. Gonzlez-Pinto, S. Prez-Rodrguez, and R. Weiner. A comparison of AMF- and Krylov-methods in Matlab for large stiff ODE systems. *Journal of Computational and Applied Mathematics*, 262(0):292 – 303, 2014. Selected Papers from NUMDIFF-13.

[40] H. Zhang and A. Sandu. FATODE: a library for forward, adjoint and tangent linear integration of stiff systems. *SIAM Journal on Scientific Computing (in publish)*, 2014.

[41] H. Zhang and A. Sandu. FATODE: a library for forward, adjoint and tangent linear integration of stiff systems. In *Proceedings of Spring Simulation Multi-conference (SpringSim) 2011*, volume High Performance Computing Symposium, pages 143–150, 2011.

[42] E. Zharovsky and A. Sandu. A class of IMEX two-step Runge-Kutta methods. Computer Science technical report CS TR-12-08, Department of Computer Science, Virginia Tech, http://eprints.cs.vt.edu/archive/00001183, February 2012.

[43] H. Zhang and A. Sandu. Implicit-explicit DIMSIM time stepping algorithms. arXiv:1302.2689.

[44] Hong Zhang and Adrian Sandu. A second-order diagonally-implicit-explicit multi-stage integration method. In *Proceedings of the International Conference on Computational Science ICCS 2012*, pages 152–159, April 2012.

[45] S. Blaise and A. St-Cyr. A dynamic hp-adaptive discontinuous Galerkin method for shallow-water flows on the sphere with application to a global tsunami simulation. *Monthly Weather Review*, 140:978–996, 2012.

[46] C. Geuzaine and J. Remacle. GMSH: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[47] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, 1993.

[48] H. Zhang and A. Sandu. FATODE: A Library for Forward, Adjoint, and Tangent Linear Integration of ODEs. Technical Report TR-11-25, Computer Science, Virginia Tech, URL: http://eprints.cs.vt.edu/archive/00001170, 2011.

[49] H. Zhang and A. Sandu. FATODE: User's Guide. URL: http://people.cs.vt.edu/~asandu/Software/FATODE/downloads/FATODE_user_guide.pdf, 2011.

[50] H. Zhang and A. Sandu. FATODE website: Forward, Adjoint, and Tangent Linear Integration of ODEs. URL: http://people.cs.vt.edu/~asandu/Software/FATODE/index.html, 2011.

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

[52] M.W. Gery, G.Z. Whitten, J.P. Killus, and M.C. Dodge. A photochemical kinetics mechanism for urban and regional scale computer modeling. *Journal of Geophysical Research*, 94(D10):12925–12956, 1989.

[53] W.P.L. Carter. A detailed mechanism for the gas-phase atmospheric reactions of organic compounds. *Atmospheric Environment*, 24:481–518, 1990.

[54] P. Eller, K. Singh, A. Sandu, K. Bowman, D. K. Henze, and M. Lee. Implementation and evaluation of an array of chemical solvers in the Global Chemical Transport Model GEOS-Chem. *Geoscientific Model Development*, 2:89–96, 2009.

[55] Verwer J.G, E.J. Spee, J.G. Blom, and W. Hunsdorfer. A second order Rosenbrock method applied to photochemical dispersion problems. *SIAM Journal on Scientific Computing*, 20:1456–1480, 1999.

[56] A. Sandu, J.G. Verwer, J.G. Blom, E.J. Spee, G.R. Carmichael, and F.A. Potra. Benchmarking stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock methods. *Atmospheric Environment*, 31:3459–3472, 1997.

[57] A. Sandu. On the properties of Runge Kutta discrete adjoints. In *ICCS 2006, IV,LNCS 3994*, pages 550–557, Berlin Heidelberg, 2006. Springer-Verlag.

[58] A. Sandu. Solution of inverse ODE problems using discrete adjoints. *Large Scale Inverse Problems and Quantification of Uncertainty*, pages 345–364, 2010.

[59] Z. Sirkes and E. Tziperman. Finite difference of adjoint or adjoint of finite difference. *Monthly Weather Review*, 125(12):3373–3378, 1997.

[60] H. M. Bücker, G. F. Corliss, P. D. Hovland, U. Naumann, and B. Norris. *Automatic Differentiation: Applications, Theory, and Tools*. Lecture Notes in Computational Science and Engineering. Springer, New York, 2005.

[61] H. Zhang and A. Sandu. Supplementary material for "FATODE: a library for forward, adjoint and tangent linear integration of stiff systems", 2013.

[62] A.M. Dunker. The decoupled direct method for calculating sensitivity coefficients in chemical kinetics. *Journal of Chemical Physics*, 81(5):2385–2393, 1984.

[63] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, Jack J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen. *LAPACK Users' guide (third ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[64] T.A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30:196–199, June 2004.

[65] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert, X.S. Li, and J.W.H. Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, 1999.

[66] A. Sandu and P. Miehe. Forward, Tangent Linear,and Adjoint Runge Kutta Methods in KPP–2.2 for Efficient Chemical Kinetic Simulations. *International Journal of Computer Mathematics*, 2008.

[67] R. Giering and T. Kaminski. Recipes for adjoint code construction. *ACM Transactions on Mathematical Software*, 24:437–474, December 1998.

[68] W. Hager. Runge Kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik*, 87(2):247–282, 2000.

[69] A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, and C.S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, 2005.

[70] D. D. Houghton and A. Kasahara. Nonlinear shallow fluid flow over an isolated ridge. *Communications on Pure and Applied Mathematics*, 21(1):1–23, 1968.

[71] K. Radhakrishnan and A.C. Hindmarsh. Description and use of lsode, the livermore solver for ordinary differential equations. Technical Report UCRL-ID-113855, Lawrence Livermore National Laboratory, 1993.

[72] J.R.R.A. Martins, P. Sturdza, and J.J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, September 2003.

[73] V. Dolejsi, M. Feistauer, and J. Hozman. Analysis of semi-implicit dgfem for nonlinear convection-diffusion problems on nonconforming meshes. *Computer Methods in Applied Mechanics and Engineering*, 196:2813–2827, 2007.

[74] F. Giraldo. Hybrid Eulerian-Lagrangian semi-implicit time-integrators. *International Journal of Computers and Mathematics with Applications*, 52:1325–1342, 2006.

[75] M. Restelli and F. Giraldo. A conservative semi-implicit discontinuous Galerkin method for the Navier-Stokes equations in nonhydrostatic mesoscale modeling. *SIAM Journal on Scientific Computing*, 31:2231–2257, 2009.

[76] U.M. Ascher and S.J. Ruuth and B.T.R. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

[77] U.M. Ascher and S.J. Ruuth and R.J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25:151–167, 1997.

[78] J.G. Verwer and B.P. Sommeijer. An implicit-explicit Runge–Kutta–Chebyshev scheme for diffusion-reaction equations. *SIAM Journal on Scientific Computing*, 25(5):1824–1835, 2004.

[79] J.C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008.

[80] J.C. Butcher and Z. Jackiewicz. Implementation of diagonally implicit multistage integration methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 34(6):2119–2141, 1997.

[81] H. Zhang and A. Sandu. A second-order diagonally-implicit-explicit multi-stage integration method. *Procedia CS*, 9:1039–1046, 2012.

[82] R. D'Ambrosio and J.C. Butcher. Multivalue numerical methods for partitioned differential problems: from second order ODEs to separable Hamiltonians. Presentation given at Auckland Numerical Ordinary Differential Equations ANODE 2013 (celebration of the 80th birthday of John Butcher), January 2013.

[83] A. Cardoneand Z. Jackiewicz, A. Sandu, and H. Zhang. Extrapolation-based implicit-explicit general linear methods. *Numerical Algorithms*, pages 1–23, 2013.

[84] W.M. Wright. *General Linear Methods with Inherent Runge–Kutta Stability*. PhD thesis, The University of Auckland, 2002.

[85] J.C. Butcher and Z. Jackiewicz. A new approach to error estimation for general linear methods. *Numerische Mathematik*, 95(3):487–502, 2003.

[86] J.C. Butcher, Z. Jackiewicz, and H.D. Mittelmann. A nonlinear optimization approach to the construction of general linear methods of high order. *Journal of Computational and Applied Mathematics*, 81(2):181–196, 1997.

[87] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, 28(125):145–162, 1974.

[88] J.C. Butcher and Z. Jackiewicz. Construction of diagonally implicit general linear methods of type 1 and 2 for ordinary differential equations. *Applied Numerical Mathematics*, 21(4):385–415, 1996.

[89] W.C. Skamarock and J.B. Klemp. Efficiency and accuracy of the KlempWilhelmson time-splitting technique. *Monthly Weather Review*, 122(11):2623–2630, 1994.

[90] F.X. Giraldo and M. Restelli. A study of spectral element and discontinuous Galerkin methods for the Navier-Stokes equations in nonhydrostatic mesoscale atmospheric modeling: Equation sets and test cases. *Journal of Computational Physics*, 227(8):3849–3877, April 2008.

[91] S. Jebens, O.Knoth, and R. Weiner. Partially implicit peer methods for the compressible Euler equations. *Journal of Computational Physics*, 230(12):4955 – 4974, 2011.

[92] S. Boscarino, L. Pareschi, and G. Russo. Implicit-Explicit Runge–Kutta schemes for hyperbolic systems and kinetic equations in the diffusion limit. *SIAM Journal on Scientific Computing*, 35(1):A22–A51, 2013.

[93] M.P. Calvo, J. de Frutos, and J. Novo. Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations. *Applied Numerical Mathematics*, 37(4):535–549, 2001.

[94] H. Zhang, A. Sandu, and S. Blaise. Partitioned and implicitexplicit general linear methods for ordinary differential equations. *Journal of Scientific Computing*, pages 1–26, 2014.

[95] J. C. Butcher and Z. Jackiewicz. Construction of high order diagonally implicit multistage integration methods for ordinary differential equations. *Applied Numerical Mathematics*, 27(1):1 – 12, 1998.

[96] S. Blaise, J. Lambrechts, and E. Deleersnijder. A stable three-dimensional discontinuous galerkin discretization for nonhydrostatic atmospheric simulations. *submitted to Journal of Computational Physics*, 2014.

[97] X. Chen. Generation, propagation, and annihilation of metastable patterns. *Journal of Differential Equations*, 206(2):399 – 437, 2004.

[98] A.R. Bahadir. A fully implicit finite-difference scheme for two-dimensional burgers' equations. *Applied Mathematics and Computation*, 137(1):131 – 137, 2003.

[99] R. Comblen, S. Blaise, V. Legat, J. Remacle, E. Deleersnijder, and J. Lambrechts. A discontinuous finite element baroclinic marine model on unstructured prismatic meshes. part i: space discretization. *Ocean Dynamics*, 60(6):1371–1393, 2010.

[100] R.D. Nair, S.J. Thomas, and R.D. Loft. A discontinuous Galerkin transport scheme on the cubed sphere. *Monthly Weather Review*, 133:814–828, 2005.

[101] A. St-Cyr and D. Neckels. A fully implicit Jacobian-free high-order discontinuous Galerkin mesoscale flow solver. In Gabrielle Allen, Jarosaw Nabrzyski, Edward Seidel, GeertDick Albada, Jack Dongarra, and PeterM.A. Sloot, editors, *Computational Science ICCS 2009*, volume 5545 of *Lecture Notes in Computer Science*, pages 243–252. Springer Berlin Heidelberg, 2009.

[102] B. Seny, J. Lambrechts, R. Comblen, V. Legat, and J.-F. Remacle. Multirate time stepping for accelerating explicit discontinuous Galerkin computations with application to geophysical flows. *International Journal for Numerical Methods in Fluids*, 71(1):41–64, 2013.

[103] T. Karna, V. Legat, and E. Deleersnijder. A baroclinic discontinuous Galerkin finite element model for coastal flows. *Ocean Modelling*, 61(0):1–20, 2013.

[104] B. Seny, J. Lambrechts, T. Toulorge, V. Legat, and J. Remacle. An efficient parallel implementation of explicit multirate RungeKutta schemes for discontinuous Galerkin computations. *Journal of Computational Physics*, 256(0):135–160, 2014.

[105] A. Kameni, J. Lambrechts, J. Remacle, S. Mezani, F. Bouillault, and C. Geuzaine. Discontinuous Galerkin Method for computing induced fields in superconducting materials. *Magnetics, IEEE Transactions on*, 48(2):591–594, 2012.

[106] K. Burrage and J.C. Butcher. Non-linear stability of a general class of differential equation methods. *BIT*, 20(2):185–203, 1980.

[107] J.C. Butcher. On the convergence of numerical solutions to ordinary differential equations. *Mathematics of Computation*, 20(93):1–10, Jan 1966.

[108] G.J. Cooper. The order of convergence of general linear methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 15(4):643–661, 1978.

[109] E. Hairer and G. Wanner. Multistep-multistage-multiderivative methods for ordinary differential equations. *Computing*, 11(3):287–303, 1973.

[110] R. Skeel. Analysis of fixed-stepsize methods. *SIAM Journal on Numerical Analysis*, 13(5):664–685, Oct 1976.

[111] Z. Bartoszewski and Z. Jackiewicz. Construction of two-step Runge–Kutta methods of high order for ordinary differential equations. *Numerical Algorithms*, 18:51–70, 1998.

[112] Z. Jackiewicz and S. Tracogna. A general class of two-step Runge-Kutta methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 32(5):1390–1427, 1995.

[113] A.R. Renaut. Two-step Runge-Kutta methods and hyperbolic partial differential equations. *Mathematics of Computation*, 55(192):563–579, 1990.

[114] L. Skvortsov. Explicit two-step runge-kutta methods. *Mathematical Models and Computer Simulations*, 2:222–231, 2010. 10.1134/S2070048210020092.

[115] E.M. Constantinescu and A. Sandu. Optimal explicit strong stability preserving general linear methods. *SIAM Journal on Scientific Computing*, 32(5):3130–3150, 2010.

[116] E. Zharovsky, A. Sandu, and H. Zhang. A class of IMEX two-step Runge-Kutta methods. *SIAM Journal on Numerical Analysis*, in publish, 2014.

[117] J.C. Butcher. Stability properties for a general class of methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 18(1):37–44, feb 1981.

[118] W. Hundsdorfer and J. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2003.

[119] K. Price and R. Storn. Differential evolution (DE) for continuous function optimization. URL: http://www1.icsi.berkeley.edu/~storn/code.html.

[120] C.A. Kennedy and M.H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44(1-2):139–181, 2003.

[121] A. Robert. The integration of a spectral model of the atmosphere by the implicit method. *Proceedings of the WMO/IUGG Symposium on NWP, Tokyo*, Japan Meteorological Agency:VII.19–VII.24, 1969.

[122] Cleve Moller. *Experiments with Matlab*. MathWorks, 2011.

[123] I. Grooms and K. Julien. Linearly implicit methods for nonlinear PDEs with linear dispersion and dissipation. *Journal of Computational Physics*, 230(9):3630 – 3650, 2011.

[124] G. Akrivis, O. Karakashian, and F. Karakatsani. Linearly implicit methods for nonlinear evolution equations. *Numerische Mathematik*, 94:403–418, 2003.

[125] G. Akrivis and M. Crouzeix. Linearly implicit methods for nonlinear parabolic equations. *Mathematics of Computation*, 73(246):pp. 613–635, 2004.

[126] R.Weiner, B.A. Schmitt, and H. Podhaisky. ROWMAP-a ROW-code with Krylov techniques for large stiff ODEs. *Applied Numerical Mathematics*, 25:303–319, 1997.

[127] J. Douglas Jr. Alternating direction methods for three space variables. *Numerische Mathematik*, 4(1):41–63, 1962.

[128] D. Peaceman and H. Rachford, Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.

[129] A. Sandu. *Numerical Aspects of Air Quality Modeling*. PhD thesis, University of Iowa, 1997.

[130] P.J. van der Houwen and B.P. Sommeijer. Approximate factorization for time-dependent partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1-2):447–466, 2001.

[131] S. Beck, R. Weiner, H. Podhaisky, and B. Schmitt. Implicit peer methods for large stiff ode systems. *Journal of Applied Mathematics and Computing*, 38(1-2):389–406, 2012.

[132] M. Berzins and J.M. Ware. Solving convection and convection-reaction problems using the method of lines. *Applied Numerical Mathematics*, 20(12):83 – 99, 1996. Method of Lines for Time-Dependent Problems.

[133] I. Ahmad and M. Berzins. An algorithm for ODEs from atmospheric dispersion problems. *Applied Numerical Mathematics*, 25(23):137 – 149, 1997. Special Issue on Time Integration.

[134] M. P. Calvo and A. Gerisch. Linearly implicit Runge-Kutta methods and approximate matrix factorization. *Applied Numerical Mathematics*, 53(2):183–200, 2005.

[135] D. Dopico, Y. Zhu, A. Sandu, and C. Sandu. Direct and adjoint sensitivity analysis of multibody systems using Maggi's equations. In *ASME 2013 IDETC/CIE 9th Int. Conf. on Multibody Systems and Nonlinear Dynamics (MSND)*, volume DETC2013-12696, Portland, OR, Aug. 4-7 2013.

[136] D. Dopico, Y. Zhu, A. Sandu, and C. Sandu. Index-1 and index-3 adjoint dae equations for the sensitivity analysis of multibody systems. In *ECCOMAS, Multibody Dynamics*, volume 208, Zagreb, Croatia, July 1-4 2013.

[137] Y. Zhu, D. Dopico, C. Sandu, and A. Sandu. Sensitivity analysis of vehicle dynamics based on multibody models. In *ASME 2013 IDETC/CIE Proc. of ASME 2013 IDETC/CIE 15th Int. Conf. on Advance Vehicle Technologies (AVT)*, volume DETC2013-13212, Portland, OR, Aug. 4-7 2013.

[138] Y. Zhu, D. Dopico, C. Sandu, and A. Sandu. Adjoint second order ODE for the computation of first order sensitivities in multibody dynamics. In *ECCOMAS, Multibody Dynamics*, volume 236, Zagreb, Croatia, July 1-4 2013.

[139] D. Dopico, A. Sandu, C. Sandu, and Y. Zhu. Sensitivity analysis of multibody dynamic systems modeled by ODEs and DAEs. In Zdravko Terze, editor, *Multibody Dynamics*, volume 35 of *Computational Methods in Applied Sciences*, pages 1–32. Springer International Publishing, 2014.

[140] V. Rao and A. Sandu. An adjoint-based scalable algorithm for time-parallel integration. *Journal of Computational Science*, 5(2):76 – 84, 2014. Empowering Science through Computing + BioInspired Computing.

[141] V. Rao and A. Sandu. A posteriori error estimates for DDDAS inference problems. *Procedia Computer Science*, 29(0):1256 – 1265, 2014. 2014 International Conference on Computational Science.

[142] Course: Applied Mathematical Modeling. URL: http://www.math.vt.edu/index.php, 2014.