Spring 5-5-2023

Adaptive Multirate Infinitesimal Time Integration

Alex Fish
Southern Methodist University, afish@smu.edu

ADAPTIVE MULTIRATE INFINITESIMAL

TIME INTEGRATION

Approved by:

Dr. Daniel Reynolds,
Professor of Mathematics

Dr. Usama El Shamy,
Professor of Civil Engineering

Dr. Thomas Hagstrom,
Professor of Mathematics

Dr. Johannes Tausch,
Professor of Mathematics

ADAPTIVE MULTIRATE INFINITESIMAL TIME INTEGRATION

A Dissertation Presented to the Graduate Faculty of the

Dedman College: School of Humanities and Sciences

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Computational and Applied Mathematics

by

Alex Fish

B.S., University of Nebraska Omaha
M.S., University of Washington

May 13, 2023

ACKNOWLEDGMENTS

I'd like to recognize and thank my advisor, committee members, collaborators, and family without whom this work would not have been possible.

My thesis advisor, Professor Daniel Reynolds, has been invaluable in my success in research. I first learned of both the depth of our field of research and his style of problem solving in his course on numerical ordinary differential equations during my first semester. His style and research interests were a perfect fit for me and I began working with him before the semester was finished. His guidance and collaboration have been instrumental in my understanding of the field and productivity in research.

I'd like to thank my thesis committee of Professor Usama El Shamy, Professor Thomas Hagstrom, and Professor Johannes Tausch. Their range of perspectives allowed them to give input which made my work more well-rounded and robust. While I did not take any courses from Professor El Shamy or Professor Hagstrom, I was able to take several courses with Professor Tausch which broadened my understanding of numerical mathematics.

I'd also like to thank my collaborators at Lawrence-Livermore National Laboratory including Dr. Carol Woodward, Dr. David Gardner, Cody Balos, and Dr. Steven Roberts. Through regular discussions, they provided valuable insight into real-world use cases of the research, providing motivation for improvements upon my research and robust numerical experiments. I'd especially like to thank Dr. Roberts, a co-author on a portion of my work, whose knowledge of numerical ordinary differential equations is unparalleled.

Without the emotional support of my friends and family, including my mother Chris and my father Kevin, throughout my entire education, this would not have been possible.

I was fortunate enough to receive support from SMU in my research pursuits through the Research Training Group Fellowship. Support for my work also came from the Scientific Discovery through Advanced Computing (SciDAC) project "Frameworks, Algorithms and

Scalable Technologies for Mathematics (FASTMath)," funded by the U.S. Department of Energy Office of Advanced Scientific Com- puting Research and National Nuclear Security Administration, under Lawrence Livermore National Laboratory subcontract B626484 and DOE award DE-SC0021354.

Fish, Alex

B.S., University of Nebraska Omaha
M.S., University of Washington

Adaptive Multirate Infinitesimal
Time Integration

Advisor: Professor Daniel Reynolds
Doctor of Philosophy degree conferred May 13, 2023
Dissertation completed April 14, 2023

As multiphysics simulations grow in complexity and application scientists desire more accurate results, computational costs increase greatly. Time integrators typically cater to the most restrictive physical processes of a given simulation, which can be unnecessarily computationally expensive for the less restrictive physical processes. Multirate time integrators are a way to combat this increase in computational costs by efficiently solving systems of ordinary differential equations that contain physical processes which evolve at different rates by assigning different time step sizes to the different processes. Adaptivity is a technique for further increasing efficiency in time integration by automatically growing and shrinking the time step size to be as large as possible to achieve a solution accurate to a prescribed tolerance value. Adaptivity requires a time step controller, an algorithm by which the time step size is changed between steps, and benefits from an integrator with an embedding, an efficient way of estimating the error arising from each step of the integrator. In this thesis, we develop these required aspects for multirate infinitesimal time integrators, a subclass of multirate time integrators which allow for great flexibility in the treatment of the processes that evolve at the fastest rates. First, we derive the first adaptivity controllers designed specifically for multirate infinitesimal methods, and we discuss aspects of their computational implementation. Then, we derive a new class of efficient,

flexible multirate infinitesimal time integrators which we name implicit-explicit multirate infinitesimal stage-restart (IMEX-MRI-SR) methods. We derive conditions guaranteeing up to fourth-order accuracy of IMEX-MRI-SR methods, explore their stability properties, provide example methods of orders two through four, and discuss their performance. Finally, we derive new instances of the class of implicit-explicit multirate infinitesimal generalized-structure additive Runge-Kutta methods, developed by Chinomona and Reynolds (2022), with embeddings and explore their stability properties and performance.

TABLE OF CONTENTS

LIST OF FIGURES

To Melissa.

Chapter 1

Introduction

1.1. Multiphysics problems

A multiphyics initial-value problem (IVP), is an IVP which contains the coupling of various physical processes. Multiphysics problems appear in many areas of science as they are used to model complex real-world phenomena. Areas of science that commonly model multiphysics behavior include aerospace engineering, climate modeling, cosmological modeling, nuclear reactor design, and more. Multiphysics problems can prove to be quite computationally challenging and are a highly active area of research with some of the largest supercomputers in the world are built with multiphysics applications in mind [27].

A canonical example of a system with multiple coupled physical processes is the combustion of a jet of flammable gas, where the reaction of the gas, forming a flame, occurs simultaneously with the general motion along the direction of travel and the dispersion of the gas.

These problems are defined in terms of differential equations. A common partial differential equation formulation of a multiphysics problem is the advection-diffusion-reaction equation,

$$y' = \nabla \cdot (D\nabla y) - \nabla \cdot F(y) + R(y,t). \tag{1.1}$$

y is a quantity of interest, such as the density of a fluid, which can vary in time and space. The term $\nabla \cdot (D\nabla y)$ corresponds to the diffusion, or general spreading over space, of the quantity. The term $\nabla \cdot F(y(t))$ corresponds to the advection, or general motion through space along a direction, of the quantity. The term $R(y,t)$ corresponds with the interaction

of the quantity with sources or sinks, often called the "reaction" term. The term y' indicates that the terms on the right-hand side of the equation describe the rate at which the quantity changes over time. The quantity of interest y and right-hand side operators are often vector-valued, or, when appropriate, matrix-valued, coupling values such as velocity, momentum, energy, or others.

1.2. Multirate problems and methods

Multirate IVPs are a subclass of multiphysics IVPs in which the various physical processes defining the problem evolve at different rates. For example, the turbulent mixing of a fluid can evolve much more rapidly than the general motion of the fluid in the direction of travel. Multirate time integrators, commonly referred to as multirate methods, are time integrators designed to efficiently solve multirate IVPs by partitioning the right-hand side into various operators and assigning a different step size to each. By assigning differing step sizes to the various operators involved, the slower-evolving operators can be evaluated less frequently, thereby saving computational cost.

1.2.1. Multirate problem formulation

An initial-value problem is defined as

$$y'(t) = f(t, y(t)),\ y(t_0) = y_0 \tag{1.2}$$

where y can be vector-valued, and y_0 is a known initial-condition value. A multirate problem additively partitions the right-hand side function f into the different physical processes,

$$y'(t) = \sum_{\nu=1}^{n} f^{\{\nu\}}(t, y(t)),\ y(t_0) = y_0. \tag{1.3}$$

While a multirate problem can consist of arbitrarily many physical processes, they are typically grouped into fast-changing and slow-changing processes, often called the fast dy-

namics and slow dynamics, respectively.

$$y'(t) = f^{\{F\}}(t, y(t)) + f^{\{S\}}(t, y(t)),\ y(t_0) = y_0. \tag{1.4}$$

In the example (1.1), the diffusion and advection commonly change much more slowly over time than the reaction term so the problem might be partitioned as

$$f^{\{S\}} = \nabla \cdot (D\nabla y) - \nabla \cdot F(y),\ f^{\{F\}} = R(y, t).$$

The slow dynamics might include stiff processes characterized by having large eigenvalues (computed by taking the maximum absolute value of the Jacobian of the partition), requiring smaller steps to be stable if the slow dynamics are treated explicitly. Instead of taking smaller steps, the method could solve the slow dynamics implicitly. This requires increased computational work per step but may allow for larger time steps. For additional flexibility and efficiency, the slow dynamics can be further partitioned into components handled implicitly and explicitly. This is commonly referred to as an IMEX partitioning,

$$y'(t) = f^{\{F\}}(t, y(t)) + f^{\{I\}}(t, y(t)) + f^{\{E\}}(t, y(t)),\ y(0) = y_0, \tag{1.5}$$

and leads to the benefit of simpler implicit solves in the time integration algorithm, stemming from the implicit partition containing less information than the full slow partition. In the above IMEX multirate partitioning (1.5), the implicit and explicit partitions are both considered to be part of the slow dynamics. In the example (1.1), the diffusion term may be both stiff and evolve on the slow timescale, so the problem might be partitioned as

$$f^{\{I\}} = \nabla \cdot (D\nabla y),\ f^{\{E\}} = -\nabla \cdot F(y),\ f^{\{F\}} = R(y, t).$$

1.2.2. Multirate infinitesimal methods

Multirate infinitesimal (MRI) methods are a subclass of multirate methods which extend the theory of Runge-Kutta methods and Additive Runge-Kutta (ARK) methods [3, 6, 23, 26]. Runge-Kutta methods are one-step multi-stage methods. That is, they compute a step of the solution using only the most recent solution value to build up the next step in the solution by computing several stages. Computing these stages involves taking linear combinations of evaluations of the IVP's right-hand side function. One step of an s-stage Runge-Kutta method is computed as follows.

$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right),\ i = 1, ..., s, \tag{1.6a}$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i. \tag{1.6b}$$

Here, y_n is the solution value at the previous step, y_{n+1} is the computed solution value at the next step, t_n is the time at which the previous step was computed, and h is the size of the time step. A is an $s \times s$ matrix and b and c are s-length vectors of coefficients defining the specific method used.

Multirate infinitesimal methods are similarly one-step multi-stage methods. However, multirate infinitesimal methods instead compute each stage by solving a sequence of modified fast IVPs. One step of a typical $s^{\{S\}}$-stage explicit multirate infinitesimal method is defined as follows.

$$\text{Let: } Y_1^{\{S\}} := y_n \tag{1.7a}$$

$$\text{For: } i = 2, ..., s^{\{S\}} \tag{1.7b}$$

$$\begin{cases} \text{Let:} & v_i(\theta_{i,0}) := Y_{i-1}^{\{S\}} \\ \text{Solve:} & v_i'(\theta) = f^{\{F\}}\left(\theta, v_i(\theta)\right) + g_i(\theta), \text{ for } \theta \in [\theta_{i,0}, \theta_{i,f}] \\ \text{Let:} & Y_i^{\{S\}} := v_i(\theta_{i,f}), \end{cases} \tag{1.7c}$$

$$\text{Let: } y_{n+1} := Y_{s^{\{S\}}}^{\{S\}}. \tag{1.7d}$$

The structure of the function $g_i(\theta)$ is determined by the class of multirate infinitesimal method used, and is typically constructed using linear combinations of $f^{\{S\}}$ from previous stages.

The fast IVPs are solved with a time step of $h = H/M$ where H is the time step size of the overall multirate method and M is the multirate ratio of the problem, a factor indicating how much faster the fast dynamics evolve than the slow dynamics. A higher value of M indicates a high degree of separation between the fast and slow dynamics.

These methods are highly flexible, allowing usage of any time integration algorithm for solving the fast IVPs. These methods are efficient in that they require far fewer evaluations of $f^{\{S\}}$ than $f^{\{F\}}$, where $f^{\{S\}}$ is often much more expensive to evaluate.

1.3. Temporal adaptivity

Basic usage of a numerical time integration algorithm is to select a fixed time step size and evolve the solution step-by-step from the initial condition to some desired time. However, frequently the user of a numerical time integration algorithm has no preference on the exact step size used, and would rather obtain a solution that is accurate to a prescribed tolerance from the exact solution of the problem in as efficient a manner as possible. For example, initial conditions may only be known to be accurate to two decimal places, so any computational work performed in obtaining further accuracy in the IVP solution may be considered wasted. In this case, the IVP can be solved with temporal adaptivity.

Temporal adaptivity, the process of adjusting the time step size between steps of the numerical integration algorithm, seeks to select the largest time step size at any point such that the solution obtained over that step is accurate to a user-prescribed tolerance. Typically, the change in time step size is informed by a short history of previous time step sizes and a short history of estimated solution error values at those steps. This history usually includes one to three time step sizes and one to three estimated error values. There are alternate formulations of temporal adaptivity which are informed by the characteristics of the problem and the spatial discretization [12], but we do not focus on those here.

Classical examples of time step controllers include the so-called "elementary" controller,

$$h_{n+1} = h_n \left(\frac{\text{TOL}}{\varepsilon_{n+1}} \right)^{\frac{1}{p}}, \tag{1.8}$$

the I-controller [54],

$$h_{n+1} = h_n \left(\frac{\text{TOL}}{\varepsilon_{n+1}} \right)^{\frac{k_1}{p}}, \tag{1.9}$$

and the PID-controller [23],

$$h_{n+1} = h_n \left(\frac{\text{TOL}}{\varepsilon_{n+1}} \right)^{\frac{k_1}{p}} \left(\frac{\text{TOL}}{\varepsilon_n} \right)^{-\frac{k_2}{p}} \left(\frac{\text{TOL}}{\varepsilon_{n-1}} \right)^{\frac{k_3}{p}}. \tag{1.10}$$

Here, k_1, k_2, and k_3 are free parameters, p is the order of accuracy of the method used, h_n is the previous time step size, h_{n+1} is the predicted next time step size, TOL is the user-prescribed tolerance value, and ε_{n+1} is the estimated solution error from using time step size h_n to compute y_{n+1}. This estimate of the solution error is efficiently computable if a given method has an embedding, a means of providing a solution with a different order of accuracy to the primary method with few (or zero) extra function evaluations.

The solution error estimate is a *local* error estimate, which assumes the previous time step solution y_n is exact. Thus, these time step controllers do not perfectly control the overall

global error observed, often allowing error buildup over time. These buildup effects are usually negligible and can be tuned to acceptable levels through the free parameters. Error control techniques exist to control global error values, but these typically involve solving the entire IVP several times, searching for a fixed time step size which provides a desired global error value.

1.4. Aim of thesis

The aim of this work is to develop robust approaches for temporal adaptivity in MRI methods. The first multirate time step controllers for MRI methods are introduced, which take advantage of the algorithmic structure of the methods for increased efficiency. Additionally, a new class of MRI methods are introduced, named implicit-explicit multirate infinitesimal stage-restart (IMEX-MRI-SR), that improves upon the class of IMEX-MRI-GARK methods. The algorithmic structure of IMEX-MRI-SR methods allows for far easier creation of high quality, efficient methods with embeddings than existing multirate infinitesimal methods. Finally, the first two IMEX-MRI-GARK methods with embeddings are introduced to provide more options and a more thorough testing environment for the ecosystem of efficient adaptivity-capable IMEX multirate infinitesimal methods.

1.5. Review of multirate infinitesimal methods

The earliest forms of multirate infinitesimal methods are operator splitting methods, such as the first-order accurate Lie–Trotter splitting method [37] and the second-order accurate Strang–Marchuk splitting method [36, 58].

One step of the Lie–Trotter splitting method applied to an IVP of the form (1.4) may be computed as follows.

$$\text{Let: } y_{n+1}^{(1)} = y_n + Hf^{\{S\}}(t_n, y_n), \tag{1.11a}$$

$$\begin{cases} \text{Let: } v(0) = y_{n+1}^{(1)}, \\ \text{Solve: } v'(\theta) = f^{\{F\}}(t_n + \theta, v), \text{ for } \theta \in [0, H], \end{cases} \tag{1.11b}$$

$$\text{Let: } y_{n+1} = v(H). \tag{1.11c}$$

This corresponds to one step with the Forward Euler method to evolve the slow dynamics, followed by a fast IVP solve to evolve the fast dynamics. The Lie–Trotter method can be extended to solve IVPs of the form (1.5). This is merely one form of a Lie–Trotter splitting method. In general, Lie–Trotter methods focus on applying one operator at a time.

One step of the Strang–Marchuk splitting method applied to an IVP of the form (1.4) may be computed as follows.

$$\text{Let: } y_{n+1}^{(1)} = y_n + \tfrac{H}{4} f^{\{S\}}\left(t_n, y_n\right) \tag{1.12a}$$

$$+ \tfrac{H}{4} f^{\{S\}}\left(t_n + \tfrac{H}{2}, y_n + \tfrac{H}{2} f^{\{S\}}\left(t_n, y_n\right)\right), \tag{1.12b}$$

$$\begin{cases} \text{Let: } v(0) = y_{n+1}^{(1)}, \\ \text{Solve: } v'(\theta) = f^{\{F\}}\left(t_n + \theta, v\right), \text{ for } \theta \in [0, H], \end{cases} \tag{1.12c}$$

$$\text{Let: } y_{n+1}^{(2)} = v(H), \tag{1.12d}$$

$$\text{Let: } y_{n+1} = y_{n+1}^{(2)} + \tfrac{H}{4} f^{\{S\}}\left(t_n + \tfrac{H}{2}, y_{n+1}^{(2)}\right) \tag{1.12e}$$

$$+ \tfrac{H}{4} f^{\{S\}}\left(t_{n+1}, y_{n+1}^{(2)} + \tfrac{H}{2} f^{\{S\}}\left(t_n + \tfrac{H}{2}, y_{n+1}^{(2)}\right)\right).$$

This corresponds to a half-step of the explicit Heun method to evolve the slow dynamics, followed by a fast IVP solve to evolve the fast dynamics, and finally another half step with the explicit Heun method to finish evolving the slow dynamics. The Strang–Marchuk method can also be extended to solve IVPs of the form (1.5). Like with Lie–Trotter methods, this is

merely one form of a Strang–Marchuk method. In general, Strang–Marchuk methods focus on applying half steps with any second order method to $f^{\{S\}}$.

These operator splitting methods have some flexibility in terms of the order in which the different dynamics are evolved but they are limiting in their low orders of accuracy. Other splitting methods which can be extended to IVPs of the form (1.5) exist [4, 57], though they, along with any operator splitting method with higher than second-order accuracy, require evolving the fast dynamics backwards in time which can lead to instability [11].

The seminal work introducing multirate infinitesimal methods outside of the operator splitting context was proposed by Knoth and Wolke in 1998 [28]. This work proposed methods which evolve the slow dynamics explicitly and the fast dynamics through a sequence of fast IVPs. In 2009, Wensch et al. coined the term multirate infinitesimal step (MIS) [61] to describe the methods and formalized the structure. The term infinitesimal implies the assumption that the fast IVPs are solved exactly, or with infinitely small steps, to achieve a solution, although in practice the fast IVPs are solved with another method as mentioned in Section 1.2.2. Subsequent work by this group and others proposed several third-order accurate MIS methods [50, 51, 62] and the first fourth-order accurate MIS method [52].

In 2015, Sandu and Günther proposed a new mathematical theory that generalized the idea of additive Runge-Kutta methods [23] such that the solution was computed using multiple sets of stages, with each set having their own corresponding sets of coefficients [46]. These generalized-structure additive Runge–Kutta (GARK) methods proved instrumental as a foundation for deriving new kinds of multirate infinitesimal methods. Notably, GARK method formulation and order conditions will be used in this thesis, and are discussed in Section 2.3.

Sandu used his GARK theory in 2019 to extend MIS methods, allowing arbitrary degree of polynomials in time for the forcing functions $g_i(\theta)$ in (1.7c) (previously these had been restricted to constant in time for MIS methods) and allowing implicit slow stages for added stability. This new class of methods was named multirate infinitesimal GARK (MRI-GARK)

[45]. Roberts et al. extended MRI-GARK methods to provide increased coupling for further stability, but with the drawback of more computationally expensive solves, with their MRI-GARK step predictor-corrector (MRI-GARK-SPC) methods [41]. In 2022, Chinomona and Reynolds proposed an extension to MRI-GARK, IMEX-MRI-GARK [5], which allowed for IMEX partitioning of the slow dynamics which which increases flexibility of usage and reduces cost compared to methods that treat the slow dynamics fully implicitly. IMEX-MRI-GARK and by reduction, MRI-GARK, method formulation, order conditions, and stability are discussed in Section 2.4.

Luan et al. proposed a new class of multirate infinitesimal methods as an extension to exponential Runge-Kutta (ExpRK) methods [21, 30, 33] called multirate exponential Runge-Kutta (MERK) [32]. The derivation of these methods took a different approach than MRI-GARK methods, in that they used ExpRK theory rather than GARK theory to build the methods. While this dramatically simplified construction of MERK methods due to their vastly reduced number of order conditions, MERK methods inherited the restrictions of ExpERK methods, namely that the fast operator of (1.4) must be linear. Luan et al. then proposed a similar new class of multirate infinitesimal methods called multirate exponential Rosenbrock (MERB) methods [32], as an extension to exponential Rosenbrock (ExpRB) methods [34, 35]. These methods inherited a similar but more lax requirement on the fast dynamics, where the underlying problem may have a nonlinear fast operator but the method acts on a form of the problem where the fast dynamics have been *linearized*. MERK method formulation is discussed in Section 2.6.

While MERK and MERB methods were restricted to being applied to certain types of problems, they could achieve higher order accuracy than any published method in the MRI-GARK family due to their reliance on exponential method theory. Luan et al. introduced a fifth-order accurate MERK method, a fifth-order accurate MERB method, and a sixth-order accurate MERB method, the first multirate infinitesimal methods higher than fourth-order accurate published.

Fish and Reynolds proposed a new class of methods that extend both IMEX-MRI-GARK and MERK methods to include IMEX treatment of the slow dynamics and allow for nonlinear fast dynamics named IMEX-MRI-SR methods [10], included in Chapter 4 of this thesis. This work allowed for far easier derivation of high quality instances of the class of methods with efficiently computable embeddings. Additionally, this work provides the first theoretical proof that despite Luan et al.'s assumptions about the linearity of the fast dynamics, MERK methods can be applied to problems with nonlinear fast dynamics without order reduction.

1.6. Review of adaptive time step controllers

The creation of time step controllers takes inspiration from control theory, a field of engineering dealing with automatic control and stabilization of machines and processes [7], and starts with an analytical expression for the error from a given step of a time integration algorithm,

$$\varepsilon_{n+1} = \phi_n h_n^{p+1} + \mathcal{O}(h^{p+2}). \tag{1.13}$$

Here, ε_{n+1} is the error in the solution achieved when evolving from time t_n to time $t_n + h_n$ and p is the order of accuracy of the method used. $\phi_n = \phi(t_n)$ is a function which depends on time but does not depend on h and is unknown in general as it incorporates high-order derivative terms of the exact solution of the IVP and depends on the specific method being used. This analytical error expression is then approximated by truncating to only the lowest-order term,

$$\varepsilon_{n+1} \approx \phi_n h_n^{p+1}. \tag{1.14}$$

Different controllers are derived by assuming different forms of the function ϕ.

The I-controller (1.9) is the most elementary time step controller. It has been discussed in the context of IVPs at least as early as 1971 [13] and likely earlier. This controller assumes ϕ is constant from one step to the next [54] and thus depends only on the most recent time

step size and error value. This controller is able to react to sudden increases in stiffness of the IVP but can often get stuck oscillating between very small steps which produce a small error value, leading to a large increases in step size which produce a large error value, leading back to very small steps.

Controllers with extended error histories attempt to fix the issue of rapid oscillations by smoothing out the change in step size, using values of the step size or error from further in the past to inform the new step size. Such controllers include the PI-controller [17],

$$h_{n+1} = h_n \left(\frac{\mathrm{TOL}}{\varepsilon_{n+1}}\right)^{\frac{k_1}{p}} \left(\frac{\mathrm{TOL}}{\varepsilon_n}\right)^{-\frac{k_2}{p}}, \tag{1.15}$$

and the PID-controller (1.10) which depend on the previous time step value and two or three previous error values, respectively. These controllers tend to perform far better than the basic I-controller [54].

Histories of time step controllers can include any number of previous time step sizes and error values, and can take on a wide range of structures [54, 55, 56, 15, 16].

One controller is of particular interest for its thorough derivation from principals of control theory. This is the controller of Gustafsson [16],

$$h_{n+1} = h_n \frac{h_n}{h_{n-1}} \left(\frac{\mathrm{TOL}}{\varepsilon_{n+1}}\right)^{\frac{k_1}{p}} \left(\frac{\varepsilon_n}{\varepsilon_{n+1}}\right)^{\frac{k_2}{p}}. \tag{1.16}$$

This controller has a further smoothing factor of h_n/h_{n-1} which dampens the potential growth in step size if the previous step size was smaller than the one before. The techniques used in the derivation of this controller were used in the development of the multirate infinitesimal controllers developed in this thesis.

The first and, before the work comprising this thesis, only multirate time step controllers was proposed by Sarshar et al. [47]. These controllers were designed for multirate GARK (MRGARK) methods [14, 39], a subclass of GARK methods which, should be noted, are not infinitesimal methods. Sarshar et al. proposed two types of controllers with different goals,

one to ensure that the error coming from the fast and slow dynamics remains approximately equal, and one to yield the lower computational costs. These controllers require three different embeddings to compute three separate error estimates ε_{n+1}, $\varepsilon_{n+1}^{\{F\}}$, and $\varepsilon_{n+1}^{\{S\}}$. These represent the overall error estimate and the estimates of errors coming specifically from the evolution of the fast and slow dynamics, respectively. Their first "balancing error" controller takes a fairly standard form as seen above,

$$H_{n+1} = H_n \left(\frac{\text{TOL}}{\varepsilon_{n+1}} \right)^{\frac{1}{p}}, \tag{1.17}$$

$$M_{n+1} = M_n \left(\frac{\varepsilon_{n+1}^{\{F\}}}{\varepsilon_{n+1}^{\{S\}}} \right)^{\frac{1}{q}}. \tag{1.18}$$

Here, p is the order of accuracy of the method, and $q = \min(p, \hat{p})$ where $\hat{p}$ is the order of accuracy of the embeddings. The H update function is equal to the I-controller with the free parameter $k_1 = 1$ and the M update function has a similarly short history. Their second "efficiency optimization" controller updates H and M to be the results of an optimization problem which simplifies to

$$M_{n+1} = \underset{M_{n+1}}{\text{argmin}} \; \frac{t^{\{S\}} + M_{n+1} t^{\{F\}}}{H_n} \left(\varepsilon_{n+1}^{\{S\}} + \varepsilon_{n+1}^{\{F\}} \frac{M_n^q}{M_{n+1}^q} \right)^{\frac{1}{q+1}}, \tag{1.19}$$

$$H_{n+1} = H_n \left(\varepsilon_{n+1}^{\{S\}} + \varepsilon_{n+1}^{\{F\}} \frac{M_n^q}{M_{n+1}^q} \right)^{-\frac{1}{q+1}}, \tag{1.20}$$

which is structured quite differently than previous controllers. Here $t^{\{S\}}$ and $t^{\{F\}}$ are the computational costs of evolving the slow and fast stages, respectively.

These multirate controllers are less suitable for multirate infinitesimal methods. Computing the four solutions necessary for three error estimates can be computationally expensive because infinitesimal methods use the last stage computed as the step solution, which might involve an implicit solve or fast IVP solve, and thus do not have as cheaply computable

embeddings as MRGARK methods. Additionally, the difference between overall error and slow error is not clear, as multirate infinitesimal methods consider the last "slow stage" to be the step solution.

Fish and Reynolds developed the first family of controllers designed for multirate infinitesimal methods, included in this thesis. The derivation of these controllers considers the infinitesimal structure and includes discussions on efficiently computing error estimates. One such controller is the so-called "Constant-Constant" controller,

$$H_{n+1} = H_n \left(\frac{\text{TOL}}{\varepsilon_{n+1}^{\{S\}}} \right)^{k_1/P}, \tag{1.21}$$

$$M_{n+1} = M_n \left(\frac{\text{TOL}}{\varepsilon_{n+1}^{\{S\}}} \right)^{(p+1)k_1/(Pp)} \left(\frac{\text{TOL}}{\varepsilon_{n+1}^{\{F\}}} \right)^{-k_2/p}, \tag{1.22}$$

where P is the order of accuracy of the multirate infinitesimal method, and p is the order of accuracy of the method used to solve the fast IVPs. Here, again, we see an I-controller like structure and history. Fish and Reynolds also introduced controllers inspired by the classical PI-controller (1.15), PID-controller (1.10), and Gustaffson's controller (1.16).

1.7. Thesis outline and summary of contributions

This dissertation consists of five chapters. These chapters are: introduction (Chapter 1, background theory (Chapter 2), new results (Chapters 3-4), and conclusion (Chapter 6).

Chapter 1 discusses the state of multirate infinitesimal method literature and the motivation to develop previously untouched aspects of the topic.

Chapter 2 lays out background theory which the work in this thesis will build upon. This includes a discussion on the motivation behind time step controllers and methods with embeddings, time step controller derivation theory, GARK theory, IMEX-MRI-GARK theory, derivation of new IMEX-MRI-GARK methods, and a description of MERK methods.

Chapter 3 develops the first time step controllers designed for multirate methods. This work is inspired by Gustafsson's control theoretic derivations of his controller (1.16) and introduces a family of new time step controllers which simultaneously update the time step H and the multirate ratio M, discusses computational implementation aspects, and tests the controllers against classical controllers with a variety of performance measurements on a suite of test problems using a range of MRI-GARK methods with embeddings. This work has been accepted for publication.

Chapter 4 derives a new class of implicit-explicit multirate infinitesimal methods named implicit-explicit mulriate infinitesimal stage-restart (IMEX-MRI-SR) methods, derives conditions guaranteeing orders of accuracy up to four, analyzes linear stability, provides example methods with embeddings, and performs numerical experiments, comparing the example methods against existing IMEX-MRI-GARK methods.

Chapter5 proposes the first IMEX-MRI-GARK methods with embeddings, one second-order method and one third-order method, details their construction, explores their stability properties, and evaluates their performance against existing IMEX multirate infinitesimal methods.

Chapter 6 summarizes the contributions and discusses potential future work.

Background theory Sections 2.1 and 2.2 are useful in understanding Chapter 3. Background theory Sections 2.3, 2.4, 2.5, 2.6, and optionally 2.1 are useful in understanding Chapter 4.

Chapter 2

Background Theory

In this chapter, we lay out the groundwork theory of some of the inspirational topics behind the work in this thesis. First, the motivation behind using time step controllers and methods with embeddings. Next, we discuss we discuss control-theoretic derivations of time-step controllers for classical "single-rate" Runge–Kutta methods. Then, we lay the groundwork GARK theory with which many multirate infinitesimal methods rely upon, including the method introduced in Chapter 4. Then, we discuss the theory of IMEX-MRI-GARK methods, their order conditions, and linear stability. We additionally discuss the design and derivation of efficient IMEX-MRI-GARK methods with embeddings. Finally, we describe MERK methods, their differences from the wider family of MRI-GARK methods, and their limitations.

2.1. Time step controller and method embedding motivation

A brute force method of choosing a step size to achieve a given error from solving an IVP numerically is to solve the IVP over the entire desired time interval several times with different choices of fixed step sizes. These solutions, when compared to a known reference solution (generated from an analytical formula or another numerical solution which used much smaller time steps), provide a measurement of the global error of the numerical solution. The step size which provided the desired global error is then chosen for future solves. While this error error measurement is desirable, solving the entire IVP several times and generation of the reference solution is very costly computationally.

A more efficient method of achieving a solution accurate to a given tolerance value is to use a time step controller. A time step controller changes the time step size between steps of

the time integration algorithm such that the local error, or an estimate of it, is approximately equal to the given tolerance value. The local error differs from the global error in that it is the error from a single step of the time integration algorithm, assuming no error occurred before the step, and therefore does not consider any accumulation of error from one step to the next. This leads to a less accurate control of the global error but is far less costly as the IVP need only be solved once, and the IVP could require no more work than a single fixed-step solve using embeddings, described below. Performance can additionally be tuned by altering the values of the controller's parameters.

Using a time step controller requires an estimate of the error between the exact solution of the IVP the time integration method's solution for a given step. This can be achieved by taking the difference between two solutions computed with methods of different orders of accuracy. In this setting, the solution with the higher order of accuracy is estimated to be the exact solution to the IVP. Often the lower order of accuracy of the two solutions is used as the value of p in the time step controllers.

This error estimate could be computed with two completely separate methods taking separate steps, but this would require roughly twice the computational work to evolve the solution a single step. The error estimate can be more cheaply computed through the use of an embedding.

An embedding is an alternate set of coefficients which can be used to compute a solution with an alternate order of accuracy for very little extra computational work once the primary solution has been computed. For example, a Runge–Kutta method's embedded solution, $\hat{y}_{n+1}$, is computed with the alternate set of coefficients $\hat{b}_i, i = 1, ..., s$ as follows:

$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right),\ i = 1, ..., s \tag{2.1a}$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \tag{2.1b}$$

$$\hat{y}_{n+1} = y_n + h \sum_{i=1}^{s} \hat{b}_i k_i \tag{2.1c}$$

y_{n+1} and $\hat{y}_{n+1}$ are solutions with different orders of accuracy. For example, they could be third- and second-order accurate, respectively. Computing the stages $k_i, i = 1, ..., s$ comprises the majority of the computational work which is usually measured in the number of times the righthand side function f is evaluated, or the number of linear solves required if the method is implicit. Once the stages are computed, computing y_{n+1} and $\hat{y}_{n+1}$ simply requires taking different linear combinations of the stages.

Multirate infinitesimal methods are not guaranteed to have this final step of simply combining the previously computed stages to form the solution as they use the last stage as the solution for the step. To efficiently compute error estimates, multirate infinitesimal methods use "last stage embeddings," which, as the name suggests, use alternate coefficients for the last stage to compute a solution with an alternate order of accuracy. Because of this, embeddings are not as computationally cheap to compute as in Runge–Kutta methods, but are still far cheaper to use than computing a step with two completely separate methods. Section 2.5 contains a discussion on designing multirate infinitesimal methods (specifically IMEX-MRI-GARK methods) to include cheaply computable embeddings.

2.2. Control theoretic derivation of time step controllers

In this section, we discuss control theoretic approaches to derive time step controllers. The process starts with the analytical error approximation formula (1.14). Controllers are developed by imposing approximations onto the leading error function $\phi(t)$. As mentioned in Section 1.6, the I-controller (1.9) is formed by the approximation that $\phi(t)$ is constant, that is, $\widehat{\phi}_n = \phi_{n-1}$, where $\widehat{\phi}_n$ is the approximated value of ϕ_n.

Time step controllers are only informed by the duration of their history–the elementary and I-controllers are only informed by the previous time step and the PID-controller (1.10) is only informed by the previous three time steps. The approximation of constant $\phi(t)$ is

actually an approximation of *piecewise* constant $\phi(t)$. This is due to each approximation $\widehat{\phi}_n$ only depending on the single value ϕ_{n-1}, which can change from step to step because of the approximate nature of the error estimate ε_n.

If we take the analytical error approximation formula (1.14) and evaluate it at the previous time step t_{n-1}, we can solve for ϕ_{n-1},

$$\phi_{n-1} = \varepsilon_n h_{n-1}^{-p} \tag{2.2}$$

We can then enforce the elementary controller's approximation of $\widehat{\phi}_n = \phi_{n-1}$ by substituting (2.2) into ϕ_n in (1.14).

$$\varepsilon_{n+1} = \varepsilon_n h_{n-1}^{-p} h_n^p \tag{2.3}$$

If we want to set the value of h_n such that $\varepsilon_{n+1} = \text{TOL}$ given the previous error ε_n and the previous time step h_{n-1}, we can replace ε_{n+1} with TOL in (2.3) and solve for h_n. This results in the elementary time step controller (1.8) after shifting the indices up one.

It it common to take the log function of the analytical error approximation formula to convert from a multiplicative formula with exponents to a simpler additive formula with coefficients,

$$\log(\varepsilon_n) \approx \log(\phi_n) + p \log(h_n). \tag{2.4}$$

It is easier to derive a time step update function in log-space and convert back to the expected form at the end by taking the exponential of both sides.

The I-controller (1.9) approximates $\phi(t)$ as piecewise constant with an introduced correction term incorporating a corresponding free parameter k_1,

$$\log(\widehat{\phi}_n) = \log(\widehat{\phi}_{n-1}) + k_1(\log(\phi_{n-1}) - \log(\widehat{\phi}_{n-1})) \tag{2.5}$$

This approximation acknowledges that the previous step likely incorporated error in its own approximation of $\phi(t)$. This correction term is called an "observer" [7, Chapters 11,13] and includes information about the difference between the true value of ϕ_{n-1}, which is only measurable after a step has been taken, and its approximation $\widehat{\phi}_{n-1}$. When $k_1 = 1$, this reduces to the elementary controller's approximation of $\phi(t)$. While it may seem that $k_1 = 1$ is the most natural, effective choice for the free parameter value, a lesser value is often more effective at regulating the step size change in practice.

Introducing the shift parameter q, common in control theory [7, 54, 55, 56, 15] and defined such that $q^a \log(\widehat{\phi}_n) = \log(\widehat{\phi}_{n+a})$ for arbitrary temporal shift a, into (2.5),

$$\log(\widehat{\phi}_n) = q^{-1}\log(\widehat{\phi}_n) + k_1(\log(\phi_{n-1}) - q^{-1}\log(\widehat{\phi}_n)), \tag{2.6a}$$

$$= (1-k_1)q^{-1}\log(\widehat{\phi}_n) + k_1\log(\phi_{n-1}), \tag{2.6b}$$

$$= \frac{k_1 q}{q + q^0(1-k_1)}\log(\phi_{n-1}), \tag{2.6c}$$

leads to an alternative piecewise constant approximation formulation for $\phi(t)$ which includes the observer correction term. The I-controller is then created by substituting (2.2) into (2.6c), then (2.6c) into (1.14), setting $\varepsilon_{n+1} = \text{TOL}$ as above, solving for h_n, and shifting the indices up one.

Gustafsson's controller (1.16) similarly includes this observer correction term in the approximation of $\phi(t)$ but uses a piecewise linear approximation rather than piecewise constant [16]. This approximation takes the following form, introducing a gradient correction to the piecewise constant approximation.

$$\log(\widehat{\phi}_n) = \log(\widehat{\phi}_{n-1}) + \nabla\log(\widehat{\phi}_{n-1}) + k_1(\log(\phi_{n-1}) - \log(\widehat{\phi}_{n-1})), \tag{2.7a}$$

$$\nabla\log(\widehat{\phi}_n) = \nabla\log(\widehat{\phi}_{n-1}) + k_2(\log(\phi_{n-1}) - \log(\widehat{\phi}_{n-1})). \tag{2.7b}$$

Chapter 3 contains more detail on how this approximation is used to derive a controller.

2.3. Generalized-structure additive Runge–Kutta theory

In this section, we summarize the main theoretical results for generalized-structure additive Runge–Kutta (GARK) [46] methods upon which many multirate infinitesimal methods rely. GARK methods can be derived to solve an IVP with arbitrarily many partitions (1.3). A GARK method computes N sets of s stages. This value of N can be less than or equal to the number of partitions n.

In the context of a multirate IVP, we compute $N = 2$ sets of stages–a set for the fast dynamics and a set for the slow dynamics. For a two-partition multirate IVP (1.4), N is equal to the number of partitions. For the three-partition multirate IVP (1.5), N is less than the number of partitions.

We focus on the case of a GARK method with two sets of stages (fast and slow) applied to the above three-partition problem. One step of this GARK method is computed as follows,

$$Y_i^{\{F\}} = y_n + H \sum_{j=1}^{s^{\{F\}}} a_{i,j}^{\{F,F\}} f^{\{F\}}(Y_j^{\{F\}}) + H \sum_{j=1}^{s^{\{I\}}} a_{i,j}^{\{F,I\}} f^{\{I\}}(Y_j^{\{S\}}) + H \sum_{j=1}^{s^{\{E\}}} a_{i,j}^{\{F,E\}} f^{\{E\}}(Y_j^{\{S\}}), \tag{2.8a}$$

$$Y_i^{\{S\}} = y_n + H \sum_{j=1}^{s^{\{F\}}} a_{i,j}^{\{S,F\}} f^{\{F\}}(Y_j^{\{F\}}) + H \sum_{j=1}^{s^{\{I\}}} a_{i,j}^{\{S,I\}} f^{\{I\}}(Y_j^{\{S\}}) + H \sum_{j=1}^{s^{\{E\}}} a_{i,j}^{\{S,E\}} f^{\{E\}}(Y_j^{\{S\}}), \tag{2.8b}$$

$$y_{n+1} = y_n + H \sum_{j=1}^{s^{\{F\}}} \mathbf{b}_j^{\{F\}} f^{\{F\}}(Y_j^{\{F\}}) + H \sum_{j=1}^{s^{\{I\}}} \mathbf{b}_j^{\{I\}} f^{\{I\}}(Y_j^{\{S\}}) + H \sum_{j=1}^{s^{\{E\}}} \mathbf{b}_j^{\{E\}} f^{\{E\}}(Y_j^{\{S\}}). \tag{2.8c}$$

Note that this formulation is in *autonomous* form, with no explicit dependence on time t. Any IVP with explicit dependence on t can be converted to autonomous form by appending

t to the end of the solution and stage vectors, and 1 to the end of the right-hand side function vector and replacing t in the functions with the corresponding new entry of the solution and state vectors. Thus, this representation is valid for all IVPs.

These sets of coefficients $\mathbf{A}^{\{\sigma,\nu\}}, \mathbf{b}^{\{\sigma\}}$, $\sigma \in \{F, S\}$, $\nu \in \{F, I, E\}$ can be written in the form of an extended Butcher tableau.

$$\begin{array}{c|c|c} \mathbf{A}^{\{F,F\}} & \mathbf{A}^{\{F,E\}} & \mathbf{A}^{\{F,I\}} \\ \hline \mathbf{A}^{\{S,F\}} & \mathbf{A}^{\{S,E\}} & \mathbf{A}^{\{S,I\}} \\ \hline \mathbf{b}^{\{F\},T} & \mathbf{b}^{\{E\},T} & \mathbf{b}^{\{I\},T} \end{array} .$$

The matrices $\mathbf{A}^{\{\sigma,\nu\}}$ relate the set of stages $Y^{\{\sigma\}}$ to the partition $f^{\{\nu\}}$.

We note that, while this GARK method structure does not imply a multirate structure, multirate infinitesimal methods can be represented with this structure. The benefit of analyzing a multirate infinitesimal method as if it were a GARK method is that order conditions are inherited.

The GARK order conditions are derived from standard Runge–Kutta theory combined with the N-tree theory of Araujo et al. [1]. A GARK method is called "internally consistent" if the following conditions hold:

$$\mathbf{c}^{\{F\}} = \mathbf{c}^{\{F,F\}} = \mathbf{c}^{\{F,I\}} = \mathbf{c}^{\{F,E\}}, \tag{2.9a}$$

$$\mathbf{c}^{\{S\}} = \mathbf{c}^{\{S,F\}} = \mathbf{c}^{\{S,I\}} = \mathbf{c}^{\{S,E\}}. \tag{2.9b}$$

where $\mathbf{c}^{\{\sigma,\nu\}} = \mathbf{A}^{\{\sigma,\nu\}}\mathbb{1}^{\{\nu\}}$. The following are the conditions guaranteeing up to fourth-order accuracy for a GARK method with the structure described above, assuming internal

consistency conditions hold, represented in matrix-vector form:

$$\mathbf{b}^{\{\sigma\},T}\mathbb{1}^{\{\sigma\}} = 1, \qquad \text{(order 1)} \tag{2.10a}$$

$$\mathbf{b}^{\{\sigma\},T}\mathbf{c}^{\{J(\nu)\}} = \frac{1}{2}, \qquad \text{(order 2)} \tag{2.10b}$$

$$\mathbf{b}^{\{\sigma\},T}(\mathbf{c}^{\{J(\nu)\}} \times \mathbf{c}^{\{J(\mu)\}}) = \frac{1}{3}, \qquad \text{(order 3)} \tag{2.10c}$$

$$\mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{J(\sigma),\nu\}}\mathbf{c}^{\{J(\nu)\}} = \frac{1}{3}, \qquad \text{(order 3)} \tag{2.10d}$$

$$\mathbf{b}^{\{\sigma\},T}(\mathbf{c}^{\{J(\nu)\}} \times \mathbf{c}^{\{J(\nu)\}} \times \mathbf{c}^{\{J(\mu)\}}) = \frac{1}{4}, \qquad \text{(order 4)} \tag{2.10e}$$

$$(\mathbf{b}^{\{\sigma\}} \times \mathbf{c}^{\{J(\nu)\}})^T\mathbf{A}^{\{J(\nu),\mu\}}\mathbf{c}^{\{J(\mu)\}} = \frac{1}{8}, \qquad \text{(order 4)} \tag{2.10f}$$

$$\mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{J(\nu),\mu\}}(\mathbf{c}^{\{J(\mu)\}} \times \mathbf{c}^{\{J(\lambda)\}}) = \frac{1}{12}, \qquad \text{(order 4)} \tag{2.10g}$$

$$\mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{J(\nu),\mu\}}\mathbf{A}^{\{J(\mu),\lambda\}}\mathbf{c}^{\{J(\lambda)\}} = \frac{1}{24}, \qquad \text{(order 4)} \tag{2.10h}$$

for $\sigma, \nu, \mu, \lambda \in \{F, I, E\}$. Here, $\mathbb{1}^{\{\sigma\}}$ is an $s^{\{\sigma\}}$-length vector of ones, $a \times b$ denotes element-wise multiplication of vectors, and J is a function which maps function partitions to sets of stages. In this case,

$$J(F) = F, \; J(I) = J(E) = S$$

A GARK method must satisfy all conditions through the desired order of accuracy to achieve that order.

2.3.1. Additive Runge–Kutta Methods

If only a single set of stages is used across all partitions of an IVP with arbitrarily many partitions (1.3), a GARK method reduces to an additive Runge–Kutta (ARK) method.

$$Y_i = y_n + H \sum_{\nu=1}^{n} \sum_{j=1}^{s} a_{i,j}^{\{\nu\}} f^{\{\nu\}}(Y_j), \tag{2.11a}$$

$$y_{n+1} = y_n + H \sum_{\nu=1}^{n} \sum_{i=1}^{s} b_{i}^{\{\nu\}} f^{\{\nu\}}(Y_i). \tag{2.11b}$$

This is again represented in autonomous form. An ARK method boils down to n separate classical Runge–Kutta methods, $(A^{\{\nu\}}, b^{\{\nu\}}, c^{\{\nu\}})$ working together. More detail on ARK methods can be found in [3, 6, 23, 26]. In particular, the work done by Kennedy and Carpenter is quite thorough.

A main strength of ARK methods is their ability to form IMEX structures, where the IVP is partitioned into a nonstiff partition, handled explicitly, and a stiff partition, handled implicitly. This is by the nonstiff partition's corresponding A matrix, $A^{\{E\}}$, having a strictly lower-triangular structure (thus, only using already-computed data) and the stiff partition's corresponding A matrix, $A^{\{I\}}$, having a lower-triangular structure (forming the common diagonally-implicit structure where each stage requires an implicit solve) or an arbitrary structure (forming the uncommon and more expensive fully-implicit structure where all stages must be solved implicitly simultaneously). This is useful in common multiphysics problems, described in Chapter 1.

2.4. Implicit-explicit multirate infinitesimal GARK theory

Implicit-explicit multirate infinitesimal GARK (IMEX-MRI-GARK) methods [5], developed by Chinomona and Reynolds in 2022, are multirate infinitesimal methods which solve a three-partition IVP (1.5). They are extensions to MRI-GARK methods [45], developed by Sandu in 2019. If $f^{\{I\}} = 0$, an IMEX-MRI-GARK method reduces to a fully explicit

MRI-GARK method. If $f^{\{E\}} = 0$, an IMEX-MRI-GARK method reduces to an implicit MRI-GARK method. Thus, analyzing IMEX-MRI-GARK methods incorporates the analysis of MRI-GARK methods.

2.4.1. Method definition

One step of an IMEX-MRI-GARK method is defined as follows,

$$\text{Let: } Y_1^{\{S\}} := y_n \tag{2.12a}$$

$$\text{For: } i = 2, ..., s^{\{S\}} \tag{2.12b}$$

$$\begin{cases} \text{Let:} & v(0) := Y_{i-1}^{\{S\}},\ T_{i-1} := t_n + c_{i-1}^{\{S\}} H,\ \Delta c_i^{\{S\}} := c_i^{\{S\}} - c_{i-1}^{\{S\}} \\ \text{Solve:} & v_i'(\theta) = \Delta c_i^{\{S\}} f^{\{F\}} \left(T_{i-1} + \Delta c_i^{\{S\}} \theta, v_i(\theta) \right) + g_i(\theta), \\ & \text{for } \theta \in [0, H] \\ & \text{where } g_i(\theta) = \sum_{j=1}^{i} \gamma_{i,j} \left(\frac{\theta}{H} \right) f_j^{\{I\}} + \sum_{j=1}^{i-1} \omega_{i,j} \left(\frac{\theta}{H} \right) f_j^{\{E\}} \\ \text{Let:} & Y_i^{\{S\}} := v_i(H), \end{cases} \tag{2.12c}$$

$$\text{Let: } y_{n+1} := Y_{s^{\{S\}}}^{\{S\}}. \tag{2.12d}$$

$$\begin{cases} \text{Let:} & \tilde{v}(0) := Y_{s^{\{S\}}-1}^{\{S\}},\ T_{s-1} := t_n + c_{s^{\{S\}}-1}^{\{S\}} H,\ \Delta c_{s^{\{S\}}}^{\{S\}} := c_{s^{\{S\}}}^{\{S\}} - c_{s^{\{S\}}-1}^{\{S\}} \\ \text{Solve:} & v_i'(\theta) = \Delta c_i^{\{S\}} f^{\{F\}} \left(T_{s-1} + \Delta c_{s^{\{S\}}}^{\{S\}} \theta, \tilde{v}_i(\theta) \right) + \tilde{g}_{s^{\{S\}}}(\theta), \\ & \text{for } \theta \in [0, H] \\ & \text{where } \tilde{g}_{s^{\{S\}}}(\theta) = \sum_{j=1}^{s^{\{S\}}} \tilde{\gamma}_j \left(\frac{\theta}{H} \right) f_j^{\{I\}} + \sum_{j=1}^{s^{\{S\}}-1} \tilde{\omega}_j \left(\frac{\theta}{H} \right) f_j^{\{E\}} \\ \text{Let:} & \tilde{y}_{n+1} := \tilde{v}(H). \end{cases} \tag{2.12e}$$

The step (2.12e) computes the embedding solution and is not required in fixed-step settings.

An IMEX-MRI-GARK method is uniquely defined by the number of stages $s^{\{S\}}$, abcissae vector $c^{\{S\}}$, n_Γ matrices of coefficients $\{\Gamma^{\{k\}}\}_{k=0}^{n_\Gamma-1}$, and n_Ω matrices of coefficients $\{\Omega^{\{k\}}\}_{k=0}^{n_\Omega-1}$. The coefficient functions $\gamma_{i,j}(\tau)$ and $\omega_{i,j}(\tau)$ are given by

$$\gamma_{i,j}(\tau) = \sum_{k=0}^{n_\Gamma-1} \gamma_{i,j}^{\{k\}} \tau^k, \tag{2.13a}$$

$$\omega_{i,j}(\tau) = \sum_{k=0}^{n_\Omega-1} \omega_{i,j}^{\{k\}} \tau^k. \tag{2.13b}$$

The embedding coefficient functions $\tilde{\gamma}(\tau)$ and $\tilde{\omega}(\tau)$ are defined similarly and the sets of embedding coefficient vectors $\{\tilde{\gamma}^{\{k\}}\}_{k=0}^{n_\Gamma-1}$ and $\{\tilde{\omega}^{\{k\}}\}_{k=0}^{n_\Omega-1}$ serve to further uniquely define the method.

For computational simplicity, either $\Delta c_i^{\{S\}}$ or $\overline{\gamma}_{i,i}$ may be nonzero for a given stage, but not both. When $\Delta c_i^{\{S\}} = 0$, the stage reduces to a simple ARK-like stage,

$$Y_i^{\{S\}} = Y_{i-1}^{\{S\}} + H \sum_{j=1}^{i} \overline{\gamma}_{i,j} f_j^{\{I\}} + H \sum_{j=1}^{i-1} \overline{\omega}_{i,j} f_j^{\{E\}} \tag{2.14}$$

where

$$\overline{\gamma}_{i,j} = \int_0^1 \gamma_{i,j}(\tau) \mathrm{d}\tau = \sum_{k=0}^{n_\Gamma-1} \frac{\gamma_{i,j}^{\{k\}}}{k+1}, \tag{2.15a}$$

$$\overline{\omega}_{i,j} = \int_0^1 \omega_{i,j}(\tau) \mathrm{d}\tau = \sum_{k=0}^{n_\Omega-1} \frac{\omega_{i,j}^{\{k\}}}{k+1}. \tag{2.15b}$$

This leads to a "solve-decoupled" structure in which each stage is computed via *either* a fast IVP solve *or* an ARK-like stage which may require an implicit solve via Newton-Raphson or some other root-finding method. If both $\Delta c_i^{\{S\}} = 0$ and $\overline{\gamma}_{i,i} = 0$, the stage is a simple explicit ARK stage.

The slow dynamics $f^{\{I\}}$ and $f^{\{E\}}$ are only evaluated once per stage and the resulting values are used as coefficients for the polynomial-in-time forcing function $g_i(\theta)$.

We note that θ in the fast IVPs is a sort of "stretched" time variable. Examining the fast IVP we can see that, despite each fast IVP solving for $\theta \in [0, H]$, the "real" time interval being solved over is only $t \in [t_n + c_{i-1}H, t_n + c_i H]$. Thus, each fast IVP starts where the previous one finished. This minimizes the total fast integration to evolve over an interval of size $\Delta c_i H$ per stage for a total duration of H per step. This is in contrast to MERK methods, discussed in Section 2.6, which typically integrate a total duration larger than H.

Because each stage integrates from $t_n + c_{i-1}H$ to $t_n + c_i H$, the abcissae vector $c^{\{S\}}$ must be non-decreasing to avoid stability issues.

2.4.2. Order conditions

Order conditions are derived by assuming the fast IVPs are solved with one step of a Runge–Kutta method defined by $A^{\{F\}}, b^{\{F\}}, c^{\{F\}}$. We note that without loss of generality, any subcycled one-step method may be expressed as a Runge–Kutta method with a sufficiently large number of stages, and thus this is a reasonable assumption. With this "fast method," one step of an IMEX-MRI-GARK method can be formulated as a single step of a GARK method (2.8a). The GARK coefficients are identified as the following:

$$\mathbf{A}^{\{F,F\}} = \operatorname{diag}(\Delta c^{\{S\}}) \otimes A^{\{F\}} + L\Delta C^{\{S\}} \otimes \mathbb{1}^{\{F\}} b^{\{F\},T} \in \mathbb{R}^{(s^{\{S\}}\cdot s^{\{F\}})\times(s^{\{S\}}\cdot s^{\{F\}})}, \tag{2.16a}$$

$$\mathbf{A}^{\{F,I\}} = LA^{\{I\}} \otimes \mathbb{1}^{\{F\}} + \sum_{k=0}^{n_\Gamma - 1} \Gamma^{\{k\}} \otimes (A^{\{F\}} c^{\{F\}\times k}) \in \mathbb{R}^{(s^{\{S\}}\cdot s^{\{F\}})\times s^{\{S\}}}, \tag{2.16b}$$

$$\mathbf{A}^{\{F,E\}} = LA^{\{E\}} \otimes \mathbb{1}^{\{F\}} + \sum_{k=0}^{n_\Gamma - 1} \Omega^{\{k\}} \otimes (A^{\{F\}} c^{\{F\}\times k}) \in \mathbb{R}^{(s^{\{S\}}\cdot s^{\{F\}})\times s^{\{S\}}}, \tag{2.16c}$$

$$\mathbf{A}^{\{S,F\}} = \Delta C^{\{S\}} \otimes b^{\{F\},T} \in \mathbb{R}^{s^{\{S\}}\times(s^{\{S\}}\cdot s^{\{F\}})}, \tag{2.16d}$$

$$\mathbf{A}^{\{S,I\}} = E\overline{\Gamma} \in \mathbb{R}^{s^{\{S\}}\times s^{\{S\}}}, \tag{2.16e}$$

$$\mathbf{A}^{\{S,E\}} = E\overline{\Omega} \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}}, \tag{2.16f}$$

$$\mathbf{b}^{\{F\}} = \Delta c^{\{S\}} \otimes b^{\{F\}} \in \mathbb{R}^{s^{\{S\}} \cdot s^{\{F\}}}, \tag{2.16g}$$

$$\mathbf{b}^{\{I\},T} = \mathbb{1}^{\{S\},T}\overline{\Gamma} \in \mathbb{R}^{s^{\{S\}}}, \tag{2.16h}$$

$$\mathbf{b}^{\{E\},T} = \mathbb{1}^{\{S\},T}\overline{\Omega} \in \mathbb{R}^{s^{\{S\}}}, \tag{2.16i}$$

where

$$\Delta c_i^{\{S\}} = \begin{cases} 0 & i = 0 \\ c_i^{\{S\}} - c_{i-1}^{\{S\}} & i = 2, ..., s^{\{S\}} \end{cases}, \; E \in \mathbb{R}^{s^{\{S\}}},$$

$$\Delta C^{\{S\}} = \begin{bmatrix} \Delta c_1^{\{S\}} & 0 & \cdots & 0 \\ \Delta c_1^{\{S\}} & \Delta c_2^{\{S\}} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \Delta c_1^{\{S\}} & \Delta c_2^{\{S\}} & \cdots & \Delta c_{s^{\{S\}}}^{\{S\}}, \end{bmatrix} \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}},$$

$$E_{i,j} = \begin{cases} 1 & i \geq j \\ 0 & else \end{cases}, \; E \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}}, \; L_{i,j} = \begin{cases} 1 & i = j+1 \\ 0 & else \end{cases}, \; L \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}},$$

$$A^{\{I\}} = \overline{\Gamma} = \sum_{k=0}^{n_\Gamma - 1} \frac{\Gamma^{\{k\}}}{k+1} \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}}, \; A^{\{E\}} = \overline{\Omega} = \sum_{k=0}^{n_\Omega - 1} \frac{\Omega^{\{k\}}}{k+1} \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}},$$

and $\otimes$ is the Kronecker product. We can insert these GARK coefficients into the GARK order conditions to get simplified order conditions.

An IMEX-MRI-GARK method reduces to its corresponding base ARK method, defined by $(A^{\{I\}}, b^{\{I\}}, A^{\{E\}}, b^{\{E\}})$, where $b^{\{I\}} = \mathbf{b}^{\{I\}}, b^{\{E\}} = \mathbf{b}^{\{E\}}$, when $\Delta c_i = 0$ or if $f^{\{F\}} = 0$. The stage is solved as defined in (2.14).

When the base ARK method, defined by $(A^{\{I\}}, b^{\{I\}}, A^{\{E\}}, b^{\{E\}})$, where $b^{\{I\}} = \mathbf{b}^{\{I\}}, b^{\{E\}} = \mathbf{b}^{\{E\}}$, is at least first order accurate, the first-order GARK conditions are automatically satisfied.

When the base ARK method is at least second-order accurate and internal consistency conditions are satisfied, the second-order GARK conditions are automatically satisfied. The internal consistency conditions for an IMEX-MRI-GARK method are:

$$\Gamma^{\{0\}} \mathbb{1}^{s^{\{S\}}} = \Omega^{\{0\}} \mathbb{1}^{\{S\}} = \Delta c^{\{S\}}, \tag{2.17a}$$

$$\Gamma^{\{k\}} \mathbb{1}^{\{S\}} = \Omega^{\{k\}} \mathbb{1}^{\{S\}} = 0^{\{S\}}, \ k \geq 1. \tag{2.17b}$$

When the base ARK method is at least third-order accurate and the fast method is order $\max(3, n_\Gamma + 1, n_\Omega + 1)$ accurate, and the IMEX-MRI-GARK method satisfies internal consistency conditions, there are two additional coupling conditions for third-order accuracy:

$$\Delta c^{\{S\},T} \mathcal{A}^{\{I,\zeta\}} c^{\{S\}} = \frac{1}{6}, \tag{2.18a}$$

$$\Delta c^{\{S\},T} \mathcal{A}^{\{E,\zeta\}} c^{\{S\}} = \frac{1}{6}, \tag{2.18b}$$

where

$$\mathcal{A}^{\{I,\zeta\}} = L A^{\{I\}} + \sum_{k=0}^{n_\Gamma - 1} \zeta_k \Gamma^{\{k\}},$$

$$\mathcal{A}^{\{E,\zeta\}} = L A^{\{E\}} + \sum_{k=0}^{n_\Omega - 1} \zeta_k \Omega^{\{k\}},$$

$$\zeta_k = b^{\{F\},T} A^{\{F\}} c^{\{F\}\times k}.$$

When the base ARK method is at least fourth-order accurate and the fast method is order $\max(4, n_\Gamma + 2, n_\Omega + 2)$ accurate, and the IMEX-MRI-GARK method satisfies internal consistency and third-order accuracy coupling conditions, there are sixteen additional coupling conditions for fourth-order accuracy:

$$\left(\Delta c^{\{S\}} \times L c^{\{S\}}\right)^T \mathcal{A}^{\{\sigma,\zeta\}} + \left(\Delta c^{\{S\}\times 2}\right)^T \mathcal{A}^{\{\sigma,\beta\}} c^{\{S\}} = \frac{1}{8}, \tag{2.19a}$$

$$\Delta c^{\{S\}} \mathcal{A}^{\{\sigma,\xi\}} c^{\{S\}\times 2} = \frac{1}{12}, \tag{2.19b}$$

$$\left(\Delta c^{\{S\}} \times (D b^{\{\sigma\}})\right)^T \mathcal{A}^{\{\nu,\zeta\}} c^{\{S\}} = \frac{1}{24}, \tag{2.19c}$$

$$\left(\Delta c^{\{S\}\times 2}\right)^T \mathcal{A}^{\{\sigma,\xi\}} c^{\{S\}} + \Delta c^{\{S\},T} L \Delta C^{\{S\}} \mathcal{A}^{\{\sigma,\zeta\}} c^{\{S\}} = \frac{1}{24}, \tag{2.19d}$$

$$\Delta c^{\{S\},T} \mathcal{A}^{\{\sigma,\zeta\}} A^{\{\nu\}} c^{\{S\}} = \frac{1}{24}, \tag{2.19e}$$

for $\sigma, \nu \in \{I, E\}$, where

$$\mathcal{A}^{\{I,\beta\}} = \frac{1}{2} L A^{\{I\}} + \sum_{k=0}^{n_\Gamma - 1} \beta_k \Gamma^{\{k\}}, \; \mathcal{A}^{\{E,\beta\}} = \frac{1}{2} L A^{\{E\}} + \sum_{k=0}^{n_\Omega - 1} \beta_k \Omega^{\{k\}},$$

$$\mathcal{A}^{\{I,\xi\}} = \frac{1}{2} L A^{\{I\}} + \sum_{k=0}^{n_\Gamma - 1} \xi_k \Gamma^{\{k\}}, \; \mathcal{A}^{\{E,\xi\}} = \frac{1}{2} L A^{\{E\}} + \sum_{k=0}^{n_\Omega - 1} \xi_k \Omega^{\{k\}},$$

$$\beta_k = (b^{\{F\}} \times c^{\{F\}})^T A^{\{F\}} c^{\{F\}\times k}, \; \xi_k = b^{\{F\},T} A^{\{F\}} A^{\{F\}} c^{\{F\}\times k}.$$

When using an existing ARK method as a base method when deriving a new IMEX-MRI-GARK method, the above listed coupling conditions are all that must be enforced. However, if simply computing the base method from the IMEX-MRI-GARK coefficients in

the derivation of a new method, the base ARK method's order conditions must also be enforced.

2.4.3. Linear stability

Linear stability analysis centers on the long-term behavior of time integration algorithms and the tendency for approximation errors to grow to infinity, regardless of the true solution to the IVP, when time step sizes grow too large. The standard means of linear stability analysis in Runge–Kutta methods is the Dahlquist test problem

$$y'(t) = \lambda y(t),\ y(t_0) = y_0,\ \lambda \in \mathbb{C}^- \tag{2.20}$$

where $\mathbb{C}^-$ is the negative real half of the complex plane. This IVP has the true solution $y(t) = y_0 e^{\lambda t}$. Attention is focused on $\lambda \in \mathbb{C}^-$ as the true solution decays from the initial condition and any growth of the solution to infinity is an artifact of the time integration algorithm.

A goal of linear stability analysis is the investigation of a method's growth function $R(h\lambda)$, which is derived through the application of the Runge–Kutta method to the IVP (2.20), arriving at the formula $y_{n+1} = R(h\lambda)y_n$. A method's "stability region" is defined as

$$\{z \in \mathbb{C}^- : |R(z)| < 1\}, \tag{2.21}$$

where $z = h\lambda$ and λ is the eigenvalue of the problem. This is the region of stepsize scaled eigenvalues in which the solution to the test problem (2.20) decays as expected.

Linear stability analysis of IMEX multirate methods generally focuses on a partitioned version of (2.20),

$$y'(t) = \lambda^{\{F\}} y(t) + \lambda^{\{I\}} y(t) + \lambda^{\{E\}} y(t),\ y(t_0) = y_0,\ \lambda^{\{F\}}, \lambda^{\{I\}}, \lambda^{\{E\}} \in \mathbb{C}^-, \tag{2.22}$$

The analysis of multirate infinitesimal linear stability assumes the fast IVPs are solved exactly to consider strictly the multirate method's effects on stability. As usual, the goal is to derive and investigate a growth function. For IMEX multirate methods, this growth function takes the form $R(z^{\{F\}}, z^{\{I\}}, z^{\{E\}})$, where $z^{\{F\}} = H\lambda^{\{F\}}, z^{\{I\}} = H\lambda^{\{I\}}, z^{\{E\}} = H\lambda^{\{E\}}$ are the stepsize scaled eigenvalues corresponding to the various partitions of the right-hand side function.

The growth function for IMEX-MRI-GARK methods is given by

$$R(z^{\{F\}},z^{\{I\}}, z^{\{E\}}) = \tag{2.23}$$

$$e_{s^{\{S\}}}^T \left(I - \operatorname{diag}\left(\varphi_0(\Delta c^{\{S\}} z^{\{F\}})\right) L - z^{\{E\}}\eta(z^{\{F\}}) - z^{\{I\}}\mu(z^{\{F\}})\right)^{-1} e_1, \tag{2.24}$$

where

$$(e_i)_j = \begin{cases} 1 & i = j \\ 0 & else \end{cases}, \ e_i \in \mathbb{R}^{s^{\{S\}}}$$

$$\varphi_0(z) = e^z, \ \varphi_k(z) = \frac{k\varphi_k(z) - 1}{z}, k \geq 1$$

$$\eta(z^{\{F\}}) = \sum_{k=0}^{n_\Omega - 1} \operatorname{diag}\left(\varphi_{k+1}(\Delta c^{\{S\}} z^{\{F\}})\right) \Omega^{\{k\}}$$

$$\mu(z^{\{F\}}) = \sum_{k=0}^{n_\Gamma - 1} \operatorname{diag}\left(\varphi_{k+1}(\Delta c^{\{S\}} z^{\{F\}})\right) \Gamma^{\{k\}}$$

Chinomona and Reynolds [5] extend the definition of "joint stability" from [63], originally devised for IMEX two-step Runge–Kutta methods, into the multirate setting. While the classical definition of stability considers whether the method is stable for a given value of z, this definition considers whether the method is stable for a given value of $z^{\{E\}}$ when $z^{\{I\}}$ and $z^{\{F\}}$ take on any value in some infinite wedge centered on the negative real axis of the

complex plane. This joint stability region is defined as

$$\mathcal{J}_{\alpha,\beta} = \{z^{\{E\}} \in \mathbb{C}^- : |R(z^{\{F\}}, z^{\{I\}}, z^{\{E\}})| < 1,\ \forall z^{\{F\}} \in \mathcal{S}_\alpha^{\{F\}},\ \forall z^{\{I\}} \in \mathcal{S}_\beta^{\{I\}}\}, \tag{2.25}$$

where

$$\mathcal{S}_\alpha^{\{\sigma\}} = \{z^{\{\sigma\}} \in \mathbb{C}^- : |\arg(z^{\{\sigma\}}) - \pi| \leq \alpha\}. \tag{2.26}$$

This definition of stability can be over-restrictive. In real-world applications, the implicit and fast eigenvalues are not infinitely more stiff than the explicit eigenvalue. This definition gives a region of $z^{\{E\}}$ which produce stable solutions for any value of $z^{\{I\}}$ and $z^{\{F\}}$ in the given wedge, but does not give the region of all $z^{\{E\}}$ which may produce a stable solution for specific values of $z^{\{I\}}$ and $z^{\{F\}}$. Further considerations on the linear stability regions of IMEX multirate infinitesimal methods are contained in Chapter 4.

2.5. Design and derivation of IMEX-MRI-GARK methods

The first step in deriving a new IMEX-MRI-GARK method is to choose a few parameters including the number of stages $s^{\{S\}}$, the number of $\Gamma^{\{k\}}$ matrices n_Γ, and the number of $\Omega^{\{k\}}$ matrices n_Ω. The number of stages corresponds roughly to the total work required to complete a step, so it is good to keep this number small. It is common for simplicity to have $n_\Gamma = n_\Omega$ and to keep this number small since although a larger number of matrices can potentially achieve a higher order of accuracy for the same number of stages, it seems as though the stability of such methods can suffer.

When these parameters are chosen, the next step is to create $s^{\{S\}}$-by-$s^{\{S\}}$ matrices filled with undetermined coefficients. The Ω matrices should be strictly lower triangular, and the Γ matrices should be lower triangular with the first row being filled with zeros. This Γ structure exists because the first stage is explicitly set to the solution from the previous time step.

The next step is to decide whether to use an existing ARK method as the base method, or simply compute the base method from the IMEX-MRI-GARK coefficients. Using an existing ARK base method simplifies the process of solving the order conditions, but can be detrimental to the IMEX-MRI-GARK method's stability. In contrast, computing the ARK base methods complicates the process of solving the order conditions as the IMEX-MRI-GARK coupling conditions must be solved alongside the base ARK method's order conditions. These competing effects can be seen in Chinomona and Reynolds [5] and in Chapter 4.

2.5.1. Using an existing ARK base method

When using an existing ARK base method, the $A^{\{I\}}$ and $A^{\{E\}}$ matrices must be padded with the $b^{\{I\}}$ and $b^{\{E\}}$ vectors and a column of zeros to ensure that the ARK method has the first-same-as-last (FSAL) structure where the last stage is equal to the stage solution, if that property does not already exist in the ARK method. This padded ARK structure has the form

$$\begin{array}{r|cc|cc} c^{\{S\}} & A^{\{I\}} & 0^{s^{\{S\}}} & A^{\{E\}} & 0^{s^{\{S\}}} \\ 1 & b^{\{I\},T} & 0 & b^{\{E\},T} & 0 \\ \hline & b^{\{I\},T} & 0 & b^{\{E\},T} & 0 \end{array},$$

where $0^{s^{\{S\}}}$ is an $s^{\{S\}}$-length vector of zeros.

If deriving an embedded IMEX-MRI-GARK method from an existing embedded ARK method, the ARK method's matrices must be padded with the embedding $\hat{b}^{\{I\}}$ and $\hat{b}^{\{E\}}$ vectors when deriving the embedding coefficients.

As mentioned in Section 2.4.1, the abcissae vector $c^{\{S\}}$ must be non-decreasing. This is an uncommon feature in Runge–Kutta methods [3, 23, 24, 25] and limits the options of base methods, especially at fourth-order and higher.

Once the FSAL ARK method is in place, it must be further padded to introduce $c^{\{S\}}$ values such that $\Delta c_i^{\{S\}} = 0$. As mentioned in Section 2.4, stages in which $\Delta c_i^{\{S\}} = 0$ correspond to an ARK stage, the only stages which can require implicit solves in the simple and efficient "solve-decoupled" structure. To introduce these stages, new rows are interwoven into the existing ARK method with the same value of $c_i^{\{S\}}$ as the stage after. To keep the matrices square, new columns of all zeros are introduced at the same indices. For example, if padding the explicit portion of a three-stage ARK method that has already been padded with the $b^{\{E\}}$ vector,

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
c_2^{\{S\}} & a_{2,1}^{\{E\}} & 0 & 0 & 0 \\
c_3^{\{S\}} & a_{3,1}^{\{E\}} & a_{3,2}^{\{E\}} & 0 & 0 \\
1 & b_1^{\{E\}} & b_2^{\{E\}} & b_3^{\{E\}} & 0 \\
\hline
 & b_1^{\{E\}} & b_2^{\{E\}} & b_3^{\{E\}} & 0
\end{array}
\quad \rightarrow \quad
\begin{array}{c|cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_2^{\{S\}} & \overline{a}_{2,1}^{\{E\}} & 0 & 0 & 0 & 0 & 0 & 0 \\
c_2^{\{S\}} & a_{2,1}^{\{E\}} & 0 & 0 & 0 & 0 & 0 & 0 \\
c_3^{\{S\}} & \overline{a}_{3,1}^{\{E\}} & 0 & \overline{a}_{3,2}^{\{E\}} & 0 & 0 & 0 & 0 \\
c_3^{\{S\}} & a_{3,1}^{\{E\}} & 0 & a_{3,2}^{\{E\}} & 0 & 0 & 0 & 0 \\
1 & \overline{b}_1^{\{E\}} & 0 & \overline{b}_2^{\{E\}} & 0 & \overline{b}_3^{\{E\}} & 0 & 0 \\
1 & b_1^{\{E\}} & 0 & b_2^{\{E\}} & 0 & b_3^{\{E\}} & 0 & 0 \\
\hline
1 & b_1^{\{E\}} & 0 & b_2^{\{E\}} & 0 & b_3^{\{E\}} & 0 & 0
\end{array}
\,.
$$

The $\overline{a}_{i,j}^{\{E\}}$ and $\overline{b}_j$ coefficients are introduced to maintain the structure of the method. These coefficients have no effect if the ARK method padded in this way is used purely *as* an ARK method due to the new zero b values, thus the padded method retains its expected order of accuracy. These coefficients *do* have an effect on the behavior of the IMEX-MRI-GARK method which is derived from the padded ARK method.

The relations $A^{\{E\}} = \overline{\Omega}$ and $A^{\{I\}} = \overline{\Gamma}$ create $2s^{\{S\}}$ simple relations enforcing the IMEX-MRI-GARK's usage of the base method.

To complete the solve-decoupled structure, in rows corresponding to $\Delta c_i^{\{S\}} \neq 0$ the coefficients on the diagonal of the Γ matrices must be manually set to zero.

2.5.2. Computing the base method from the IMEX-MRI-GARK method

Another valid technique of handling the base method in deriving an IMEX-MRI-GARK method is to simply compute it in terms of the IMEX-MRI-GARK coefficients through the relations $A^{\{E\}} = \overline{\Omega}$ and $A^{\{I\}} = \overline{\Gamma}$. This gives an ARK method defined in terms of undetermined IMEX-MRI-GARK coefficients. The base method's order conditions must then be enforced up through the desired order of the IMEX-MRI-GARK method, along with the coupling conditions.

To achieve the solve-decoupled structure, the vector $c^{\{S\}}$ must be chosen such that there are repeated values, thereby introducing $\Delta c_i^{\{S\}} = 0$. In rows corresponding to $\Delta c_i^{\{S\}} \neq 0$ the coefficients on the diagonal of the Γ matrices must be manually set to zero.

2.5.3. Choosing a solve-decoupled structure

It is natural to introduce a $\Delta c_i = 0$ for every other stage, thus creating an alternating fast IVP solve-implicit correction pattern. However, several implicit corrections could be chained together, or several fast IVP solves could happen in a row. There is a balance to be struck between the increased work in adding implicit solve stages, considered costly for the increased amount of slow dynamics and Jacobian evaluations, and the increased stability from them.

As with classical Runge–Kutta methods (2.1), it is desirable to have an embedding be as cheap as possible to compute. The computationally cheapest stage would correspond to $\Delta c_i^{\{S\}} = 0$ and $\overline{\gamma}_{i,i} = 0$, a simple explicit ARK stage which requires no additional function evaluations.

2.5.4. Solving the order conditions

Once the base method and IMEX-MRI-GARK method have been appropriately set up, they can be plugged into the order conditions. All order conditions for the base method and the coupling conditions on the IMEX-MRI-GARK method must be satisfied up through the desired order of accuracy. The order conditions take the form of polynomials of the coefficients with potentially many terms, especially at fourth-order. These can be solved by hand or through the use of a computer algebra system.

Often, solving the order conditions leaves a few free variables which can be set to any value while the method retains its expected order of accuracy. There are a variety of ways to use these free variables. Some involve trying to decrease the expected error from a step by solving a subset of the order conditions one higher up, minimizing the norm of the residuals of the order conditions one higher up, etc. Others involve maximizing the size of the joint stability region by solving extra algebraic constraints or by setting up an expression to optimize.

2.6. Multirate exponential Runge–Kutta methods

Multirate exponential Runge–Kutta (MERK) methods [31] are explicit multirate infinitesimal methods which use Exponential Runge–Kutta (ExpRK) theory [21, 33, 59] rather than GARK theory as the foundation for derivation and analysis. MERK methods solve IVPs which are partitioned with linear fast dynamics and arbitrarily nonlinear slow dynamics,

$$y'(t) = f^{\{F\}}(t, y(t)) + f^{\{S\}}(t, y(t)) = \mathcal{L}y(t) + \mathcal{N}(t, y(t)) \tag{2.27}$$

where $\mathcal{L}$ is a constant scalar or matrix and $\mathcal{N}$ is an arbitrarily nonlinear function. This restriction on the linearity of $f^{\{F\}}$ is inherited from ExpRK methods.

One step of a MERK method is defined as follows.

$$\text{Let: } Y_1^{\{S\}} = y_n \tag{2.28a}$$

$$\text{For } i = 2, ..., s^{\{S\}} - 1 \tag{2.28b}$$

$$\begin{cases} \text{Let:} & v_i(0) = y_n \\ \text{Solve:} & v_i'(\tau) = \mathcal{L}v(\tau) + g_i(\tau), \text{ for } \tau \in [0, c_i H] \\ \text{where} & g_i(\tau) = \sum_{j=2}^{i-1} \gamma_{i,j} \widehat{D}_{n,j}, \ \gamma_{i,j} = \sum_{k=1}^{\ell_{i,j}} \frac{\alpha_{i,j}^{(k)}}{c_i^k H^{k-1}(k-1)!} \tau^{k-1}, \\ & \widehat{D}_{n,j} = \mathcal{N}(t_n + c_j H, Y_j) - \mathcal{N}(t_n, Y_1). \\ & \text{Let: } Y_i^{\{S\}} = v_i(H) \end{cases} \tag{2.28c}$$

$$\text{Let: } v_{s^{\{S\}}}(0) = y_n \tag{2.28d}$$

$$\text{Solve: } v'_{s^{\{S\}}}(\tau) = \mathcal{L}v_{s^{\{S\}}}(\tau) + g_{s^{\{S\}}}(\tau), \text{ for } \tau \in [0, H] \tag{2.28e}$$

$$\text{where } g_{s^{\{S\}}}(\tau) = \sum_{j=2}^{s^{\{S\}}-1} \gamma_{s^{\{S\}},j} \widehat{D}_{n,j}, \ \gamma_{s^{\{S\}},j} = \sum_{k=1}^{m_j} \frac{\beta_j^{(k)}}{c_i^k H^{k-1}(k-1)!} \tau^{k-1}. \tag{2.28f}$$

$$\text{Let: } y_{n+1} = v_{s^{\{S\}}}(H) \tag{2.28g}$$

The values of $\alpha_{i,j}^{(k)}, \beta_i^{(k)}, \ell_{i,j}, m_j$ come from the base ExpRK method. Luan et al. provide example MERK methods of orders two through five with the coefficients $\gamma_{i,j}$ computed and simplified. Like the general class of MRI-GARK methods, the slow dynamics $\mathcal{N}$ are only evaluated once per stage and used as coefficients for the polynomial-in-time forcing function $g_i(\tau)$. We further note that all MERK methods introduced by Luan et al. do not include embeddings.

While each stage of an MRI-GARK method evolves each fast IVP over an interval in "real" time of size $\Delta c_i H$, MERK methods evolve each fast IVP over an interval of size $c_i H$.

This leads to increased computational work per step but can have benefits in terms of stability and decreased error from the true solution to the multirate IVP (2.27) because each fast IVP contains more information in the forcing function $g_i(\theta)$. The increased computational work from solving fast IVPs over longer intervals is essentially negligible as the fast dynamics are assumed to be cheap to evaluate.

Because MERK methods start each stage i at the beginning of the time interval t_n and evolve the solution to $t_n + c_i^{\{S\}} H$, a MERK method can be defined by non-sorted abcissae vector $c^{\{S\}}$. As mentioned in Section 2.5, IMEX-MRI-GARK's restriction to non-decreasing abcissae vector (such that $c_i^{\{S\}} \geq c_{i-1}^{\{S\}}$) is quite limiting for the derivation of new methods, thus MERK methods have a greater freedom of choice in their coefficient values, potentially leading to more efficient methods.

Chapter 3

Adaptive time step control for multirate infinitesimal methods

The contents of this chapter have been accepted for publication to the SIAM Journal on Scientific Computing with the title "Adaptive time step control for multirate infinitesimal methods" in collaboration with Dr. Daniel Reynolds. Background theory Sections 2.1 and 2.2 are useful in understanding this work.

3.1. Introduction

Multirate methods are numerical time integration algorithms used to approximate solutions to ordinary differential equation (ODE) initial-value problems (IVP),

$$y'(t) = f(t,y) = f^{\{S\}}(t,y) + f^{\{F\}}(t,y), \quad y(0) = y_0,$$

in which some portion of the right-hand side function $f(t,y)$ evolves on a slower time scale (and can therefore be evaluated less frequently) than the rest of the function. Multirate methods typically split $f(t,y)$ additively into two parts, $f^{\{S\}}$, the *slow function*, which is evaluated less frequently, and $f^{\{F\}}$, the *fast function*, which is evaluated more frequently. For many large-scale applications, multirate methods become particularly attractive when $f^{\{S\}}$ has significantly larger computational cost than $f^{\{F\}}$, and thus a method that minimizes calls to $f^{\{S\}}$ may achieve significant gains in computational efficiency for achieving a desired solution accuracy.

A variety of families of multirate methods have been proposed, including MrGARK [14], MIS [49, 61], MRI-GARK [45], IMEX-MRI-GARK [5], MERK [31], MERB [32], and others. In this work, we focus on so-called *infinitesimal*-type methods, which include all of the above methods except MrGARK. Generally speaking, an infinitesimal method for evolving a single

time step $y_n \approx y(t_n)$ to $y_{n+1} \approx y(t_n + H)$, with its embedded solution $\tilde{y}_{n+1} \approx y(t_n + H)$, proceeds through the following sequence of steps.

1. Let: $Y_1 = y_n$
2. For $i = 2, ..., s$:
   (a) Solve: $v_i'(\theta) = C_i\, f^{\{F\}}(\theta, v_i(\theta)) + r_i(\theta)$, for $\theta \in [\theta_{0,i}, \theta_{F,i}]$ with $v_i(\theta_{0,i}) = v_{0,i}$.
   (b) Let: $Y_i = v_i(\theta_{F,i})$.
3. Solve: $\tilde{v}_s'(\theta) = C_s\, f^{\{F\}}(\theta, \tilde{v}_s(\theta)) + \tilde{r}_s(\theta)$, for $\theta \in [\theta_{0,s}, \theta_{F,s}]$ with $\tilde{v}_s(\theta_{0,s}) = v_{0,s}$.
4. Let: $y_{n+1} = Y_s$ and $\tilde{y}_{n+1} = \tilde{v}_s(\theta_{F,s})$.

In the above, the multirate method is uniquely defined by its choice of leading constant C_i, fast stage time intervals $[\theta_{0,i}, \theta_{F,i}]$, initial conditions $v_{0,i}$, forcing functions $r_i(\theta)$, and embedding forcing function $\tilde{r}_s(\theta)$. These forcing functions are typically constructed using linear combinations of $\left\{f^{\{S\}}(\theta_{F,j}, Y_j)\right\}$, that serve to propagate information from the slow to the fast time scales. The embedded solution $\tilde{y}_{n+1}$ is similar to embedded solutions in standard Runge-Kutta methods. It is a solution with an alternate order of accuracy computed with minimal extra effort once the primary solution y_{n+1} is computed, in this case it is computed by re-solving the last stage with an alternate forcing function, $\tilde{r}_s(\theta)$.

As seen in step 2a above, computation of each stage in an infinitesimal method requires solving a secondary, *inner*, IVP. Typically, these inner IVPs are not solved exactly, and instead they are approximated through another numerical method using inner time step size $h \ll H$. In this work, we consider these inner IVPs to be solved using a one-step method that "subcycles" the solution using step sizes $h = \frac{H}{M}$, where M is an integer multirate ratio that describes the difference in dynamical time scales between $f^{\{S\}}$ and $f^{\{F\}}$.

Adaptive time step control is the technique of changing the time step size throughout a solve with the goal of producing a solution which is accurate to a given tolerance of the true solution. This topic has been thoroughly researched with numerous algorithms developed

for single-rate methods, i.e., methods that use a shared step size H for all components of $f(t, y)$ [16, 55, 56, 54]. Savcenco and collaborators developed a component-wise multirate adaptivity strategy designed for use with single-rate methods [48]; however, minimal work has been done in developing multirate controllers for use with multirate methods. Specifically, to our knowledge, the only published work on error-based multiple time step adaptivity for multirate methods was developed by Sarshar and collaborators [47]. The two simple adaptivity schemes proposed in that work were designed for MrGARK methods, and were not a primary focus of the paper.

To address the need for a wider ecosystem of algorithms for multirate time step adaptivity, in Section 3.2 we develop a family of simultaneous controllers for both H and M, following the techniques of Gustafsson [16], derived using constant and linear approximations of the principal error function. We additionally introduce two methods that modify our derived controllers so that they more closely emulate the structure of standard single-rate PI [23, 19, Chapter IV.2] and PID [23, 55] controllers.

All of the proposed controllers update both the values of the slow step size H and the multirate ratio M because single-rate adaptivity of just H while keeping M static may lead to an unnecessary amount of computational effort when H is small, or stages with insufficient accuracy when H is large. Conversely, single-rate adaptivity of only $h = H/M$ while holding H static inhibit the solver flexibility, again either resulting in excessive computational effort or solution error, particularly for problems where the dynamical time scales change throughout the simulation.

Each of our proposed controllers strive to achieve a target overall solution accuracy. However, errors in multirate infinitesimal approximations can arise from both the choice of H and M, where the former determines the approximation error arising from the slow time scale, and the latter determining the error that results from approximating solutions in step 2a of the above algorithm. We thus propose a set of options for estimating the errors arising from each of these contributions in Section 3.3.

Finally, in Section 3.4 we identify a variety of test problems and a range of multirate infinitesimal methods to use as a testing suite and define metrics with which to measure performance of our controllers. We then use this test suite to examine the performance of our proposed algorithms for multirate infinitesimal error estimation, followed by a comparison of the performance of our proposed controllers.

3.2. Control-theoretic approaches for multirate adaptivity

The error in a multirate infinitesimal method at the time t_n, $\varepsilon_n := \|y(t_n) - y_n\|$, can be bounded by the sum of the *slow error* $\varepsilon_n^{\{S\}}$ inherent to the method itself (which assumes that stages in step 2a are solved *exactly*), and the *fast error* $\varepsilon_n^{\{F\}}$ caused by approximating these stage solutions using some other "inner" solver, i.e.,

$$\varepsilon_n = \|y(t_n) - y_n\| = \|y(t_n) - y_n^* + y_n^* - y_n\| \leq \|y(t_n) - y_n^*\| + \|y_n^* - y_n\| = \varepsilon_n^{\{S\}} + \varepsilon_n^{\{F\}}.$$

Here, we use y_n^* to denote the "imagined" solution in which each stage is solved exactly. We can express the above error contributions over the time step $t_n \to t_{n+1} = t_n + H_n$ as

$$\begin{aligned} \varepsilon_{n+1}^{\{S\}} &= \phi_n^{\{S\}} H_n^P + \mathcal{O}\big(H_n^{P+1}\big) \\ \varepsilon_{n+1}^{\{F\}} &= \phi_n^{\{F\}} \left(\frac{H_n}{M_n}\right)^p H_n + \mathcal{O}\left(\left(\frac{H_n}{M_n}\right)^{p+1} H_n\right), \end{aligned} \tag{3.1}$$

where P and p are the global orders of accuracy for the multirate method embedding and inner method embedding, respectively. We use the embedding orders of accuracy (as opposed to the order of the main methods) since those will be used in Section 3.3 for computing our estimates of $\varepsilon^{\{S\}}$ and $\varepsilon^{\{F\}}$. The above expression for $\varepsilon^{\{F\}}$ is the global error for a one-step method with step-size H_n/M_n over a time interval of size H_n [18, Chapter II.3]. We note the introduction of the subscript "n" to indicate that these values only apply to the single multirate time step, and that these will be adaptively changed as the simulation proceeds.

The principal error functions $\phi^{\{S\}}$ and $\phi^{\{F\}}$ are functions of t that are unknown in general and assumed to be always nonzero, as they depend on the specific test problem and method used; however, they do not depend on the step size H_n or multirate ratio M_n. The notation ϕ_n is used in place of $\phi(t_n)$ for brevity. We note that, by convention, H_n, M_n, $\phi_n^{\{S\}}$, and $\phi_n^{\{F\}}$ give rise to errors $\varepsilon_{n+1}^{\{S\}}$ and $\varepsilon_{n+1}^{\{F\}}$, i.e., at the following time step t_{n+1}.

Although stages are typically solved over intervals of size less than H_n, the upper bound of the size of the interval is H_n, so we use H_n as an approximation for the interval size for the purposes of error estimation. Additionally, the inner method's step size H_n/M_n may be lead to overstepping the end of the interval, in which case the step size for the last step would need to be reduced. We similarly use the upper bound for the inner time step, H_n/M_n, as an approximation for the inner time step size for the purposes of error estimation.

In the derivations that follow we treat M_n as a real number, thus when implementing these controllers M must be rounded up to the nearest integer. Dropping the higher order terms from (3.1) and solving them for $\log(H_n)$ and $\log(M_n)$ gives

$$\begin{aligned}\log(H_n) &= \frac{\log(\varepsilon_{n+1}^{\{S\}}) - \log(\phi_n^{\{S\}})}{P} \\ \log(M_n) &= \frac{(p+1)(\log(\varepsilon_{n+1}^{\{S\}}) - \log(\phi_n^{\{S\}})) - P(\log(\varepsilon_{n+1}^{\{F\}}) - \log(\phi_n^{\{F\}}))}{Pp}.\end{aligned} \tag{3.2}$$

We wish to choose values of H_n and M_n such that the resulting error ε_{n+1} is equal to TOL, the desired tolerance for the numerical solution to approximate the true solution of the IVP. Since ε_n has contributions from both $\varepsilon^{\{F\}}$ and $\varepsilon^{\{S\}}$, we enforce that these sources be equal to their time-scale-specific tolerance values, i.e., we set $\varepsilon_{n+1}^{\{S\}} = \text{TOL}^{\{S\}}$ and $\varepsilon_{n+1}^{\{F\}} = \text{TOL}^{\{F\}}$, where $\text{TOL} = \text{TOL}^{\{S\}} + \text{TOL}^{\{F\}}$.

$$\begin{aligned}\log(H_n) &= \frac{\log(\text{TOL}^{\{S\}}) - \log(\phi_n^{\{S\}})}{P} \\ \log(M_n) &= \frac{(p+1)(\text{TOL}^{\{S\}}) - \log(\phi_n^{\{S\}})) - P(\log(\text{TOL}^{\{F\}}) - \log(\phi_n^{\{F\}}))}{Pp}.\end{aligned}$$

Because the principal error function values are generally unknown, we must approximate $\log(\phi_n^{\{S\}})$ and $\log(\phi_n^{\{F\}})$ with some $\log(\widehat{\phi}_n^{\{S\}})$ and $\log(\widehat{\phi}_n^{\{F\}})$. Assuming sufficiently accurate approximations of the principal error functions, we insert these in place of the exact principal error functions above to arrive at our final log-space formulation for the update functions,

$$\begin{aligned} \log(H_n) &= \frac{\log(\text{TOL}^{\{S\}}) - \log(\widehat{\phi}_n^{\{S\}})}{P} \\ \log(M_n) &= \frac{(p+1)(\log(\text{TOL}^{\{S\}}) - \log(\widehat{\phi}_n^{\{S\}})) - P(\log(\text{TOL}^{\{F\}}) - \log(\widehat{\phi}_n^{\{F\}}))}{Pp}. \end{aligned} \tag{3.3}$$

3.2.1. Approximations for $\log(\phi(t))$

In [16], Gustafsson developed a single-rate controller by approximating $\log(\phi(t))$ with a piecewise linear function. We extend this work by approximating $\log(\phi^{\{S\}}(t))$ and $\log(\phi^{\{F\}}(t))$ with piecewise constant functions and reiterate Gustafsson's piecewise linear derivation with further details on motivations throughout. As with Gustafsson's work, we introduce the shift operator q, defined as $q^a \log(\phi_n) = \log(\phi_{n+a})$; this is a common operator used in control theory to derive algebraic expressions in terms of a single iteration index [43]. By convention, the shift operator is assumed to be invertible, i.e., $q^{-a}q^a \log(\phi_n) = q^{-a} \log(\phi_{n+a}) = \log(\phi_n)$. The identity shift operator is defined as q^0 such that $q^0 \log(\phi_n) = \log(\phi_n)$.

3.2.1.1. Piecewise Constant $\log(\phi(t))$ *Approximation*

If we assume $\log(\phi(t))$ is constant in time, then $\log(\widehat{\phi}_n) = \log(\widehat{\phi}_{n-1})$. We can convert this to a piecewise constant by adding a corrector based on the true value of $\log(\phi)$, via an "observer" in the control [7, Chapters 11,13]. Introducing the observer with free parameter $k > 0$ gives

$$\log(\widehat{\phi}_n) = \log(\widehat{\phi}_{n-1}) + k(\log(\phi_{n-1}) - \log(\widehat{\phi}_{n-1}))$$

$$= (1-k)q^{-1}\log(\widehat{\phi}_n) + k\log(\phi_{n-1}).$$

As in Gustafsson [16], we assume that $q+(k-1)q^0$ is invertible to solve for $\log(\widehat{\phi}_n)$ to arrive at the piecewise constant approximation,

$$\log(\widehat{\phi}_n) = \frac{kq}{q+q^0(k-1)}\log(\phi_{n-1}). \tag{3.4}$$

3.2.1.2. Piecewise Linear $\log(\phi(t))$ *Approximation*

In this section we reproduce the derivation of Gustafsson, albeit with additional details that were lacking from [16]. Thus, assuming that $\log(\phi(t))$ is linear (and its time derivative is constant), we have

$$\log(\widehat{\phi}_n) = \log(\widehat{\phi}_{n-1}) + \nabla\log(\widehat{\phi}_{n-1}),$$

$$\nabla\log(\widehat{\phi}_n) = \nabla\log(\widehat{\phi}_{n-1}).$$

where ∇ is the gradient with respect to the iteration number, indexed by n, as in Gustafsson. We rewrite the system as

$$\widehat{x}_n = A\widehat{x}_{n-1}$$

$$\log(\widehat{\phi}_n) = C\widehat{x}_n$$

where

$$\widehat{x}_n = \begin{bmatrix} \log(\widehat{\phi}_n) \\ \\ \nabla\log(\widehat{\phi}_n) \end{bmatrix},\ \widehat{x}_{n-1} = \begin{bmatrix} \log(\widehat{\phi}_{n-1}) \\ \\ \nabla\log(\widehat{\phi}_{n-1}) \end{bmatrix},\ A = \begin{bmatrix} 1 & 1 \\ \\ 0 & 1 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

Again inserting a corrector based on the true values of $\log(\phi)$ and $\nabla \log(\phi)$, via an observer with free parameter vector $K = [k_1 \ k_2]^T$ in the control (here both $k_1, k_2 > 0$), we have

$$\widehat{x}_n = A\widehat{x}_{n-1} + K(\log(\phi_{n-1}) - \log(\widehat{\phi}_{n-1}))$$

$$= (A - KC)\widehat{x}_{n-1} + K\log(\phi_{n-1}).$$

Using a backwards difference approximation for ∇, applying the shift operator to convert all approximate $\log(\widehat{\phi})$ terms to the same iteration, summing the equations, and solving for $\log(\widehat{\phi}_n)$ in terms of $\log(\phi_{n-1})$ leads to the piecewise linear approximation for $\log(\phi)$,

$$\log(\widehat{\phi}_n) = \frac{(k_1 + k_2)q^2}{2q^2 + q(k_1 + k_2 - 4) + 2q^0} \log(\phi_{n-1}), \tag{3.5}$$

where we again assume that $2q^2 + q(k_1 + k_2 - 4) + 2q^0$ is invertible. We note that this approximation will lead to H-M controllers with the following structure, which depend on the two most recent values of H, but only on the most recent error term:

$$H_{n+1} = H_n^{\gamma_1} H_{n-1}^{\gamma_2} \left(\frac{\mathrm{TOL}}{\varepsilon_{n+1}^{\{S\}}} \right)^{\alpha}$$

This short error history was expected for our earlier piecewise constant derivation, but, because the controller now depends on a longer history of H, it is typical to convert this to depend on an equal history of error terms. We follow Gustafsson's derivation by enforcing the additional condition

$$k_1 \log(\widehat{\phi}_{n-1}) = k_1 \log(\phi_{n-1}) \tag{3.6}$$

that enforces continuity in the piecewise approximation. Subtracting this from (3.5), we may arrive at the modified approximation for $\log(\phi_n)$, assuming $2q^2 + q(k_1 + k_2 - 4) + q^0(2 - k_1)$

is invertible,

$$\log(\widehat{\phi}_n) = \frac{(k_1+k_2)q^2 - k_1 q}{2q^2 + q(k_1+k_2-4) + q^0(2-k_1)} \log(\phi_{n-1}). \tag{3.7}$$

3.2.1.3. Expression for $\log(\phi_{n-1})$

Both the piecewise constant and piecewise quadratic approaches above resulted in approximations (3.4) and (3.7) that depend on the true values of the principal error function at the previous step, $\log(\phi_{n-1})$. Although these principal error function values are generally unknown, they may be approximated using estimates of the error at the end of the previous step. Depending on whether $\log(\phi^{\{S\}})$ or $\log(\phi^{\{F\}})$ is being approximated, we may estimate the principal error function at the previous time step by rearranging (3.2),

$$\begin{aligned} \log(\phi_{n-1}^{\{S\}}) &= \log(\varepsilon_n^{\{S\}}) - Pq^{-1}\log(H_n) \\ \log(\phi_{n-1}^{\{F\}}) &= \log(\varepsilon_n^{\{F\}}) - (p+1)q^{-1}\log(H_n) + q^{-1}p\log(M_n), \end{aligned} \tag{3.8}$$

3.2.2. H-M Controllers

At this point we have all of the requisite parts to finalize controller formulations for H_n and M_n. To this end, we insert the expressions (3.8) into the approximations (3.4) or (3.7). We then insert the resulting formulas into the log-space formulation of the update functions (3.3), apply the shift operator, and solve the resulting equations for $\log(H_{n+1})$ and $\log(M_{n+1})$ in terms of known or estimated values of $\log(H_n)$, $\log(M_n)$, $\log(\varepsilon_n^{\{S\}})$, and $\log(\varepsilon_n^{\{F\}})$. We then finalize each controller by taking the exponential of these log-space update functions.

For simplicity of notation, we introduce the "oversolve" factors, $\eta_n^{\{S\}} := \left(\text{TOL}^{\{S\}}\right)/\varepsilon_n^{\{S\}}$ and $\eta_n^{\{F\}} := \left(\text{TOL}^{\{F\}}\right)/\varepsilon_n^{\{F\}}$. The values of these factors reflect the deviation between the achieved errors and their target, where a value of one indicates an optimal control.

There are four potential combinations of our two piecewise approximations for each of $\log(\phi_n^{\{S\}})$ and $\log(\phi_n^{\{F\}})$. While using differing order approximations is valid in theory, we

have not found this useful in practice. In particular, due to the coupled nature of the general controllers (3.3), the extent of the error history for both fast *and* slow errors in the M_n update function will correspond to the higher-order of the $\log(\phi_n^{\{S\}})$ and $\log(\phi_n^{\{F\}})$ approximations, and thus it makes little sense to use a lower-order approximation for $\log(\phi_n^{\{F\}})$ than $\log(\phi_n^{\{S\}})$. Additionally, we expect the fast time scale (associated with M) to vary more rapidly than the slow time scale for multirate applications, so we also we rule out cases where the approximation order for $\log(\phi_n^{\{F\}})$ exceeds that of $\log(\phi_n^{\{S\}})$.

3.2.2.1. Constant-Constant Controller

When using a piecewise constant approximation for both $\log(\phi^{\{S\}})$ and $\log(\phi^{\{F\}})$, we follow the procedure outlined above to arrive at the following equations for $\log(H_n)$ and $\log(M_n)$,

$$\log(H_n) = \frac{k_1 q}{P(q-1)} \log\left(\eta_n^{\{S\}}\right),$$

$$\log(M_n) = \frac{(p+1)k_1 q}{Pp(q-1)} \log\left(\eta_n^{\{S\}}\right) - \frac{k_2 q}{p(q-1)} \log\left(\eta_n^{\{F\}}\right).$$

These give the controller pair

$$\begin{aligned} H_{n+1} &= H_n \left(\eta_{n+1}^{\{S\}}\right)^{\alpha}, \\ M_{n+1} &= M_n \left(\eta_{n+1}^{\{S\}}\right)^{\beta_1} \left(\eta_{n+1}^{\{F\}}\right)^{\beta_2}, \end{aligned} \tag{3.9}$$

where

$$\alpha = \frac{k_1}{P}, \quad \beta_1 = \frac{(p+1)k_1}{Pp}, \quad \text{and} \quad \beta_2 = \frac{-k_2}{p}.$$

3.2.2.2. Linear-Linear Controller

Similarly, using a piecewise linear approximation for both $\log(\phi^{\{S\}})$ and $\log(\phi^{\{F\}})$, and solving for $\log(H_n)$ and $\log(M_n)$, we have

$$\log(H_n) = \frac{(k_{11}+k_{12})q^2 - k_{11}q}{2P(q^2-2q+1)} \log\left(\eta_n^{\{S\}}\right)$$

$$\log(M_n) = \frac{(p+1)\left((k_{11}+k_{12})\,q^2 - k_{11}q\right)}{2Pp(q^2-2q+1)} \log\left(\eta_n^{\{S\}}\right)$$

$$- \frac{(k_{21}+k_{22})q^2 - k_{21}q}{2p(q^2-2q+1)} \log\left(\eta_n^{\{F\}}\right).$$

This results in the controller pair

$$\begin{aligned} H_{n+1} &= H_n^2 H_{n-1}^{-1} \left(\eta_{n+1}^{\{S\}}\right)^{\alpha_1} \left(\eta_n^{\{S\}}\right)^{\alpha_2}, \\ M_{n+1} &= M_n^2 M_{n-1}^{-1} \left(\eta_{n+1}^{\{S\}}\right)^{\beta_{11}} \left(\eta_n^{\{S\}}\right)^{\beta_{12}} \left(\eta_{n+1}^{\{F\}}\right)^{\beta_{21}} \left(\eta_n^{\{F\}}\right)^{\beta_{22}}, \end{aligned} \tag{3.10}$$

where

$$\begin{aligned} \alpha_1 &= \frac{k_{11}+k_{12}}{2P}, & \alpha_2 &= \frac{-k_{11}}{2P} \\ \beta_{11} &= \frac{(p+1)(k_{11}+k_{12})}{2Pp}, & \beta_{12} &= \frac{-(p+1)k_{11}}{2Pp}, \\ \beta_{21} &= \frac{-(k_{21}+k_{22})}{2p}, & \beta_{22} &= \frac{k_{21}}{2p}. \end{aligned} \tag{3.11}$$

3.2.3. A Note on Higher Order $\log(\phi(t))$ Approximations

Following the preceding approaches, it is possible to form controllers based on higher order polynomial approximations to $\log(\phi(t))$. While we have investigated these, we did not find such multirate H-M controllers to have robust performance. For example, a "Quadratic-Quadratic" controller that uses piecewise quadratic approximation for both $\log(\phi_n^{\{S\}})$ and $\log(\phi_n^{\{F\}})$ depends on an extensive history of both H or M terms, and by design attempts to

limit changes in H and M to ensure that these vary smoothly throughout a simulation. In our experiments, we found that this over-accentuation of smoothness rendered the controller to lack sufficient flexibility to grow or shrink H or M at a rate at which some multirate problems demanded. The Quadratic-Quadratic controller thus exhibited a "bimodal" behavior, in that for some problems it failed quickly, whereas for others it performed excellently, but we were unable to tune it so that it would provide robustness across our test problem suite. Since our goal in this paper is to construct controllers with wide applicability to many multirate applications, we omit the Quadratic-Quadratic controller, or any other controllers derived from a higher order $\log(\phi_n)$ approximations, from this paper.

However, to address the over-smoothness issue we introduce two additional controllers, one that results from a simple modification of the Linear-Linear controller above, and the other that extends this idea to include additional error estimates.

3.2.4. PIMR Controller

The "PI" controller, popular in single-rate temporal adaptivity, has a similar qualitative structure to the Linear-Linear controller (3.10), albeit with a reduced dependence on the history of H_n as in [23] and [19, Chapter IV.2],

$$H_{n+1} = H_n \left(\frac{\text{tol}}{\varepsilon_{n+1}}\right)^{\alpha_1} \left(\frac{\text{tol}}{\varepsilon_n}\right)^{\alpha_2}. \tag{3.12}$$

We thus introduce a "PIMR" controller by eliminating one factor of both H_n/H_{n-1} and M_n/M_{n-1} from (3.10):

$$\begin{aligned} H_{n+1} &= H_n \left(\eta_{n+1}^{\{S\}}\right)^{\alpha_1} \left(\eta_n^{\{S\}}\right)^{\alpha_2}, \\ M_{n+1} &= M_n \left(\eta_{n+1}^{\{S\}}\right)^{\beta_{11}} \left(\eta_n^{\{S\}}\right)^{\beta_{12}} \left(\eta_{n+1}^{\{F\}}\right)^{\beta_{21}} \left(\eta_n^{\{F\}}\right)^{\beta_{22}}, \end{aligned} \tag{3.13}$$

where α_1, α_2, β_{11}, β_{12}, β_{21} and β_{22} are defined as in (3.11).

3.2.5. PIDMR Controller

The higher-order "PID" controller from single-rate temporal adaptivity has a similar qualitative structure to the PI controller, but includes dependence on one additional error term [23, 55],

$$H_{n+1} = H_n \left(\frac{\text{tol}}{\varepsilon_{n+1}}\right)^{\alpha_1} \left(\frac{\text{tol}}{\varepsilon_n}\right)^{\alpha_2} \left(\frac{\text{tol}}{\varepsilon_{n-1}}\right)^{\alpha_3}. \tag{3.14}$$

The parameters in PID controllers typically alternate signs, with $\alpha_1 > 0$, $\alpha_2 < 0$ and $\alpha_3 > 0$, where each takes the form of a constant divided by the asymptotic order of accuracy for the single-rate method.

We thus introduce an extension of the PIMR controller to emulate this "PID" structure, resulting in the "PIDMR" controller. We strive to make this extension as natural as possible – we thus introduce new free parameters k_{13} and k_{23}, adjust the coefficient in the denominator of the exponents from 2 to 3, and assume that the exponents alternate signs. We thus define the PIDMR controller as

$$\begin{aligned} H_{n+1} &= H_n \left(\eta_{n+1}^{\{S\}}\right)^{\alpha_1} \left(\eta_n^{\{S\}}\right)^{\alpha_2} \left(\eta_{n-1}^{\{S\}}\right)^{\alpha_3}, \\ M_{n+1} &= M_n \left(\eta_{n+1}^{\{S\}}\right)^{\beta_{11}} \left(\eta_n^{\{S\}}\right)^{\beta_{12}} \left(\eta_{n-1}^{\{S\}}\right)^{\beta_{13}} \left(\eta_{n+1}^{\{F\}}\right)^{\beta_{21}} \left(\eta_n^{\{F\}}\right)^{\beta_{22}} \left(\eta_{n-1}^{\{F\}}\right)^{\beta_{23}}, \end{aligned} \tag{3.15}$$

where

$$\begin{aligned} \alpha_1 &= \frac{k_{11}+k_{12}+k_{13}}{3P}, & \alpha_2 &= \frac{-(k_{11}+k_{12})}{3P}, & \alpha_3 &= \frac{k_{11}}{3P}, \\ \beta_{11} &= \frac{(p+1)(k_{11}+k_{12}+k_{13})}{3Pp}, & \beta_{12} &= \frac{-(p+1)(k_{11}+k_{12})}{3Pp}, & \beta_{13} &= \frac{(p+1)k_{11}}{3Pp}, \\ \beta_{21} &= \frac{-(k_{21}+k_{22}+k_{23})}{3p}, & \beta_{22} &= \frac{k_{21}+k_{22}}{3p}, & \beta_{23} &= \frac{-k_{21}}{3p}. \end{aligned}$$

3.3. Multirate infinitesimal method error estimation

In infinitesimal methods, the embedded solution is only computed in step 3, i.e., as with standard Runge–Kutta methods the embedding *reuses* the results of each internal stage computation (step 2). Thus to measure $\varepsilon_{n+1}^{\{S\}}$ we will use a standard last-stage embedding of the infinitesimal method, e.g., $\varepsilon_{n+1}^{\{S\}} \approx \|y_{n+1} - \tilde{y}_{n+1}\|$. However, estimation of $\varepsilon_{n+1}^{\{F\}}$ is less obvious.

Within each of the fast IVP solves in steps 2a and 3, one must employ a separate IVP solver. For these, we assume that an embedded Runge–Kutta method is used which provides two distinct solutions of differing orders of accuracy for the fast IVP solution. There are a number of ways to utilize the embedded fast Runge–Kutta method to estimate $\varepsilon_{n+1}^{\{F\}}$. We outline five potential approaches with a range of computational costs here, and will compare their performance in Section 3.4.

3.3.1. Full-Step (FS) strategy

Our first approach for estimating the fast time scale error runs the entire infinitesimal step twice, once with a primary fast method for all fast IVP solves (producing the solution y_{n+1}) and once with the fast method's embedding (to produce the solution $\widehat{y}_{n+1}$). We use the difference of these to estimate the fast error, $\varepsilon_{n+1}^{\{F\}} \approx \|y_{n+1} - \widehat{y}_{n+1}\|$:

1. Let: $Y_1 = y_n$
2. For $i = 2, ..., s$:
   (a) Solve with primary fast method: $v_i{}'(\theta) = C_i\, f^{\{F\}}(\theta, v_i(\theta)) + r_i(\theta)$, for $\theta \in [\theta_{0,i}, \theta_{F,i}]$ with $v_i(\theta_{0,i}) = v_{0,i}$.
   (b) Solve with embedded fast method: $\widehat{v}_i'(\theta) = C_i\, f^{\{F\}}(\theta, \widehat{v}_i(\theta)) + \widehat{r}_i(\theta)$, for $\theta \in [\theta_{0,i}, \theta_{F,i}]$ with $\widehat{v}_i(\theta_{0,i}) = \widehat{v}_{0,i}$.
   (c) Let: $Y_i = v_i(\theta_{F,i})$ and $\widehat{Y}_i = \widehat{v}_i(\theta_{F,i})$.
3. Let: $y_{n+1} = Y_s$ and $\widehat{y}_{n+1} = \widehat{Y}_s$.

4. Let: $\varepsilon_{n+1}^{\{F\}} = \|y_{n+1} - \widehat{y}_{n+1}\|$.

While this is perhaps the most accurate method for estimating $\varepsilon^{\{F\}}$, it requires two separate computations of the full infinitesimal step, including two sets of evaluations of the slow right-hand side function, $f^{\{S\}}$. See Figure 3.1 for a diagram of this strategy.

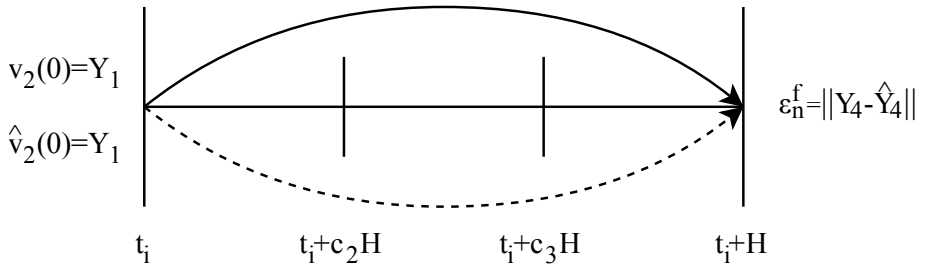


Figure 3.1: *Full-Step* strategy for fast error estimation. A full step is solved with both the primary fast method (solid line) and the embedded fast method (dashed line), which start from the same initial condition Y_1 at the first stage solve.

3.3.2. Stage-Aggregate (SA) strategy

As above, we consider two separate fast IVP solves for each stage, once with the primary fast method and once with its embedding. However, here both fast IVP solves use the results from the *primary* fast method for their initial conditions and the forcing terms $r_i(\theta)$. Here, we estimate the fast error by aggregating the norms of the difference in computed stages, denoted as

$$\varepsilon_{n+1}^{\{F\}} \approx \text{aggregate}(\|Y_i - \widehat{Y}_i\|,\ i = 2, \ldots, s),$$

where Y_i are the stages computed using the primary fast method, and $\widehat{Y}_i$ are the stages computed using the fast method's embedding. We consider aggregating by mean and max, and refer to these as *Stage-Aggregate-mean* (SA-mean) and *Stage-Aggregate-max* (SA-max), respectively:

1. Let: $Y_1 = y_n$

2. For $i = 2, ..., s$:

(a) Solve once using primary fast method and once using embedded fast method: $v_i'(\theta) = C_i\, f^{\{F\}}(\theta, v_i(\theta)) + r_i(\theta)$, for $\theta \in [\theta_{0,i}, \theta_{F,i}]$ with $v_i(\theta_{0,i}) = v_{0,i}$.

(b) Let: $Y_i = v_i(\theta_{F,i})$ when using primary fast method.

(c) Let: $\widehat{Y}_i = v_i(\theta_{F,i})$ when using embedded fast method.

3. Let: $y_{n+1} = Y_s$.

4. Let: $\varepsilon_{n+1}^{\{F\}} = \text{aggregate}(\|Y_i - \widehat{Y}_i\|,\ i = 2, \ldots, s)$.

Because this estimation strategy solves the *same* IVP at each stage using two separate methods, the number of $f^{\{S\}}$ evaluations is cut in half compared to the *Full-Step* strategy. See Figure 3.2 for a diagram of this strategy.

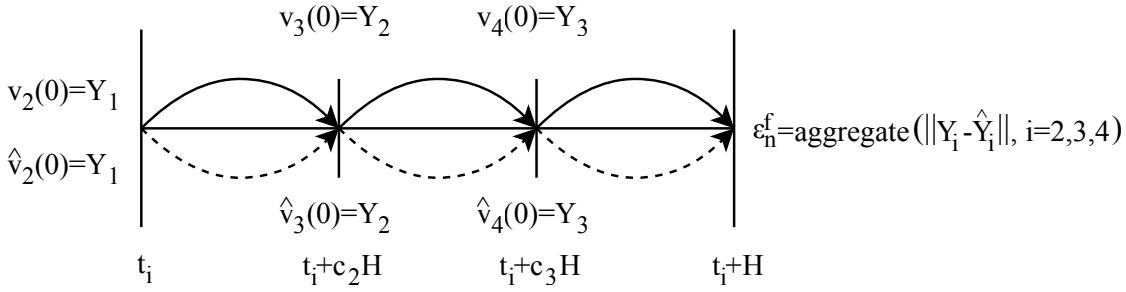


Figure 3.2: *Stage-Aggregate* strategy for fast error estimation. Each stage is solved with both the primary fast method (solid line) and the embedded fast method (dashed line). Both the primary and embedded fast methods use the result of the primary fast method as their initial condition for the next stage solve. The error is computed with an aggregating function on the stage errors.

3.3.3. Local-Accumulation-Stage-Aggregate (LASA) strategy

Our final $\varepsilon_{n+1}^{\{F\}}$ measurement strategy is designed to minimize the overall algorithmic cost. As with the *Stage-Aggregate* approaches above, we solve a single set of fast IVPs, however instead of using the embedded method solution as a separate fast solver, we evolve only the primary fast method and use its embedding to estimate the temporal error within each fast sub-step. To estimate the overall error within each slow stage, we sum the fast sub-step

errors, i.e., for the slow stage i and fast step j, we define $d_{j,i} = \|v_{j,i} - \widehat{v}_{j,i}\|$ as the norm of the difference between the primary fast method and its embedding. Then as with the *Stage-Aggregate* strategy, we aggregate the stage errors to estimate

$$\begin{aligned}\varepsilon_{n+1}^{\{F\}} &\approx \text{aggregate}(\|Y_i - \widehat{Y}_i\|,\ i = 2, \ldots, s) \\ &\approx \text{aggregate}\left(\sum_{j=1}^{M} d_{j,i},\ i = 2, \ldots, s\right).\end{aligned}$$

We again consider aggregating by mean and max, and we refer to the corresponding estimation strategies as *Local-Accumulation-Stage-Aggregate-mean* (LASA-mean) and *Local-Accumulation-Stage-Aggregate-max* (LASA-max), respectively:

1. Let: $Y_1 = y_n$
2. For $i = 2, ..., s$:
   (a) Solve : $v_i'(\theta) = C_i\, f^{\{F\}}(\theta, v_i(\theta)) + r_i(\theta)$, for $\theta \in [\theta_{0,i}, \theta_{F,i}]$ with $v_i(\theta_{0,i}) = v_{0,i}$.
      i. Let: $v_{j,i}$ be the step solution using the primary fast method at sub-step j, $j = 1, ..., M$.
      ii. Let: $\widehat{v}_{j,i}$ be the step solution using the embedded fast method at sub-step j, $j = 1, ..., M$.
      iii. Let: $d_{j,i} = \|v_{j,i} - \widehat{v}_{j,i}\|$.
      iv. Use $v_{j,i}$ as the initial condition for the next step.
   (b) Let: $Y_i = v_i(\theta_{F,i})$
   (c) Let: $\|Y_i - \widehat{Y}_i\| \approx \sum_{j=1}^{M} d_{j,i}$.
3. Let: $y_{n+1} = Y_s$.
4. Let: $\varepsilon_{n+1}^{\{F\}} = \text{aggregate}(\|Y_i - \widehat{Y}_i\|,\ i = 2, \ldots, s)$.

Here, since all fast IVPs use the same set of $f^{\{S\}}$ evaluations, then this approach uses half as many slow right-hand side evaluations as the *Full-Step* strategy. Additionally, since each

fast IVP is solved only once, then it uses approximately half the number of $f^{\{F\}}$ evaluations in comparison to both the *Full-Step* and *Stage-Aggregate* strategies. See Figure 3.3 for a diagram of this strategy.

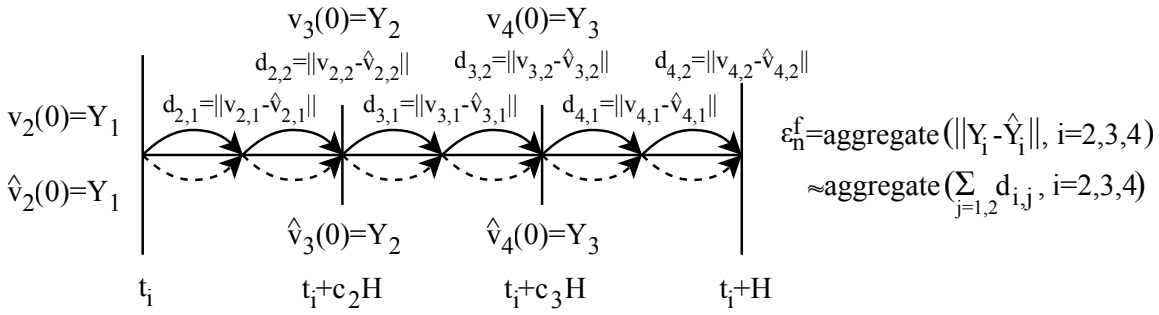


Figure 3.3: *Local-Accumulation-Stage-Aggregate* strategy for fast error estimation. Each step of each stage is solved with both the primary fast method (solid line) and the embedded fast method (dashed line), requiring no extra function evaluations. Each subsequent step of the fast method uses the result from the primary fast method as its initial condition. The norm of the difference between the fast step solutions is summed for each multirate stage and used as an approximation of the stage error, and the overall error is computed by aggregating the stage error approximations.

In Section 3.4.5, we compare the performance of these five strategies, based on both how close their solutions come to the desired tolerance, and on the relative computational cost of each strategy.

3.4. Numerical results

We assess performance for our five measurement strategies and four multirate controllers. We base these assessments both with respect to how well their resulting solutions match the desired tolerance, and also how close their computational costs were to a set of estimated optimal costs, which are discussed in Section 3.4.3 and Appendix A.1. All of the codes used for these computational results are available in the public GitHub repository: https://github.com/fishac/AdaptiveHMControllerPaper.

3.4.1. Test problems

We assessed the performance of both our fast error estimation strategies and multirate controllers using a test suite comprising seven multirate test problems. For each problem, we compute error using analytical solutions when available; otherwise we use MATLAB's *ode15s* with tight tolerances $AbsTol = 10^{-14}$ and $RelTol = 2.5 \times 10^{-14}$ to compute reference solutions at ten evenly spaced points in the problem's time interval.

3.4.1.1. Bicoupling

The Bicoupling problem is a nonlinear and nonautonomous multirate test problem proposed in [32],

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \gamma v - w - pt \\ -\gamma u \\ -lw - lpt - p\left(u - \dfrac{aw}{al+b\gamma} - \dfrac{apt}{al+b\gamma}\right)^2 - p\left(v - \dfrac{bw}{al+b\gamma} - \dfrac{bpt}{al+b\gamma}\right)^2 \end{bmatrix},$$

for $t \in [0, 1]$, with initial conditions $u(0) = 1 + a$, $v(0) = b$, $w(0) = al + b\gamma$ and parameters $a = 1$, $b = 20$, $\gamma = 100$, $l = 5$, and $p = 0.01$. This IVP has true solution

$$u(t) = \cos(\gamma t) + ae^{-lt}, \quad v(t) = -\sin(\gamma t) + be^{-lt}, \quad w(t) = (al + b\gamma)e^{-lt} - pt.$$

We apply the same multirate splitting of the right-hand side function as [32], that used $f^{\{S\}} = \begin{bmatrix} \gamma v & -\gamma u & 0 \end{bmatrix}^T$ and $f^{\{F\}} = f - f^{\{S\}}$.

3.4.1.2. Stiff Brusselator ODE

The Brusselator is an oscillating chemical reaction problem which is widely used to test multirate, implicit, and mixed implicit-explicit methods. We define the problem with the

same parameters as in [31],

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} a - (w+1)u + u^2 v \\ uw - u^2 v \\ \dfrac{b-w}{\epsilon} - uw \end{bmatrix}, \qquad \begin{bmatrix} u(0) \\ v(0) \\ w(0) \end{bmatrix} = \begin{bmatrix} 1.2 \\ 3.1 \\ 3 \end{bmatrix}$$

for $t \in [0, 2]$, and using the parameters $a = 1$, $b = 3.5$, and $\varepsilon = 0.01$. As we are unaware of an analytical solution to this IVP, we compute reference solutions as described above.

We apply the same multirate splitting of the right-hand side function as in [31],

$$f^{\{S\}} = \begin{bmatrix} a + (w+1)u + u^2 v \\ uw - u^2 v \\ \dfrac{b}{\epsilon} - uw \end{bmatrix}, \qquad f^{\{F\}} = \begin{bmatrix} 0 \\ 0 \\ -\dfrac{w}{\epsilon} \end{bmatrix}.$$

3.4.1.3. Kaps

The Kaps problem is an autonomous nonlinear problem with analytical solution that has been frequently used to test Runge–Kutta methods, presented in [53],

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} -(\mu+2)u + \mu v^2 \\ -v^2 + u - v \end{bmatrix}, \qquad \begin{bmatrix} u(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

for $t \in [0, 2]$, where we use the stiffness parameter $\mu = 100$. This IVP has true solution

$$u(t) = e^{-2t}, \qquad v(t) = e^{-t},$$

and we split the right-hand side function into slow and fast components as

$$f^{\{S\}} = \begin{bmatrix} 0 \\ \\ -v^2 + u - v \end{bmatrix}, \qquad f^{\{F\}} = \begin{bmatrix} -(\mu+2)u + \mu v^2 \\ \\ 0 \end{bmatrix}.$$

3.4.1.4. KPR

The KPR problem is a nonlinear IVP system with analytical solution, with variations that have been widely applied to test multirate algorithms. We use the same formulation as in [5],

$$\begin{bmatrix} u' \\ \\ v' \end{bmatrix} = \Lambda \begin{bmatrix} \frac{-3+u^2-\cos(\beta t)}{2u} \\ \\ \frac{-2+v^2-\cos(t)}{2v} \end{bmatrix} - \begin{bmatrix} \frac{\beta \sin(\beta t)}{2u} \\ \\ \frac{\sin(t)}{2v} \end{bmatrix}, \qquad \begin{bmatrix} u(0) \\ \\ v(0) \end{bmatrix} = \begin{bmatrix} 2 \\ \\ \sqrt{3} \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} \lambda^{\{F\}} & \frac{1-\varepsilon}{\alpha}(\lambda^{\{F\}} - \lambda^{\{S\}}) \\ \\ -\alpha\varepsilon(\lambda^{\{F\}} - \lambda^{\{S\}}) & \lambda^{\{S\}} \end{bmatrix},$$

for $t \in [0, 5\pi/2]$, and with the parameters $\lambda^{\{S\}} = -1$, $\lambda^{\{F\}} = -10$, $\alpha = 1$, $\beta = 20$, and $\epsilon = 0.1$. This IVP has true solution

$$u(t) = \sqrt{3 + \cos(\beta t)}, \qquad v(t) = \sqrt{2 + \cos(t)},$$

and we split the right-hand side function component-wise as in [5, 45],

$$f^{\{S\}} = \begin{bmatrix} 0 & 0 \\ \\ 0 & 1 \end{bmatrix} \Lambda \begin{bmatrix} \frac{-3+u^2-\cos(\beta t)}{2u} \\ \\ \frac{-2+v^2-\cos(t)}{2v} \end{bmatrix} - \begin{bmatrix} 0 \\ \\ \frac{\sin(t)}{2v} \end{bmatrix},$$

$$f^{\{F\}} = \begin{bmatrix} 1 & 0 \\ \\ 0 & 0 \end{bmatrix} \Lambda \begin{bmatrix} \frac{-3+u^2-\cos(\beta t)}{2u} \\ \\ \frac{-2+v^2-\cos(t)}{2v} \end{bmatrix} - \begin{bmatrix} \frac{\beta \sin(\beta t)}{2u} \\ \\ 0 \end{bmatrix}.$$

3.4.1.5. Forced Van der Pol

We consider a forced version of the widely-used Van der Pol oscillator test problem, defined in [40],

$$\begin{bmatrix} u' \\ \\ v' \end{bmatrix} = \begin{bmatrix} v \\ \\ -u - 8.53(u^2-1)v + 1.2\sin\left(\frac{\pi}{5}t\right) \end{bmatrix}, \qquad \begin{bmatrix} u(0) \\ \\ v(0) \end{bmatrix} = \begin{bmatrix} 1.45 \\ \\ 0 \end{bmatrix}$$

for $t \in [0, 25]$. As this IVP does not have an analytical solution, we compute reference solutions as described above.

We split the right-hand side function into linear and nonlinear slow and fast components, respectively,

$$f^{\{S\}} = \begin{bmatrix} v \\ \\ -u \end{bmatrix}, \qquad f^{\{F\}} = \begin{bmatrix} 0 \\ \\ -8.53(u^2-1)v + 1.2\sin(\frac{\pi}{5}t) \end{bmatrix}.$$

3.4.1.6. Pleiades

The Pleiades problem is a special case of the general N-Body problem from [18, Chapter II.10], here comprised of seven bodies in two physical dimensions, resulting in fourteen position and fourteen velocity components (p and v). This problem has initial conditions

$$p_1(0) = \begin{bmatrix} 3 & 3 \end{bmatrix}, \quad p_2(0) = \begin{bmatrix} 3 & -3 \end{bmatrix}, \quad p_3(0) = \begin{bmatrix} -1 & 2 \end{bmatrix}, \quad p_4(0) = \begin{bmatrix} -3 & 0 \end{bmatrix},$$

$$p_5(0) = \begin{bmatrix} 2 & 0 \end{bmatrix}, \quad p_6(0) = \begin{bmatrix} -2 & 4 \end{bmatrix}, \quad p_7(0) = \begin{bmatrix} 2 & 4 \end{bmatrix}$$

$$v_1(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}, \quad v_2(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}, \quad v_3(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}, \quad v_4(0) = \begin{bmatrix} 0 & -1.25 \end{bmatrix},$$

$$v_5(0) = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad v_6(0) = \begin{bmatrix} 1.75 & 0 \end{bmatrix}, \quad v_7(0) = \begin{bmatrix} -1.5 & 0 \end{bmatrix},$$

and solutions were considered on the interval $t \in [0, 3]$.

This IVP has no analytical solution, so we approximate reference solutions as described previously. We split the right-hand side function component-wise, such that $f^{\{S\}}$ contains the time derivatives of the positions, and $f^{\{F\}}$ contains the time derivatives of the velocities.

3.4.1.7. FourBody3D

The FourBody3D problem is another special case of the general N-Body problem, with four bodies in three spatial dimensions, defined in [40]. This problem has initial conditions

$$p_1(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \; p_2(0) = \begin{bmatrix} 4 & 3 & 1 \end{bmatrix}, \; p_3(0) = \begin{bmatrix} 3 & -4 & -2 \end{bmatrix},$$

$$p_4(0) = \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}, \; v_1(0) = v_2(0) = v_3(0) = v_4(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix},$$

and solutions were considered on the interval $t \in [0, 15]$.

This IVP has no analytical solution, and so reference solutions are approximated appropriately. As with the Pleiades problem, we split the right-hand side function so that $f^{\{S\}}$ contains the time derivatives of the positions, while $f^{\{F\}}$ contains the time derivatives of the velocities.

3.4.2. Testing Suite

We evaluated the performance of each of the five proposed fast error estimation strategies, *FS*, *SA-mean*, *SA-max*, *LASA-mean*, and *LASA-max*, over the above test problems, over the tolerances $\{10^{-3}, 10^{-5}, 10^{-7}\}$, and using each of the Constant-Constant, Linear-Linear, PIMR, and PIDMR controllers. For simplicity of presentation, we always set $\text{TOL}^{\{S\}} = \text{TOL}^{\{F\}} = \frac{1}{2}\text{TOL}$. When using controllers with extended histories, we used the Constant-Constant controller until a sufficient history had built up. Because we consider integer values of M, we take the *ceil* of the value from the M update functions resulting from each controller.

MRI-GARK [45] is the only family of infinitesimal methods we are aware of that has available embeddings for temporal error estimation. For our testing set we used:

- MRI-GARK-ERK33, a four-stage third-order MRI-GARK method with a second-order embedding, which is explicit at each slow stage.
- MRI-GARK-ERK45a, a six-stage fourth-order MRI-GARK method with a third-order embedding, which is explicit at each slow stage.
- MRI-GARK-IRK21a, a four-stage second-order MRI-GARK method with a first-order embedding, which is explicit in three slow stages, and implicit in one.
- MRI-GARK-ESDIRK34a, a seven-stage third-order MRI-GARK method with a second-order embedding, which is explicit in four slow stages, and implicit in three.

We chose these methods because they cover a range of orders of accuracy, and include an equal number of explicit and implicit methods. A post-publication correction was made to the method MRI-GARK-ERK45a [44], where the embedding coefficient row of the Γ^0 matrix is replaced with

$$\widehat{\Gamma}_6^0 = \begin{bmatrix} -\frac{1482837}{759520} & \frac{175781}{71205} & -\frac{790577}{1139280} & -\frac{6379}{56964} & \frac{47}{96} & 0 \end{bmatrix}.$$

We used the corrected coefficients in our tests.

In our numerical tests, we pair each MRI-GARK method with a fast explicit Runge–Kutta method of the same order. Specifically, for MRI-GARK-IRK21a we use the second-order Heun–Euler method, for MRI-GARK-ERK33 and MRI-GARK-ESDIR34a we use the third-order Bogacki–Shampine method [2], and for MRI-GARK-ERK45a we use the fourth-order Zonneveld method [64].

3.4.3. Assessing Adaptive Performance

To assess performance of adaptivity controllers, we defined a brute force algorithm which, for a given IVP problem, numerical method, and TOL, finds the largest value of H_n and smallest value of M_n at each step that will result in an approximate solution with error at or very near TOL. We consider the resulting total numbers of $f^{\{S\}}$ and $f^{\{F\}}$ evaluations, denoted $f_{\text{opt}}^{\{S\}}$ and $f_{\text{opt}}^{\{F\}}$, to be the estimated optimal costs of an adaptive solve. We provide a detailed description if this brute force algorithm (including pseudocode) in Appendix A.1.

3.4.4. Controller Parameter Optimization

Each of the controllers derived in Section 3.2 depend on a set of two to six free parameters. To compare the "best case" for each controller and error measurement strategy, we first numerically optimized the controller parameters across our testing suite of problems, methods, tolerances, and fast error measurement strategies.

To measure the quality of a given set of parameters, we computed three metrics. We first define the "Error Deviation" arising from a given adaptive controller on a given test τ as

$$(\text{Error Deviation})_\tau = \log_{10}\left(\frac{\varepsilon}{\text{TOL}}\right), \tag{3.16}$$

where ε is defined as the maximum relative error over ten equally spaced points in the test problem's time interval, which provides a robust error performance measurement across the entire interval. Here, a method that achieves solution accuracy close to the target TOL will

have (Error Deviation)$_\tau$ close to zero. A value of (Error Deviation)$_\tau > 0$ indicates $\varepsilon >$ TOL and (Error Deviation)$_\tau < 0$ indicates $\varepsilon <$ TOL.

Next, we use the total number of $f^{\{S\}}$ and $f^{\{F\}}$ evaluations, $f^{\{S\}}_{\text{evals}}$ and $f^{\{F\}}_{\text{evals}}$ (referred to as the "slow cost" and "fast cost", respectively), in relative factors to measure performance on a given test τ.

$$\text{(Slow Cost Deviation)}_\tau = \frac{f^{\{S\}}_{\text{evals}}}{f^{\{S\}}_{\text{opt}}}, \tag{3.17}$$

$$\text{(Fast Cost Deviation)}_\tau = \frac{f^{\{F\}}_{\text{evals}}}{f^{\{F\}}_{\text{opt}}}, \tag{3.18}$$

A method that achieves costs close to optimal will have (Slow Cost Deviation)$_\tau$ and (Fast Cost Deviation)$_\tau$ close to one, although typically these values are significantly larger. The primary purpose of this relative-to-optimal definition of cost is to accommodate the fact that different problems may require vastly different amounts of function evaluations, and we would like to consider the controllers' average performance across the testing suite.

To severely penalize parameter values that led to a lack of controller robustness, we defined an optimization objective function to be

$$E(k) = \sum_{\tau \in \text{test set}} E_\tau(k),$$

$$E_\tau(k) = \begin{cases} 10(\text{Slow Cost Deviation})_\tau + (\text{Fast Cost Deviation})_\tau & \\ \quad +10(\text{Error Deviation})^2_\tau, & \text{if } \tau \text{ finished} \\ 10^{10}, & \text{if } \tau \text{ failed.} \end{cases} \tag{3.19}$$

Here, the contribution to the objective function for a particular test τ is small if the computed error is close to the target tolerance, and if the computational costs are not much larger than the "optimal" values. We chose a factor of 10 for the Slow Cost Deviation to ensure that

$f_{\text{evals}}^{\{S\}}$ has a greater weight than $f_{\text{evals}}^{\{F\}}$ in the optimizer, and for the squared Error Deviation to prioritize computations that achieve the target accuracy.

Due to the lack of differentiability in $E(k)$ due to failed solves and integer-valued M, we performed a simple optimization strategy consisting of an iterative search over the two- to six-dimensional parameter space. We performed successive mesh refinement over an n-dimensional mesh $[0, 1]^n$, with an initial mesh using a spacing of 0.2. After evaluating the controller's performance on all sets of the parameters in the initial mesh, we refined the mesh around the parameter point having smallest objective function value with a mesh width of 0.4 in all n directions, and a spacing of 0.04. After evaluating the controller's performance on all of the parameters in this refined mesh, we refined the mesh a final time around the parameter point having smallest objective function value, with a mesh width of 0.08 in all n directions and a spacing of 0.02.

3.4.5. Fast error estimation strategy performance

Our primary question for the quality of each of our fast error estimation strategies is how well it can estimate the solution error arising from approximation of each fast IVP. Thus for each fast error measurement strategy, we define the average Error Deviation from the target tolerance as the average value of $(\text{Error Deviation})_\tau$ over τ in a test set comprised of all combinations of our seven test problems, four IVP methods, three tolerances, and four controllers. We plot these results in Figure 3.4, where we see that although each strategy followed a drastically different approach for error estimation, all were able to achieve approximations that achieved the target solution accuracy. However, we note that the FS, SA-max and LASA-max appear to have over estimated the fast error, leading to overly accurate results.

Given that each fast error estimation strategy is able to obtain results of desirable accuracy, our second question focuses on the efficiency of using each approach in practice. We thus define the relative cost of each fast error estimation strategy as the average value of both

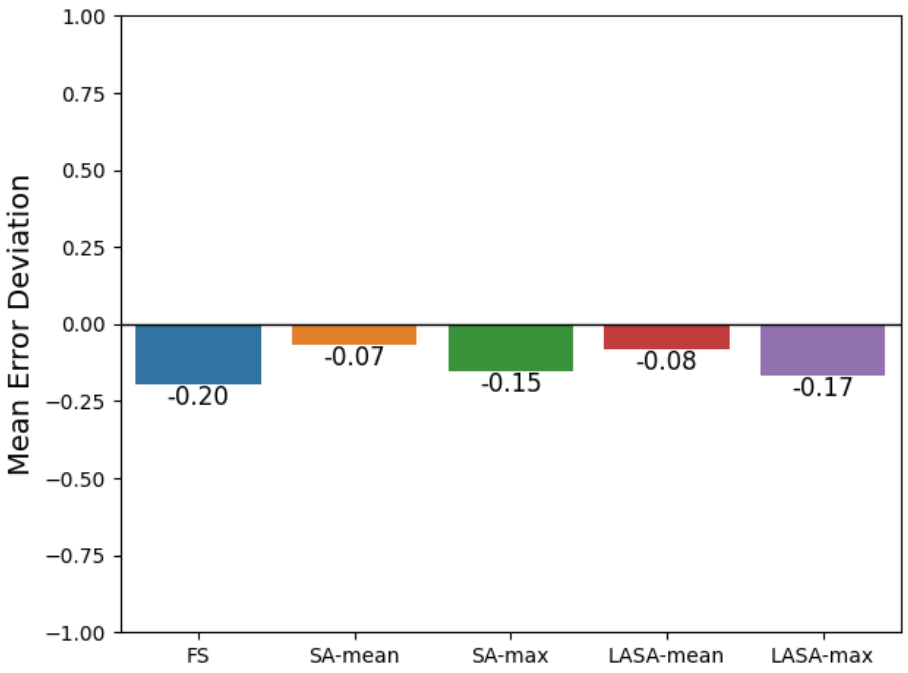
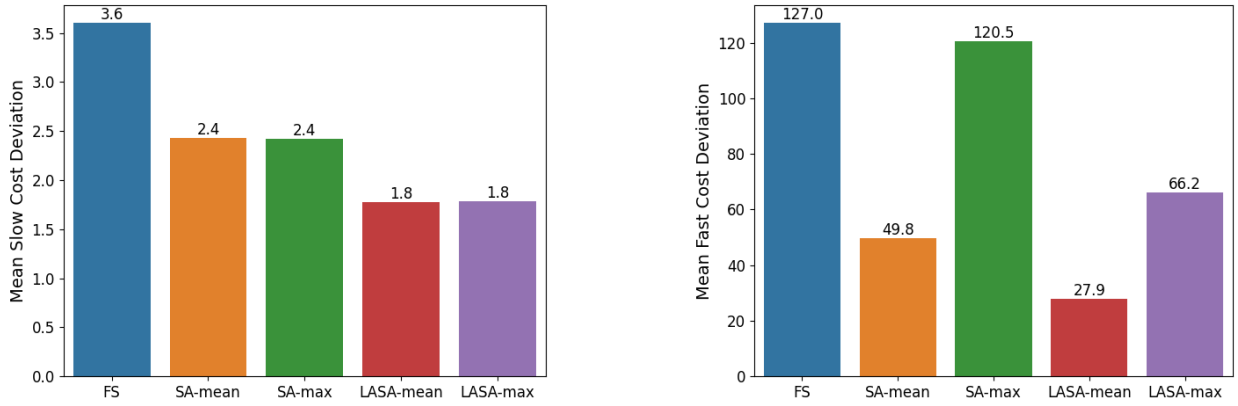


Figure 3.4: Mean Error Deviation arising from each fast error measurement strategy. All proposed methods obtained results that achieved the desired tolerance.

$(\text{Slow Cost Deviation})_\tau$ and $(\text{Fast Cost Deviation})_\tau$ over τ in the same test set comprising all combinations of our seven test problems, four IVP methods, three tolerances, and four controllers. We provide these plots in Figure 3.5, where we see that all of the fast error estimation strategies had average Slow Cost Deviation within a factor of two from one another, with the LASA strategies providing the closest-to-optimal slow cost. Additionally, the "LASA-mean" strategy provided by far the closest-to-optimal fast cost by a significant margin. This was expected, since the two LASA strategies were designed to minimize computational cost, yet their error estimates were sufficiently accurate. We believe that LASA-mean outperformed LASA-max because it provided a sharper estimate of fast solution error, as seen in Figure 3.4. Based on these results, in all subsequent numerical results we restrict our attention to the LASA-mean strategy alone.



(a) Slow Cost Deviation

(b) Fast Cost Deviation

Figure 3.5: Mean Slow and Fast Cost Deviation over test suite by each fast error measurement strategy.

3.4.6. Optimized Controller Parameters

After refining our focus to only the LASA-mean fast error estimation strategy, we reran the optimization approach described in Section 3.4.4 to determine an "optimal" set of parameters for each H-M controller. These final parameters were:

- Constant-Constant controller (3.9):

$$k_1 = 0.42, \qquad k_2 = 0.44. \tag{3.20}$$

- Linear-Linear controller (3.10):

$$K_1 = \begin{bmatrix} 0.82 & 0.54 \end{bmatrix}^T, \qquad K_2 = \begin{bmatrix} 0.94 & 0.9 \end{bmatrix}^T. \tag{3.21}$$

- PIMR controller (3.13):

$$K_1 = \begin{bmatrix} 0.18 & 0.86 \end{bmatrix}^T, \qquad K_2 = \begin{bmatrix} 0.34 & 0.80 \end{bmatrix}^T. \tag{3.22}$$

- PIDMR controller (3.15):

$$K_1 = \begin{bmatrix} 0.34 & 0.10 & 0.78 \end{bmatrix}^T, \qquad K_2 = \begin{bmatrix} 0.46 & 0.42 & 0.74 \end{bmatrix}^T. \tag{3.23}$$

3.4.7. Controller performance

With our chosen fast error estimation strategy and re-optimized parameters in place, we now compare the performance of our newly-proposed H-M controllers against the standard single-rate I, PI, and PID controllers, as well as Gustafsson's controller. For these tests, we utilized a subset of the testing suite above – namely, for each controller we considered all seven test problems, all four IVP methods, and all three accuracy tolerances. For the single-rate controllers, we held $M = 10$ constant for each test and used ε^s as the temporal error estimate.

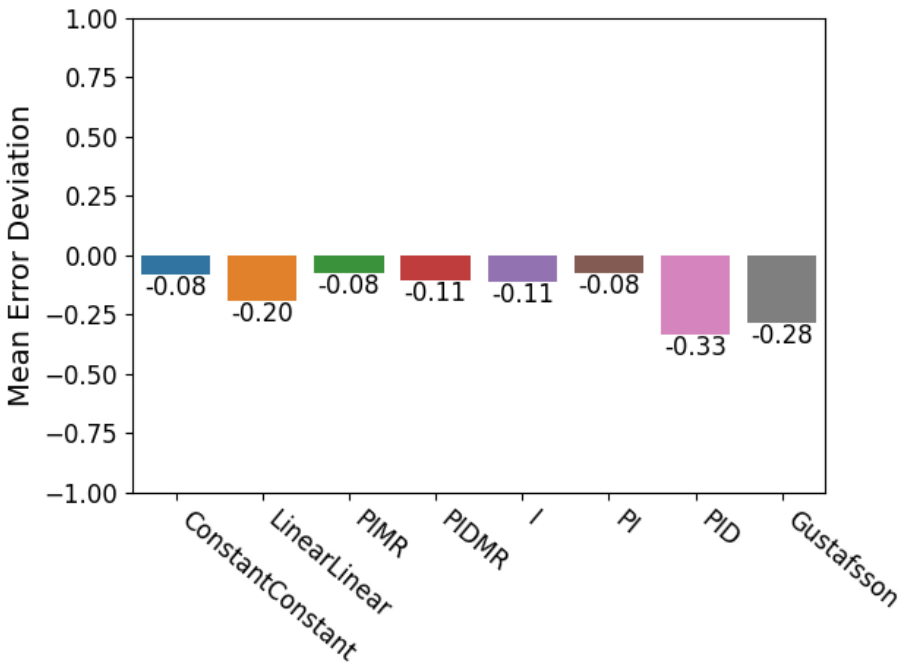
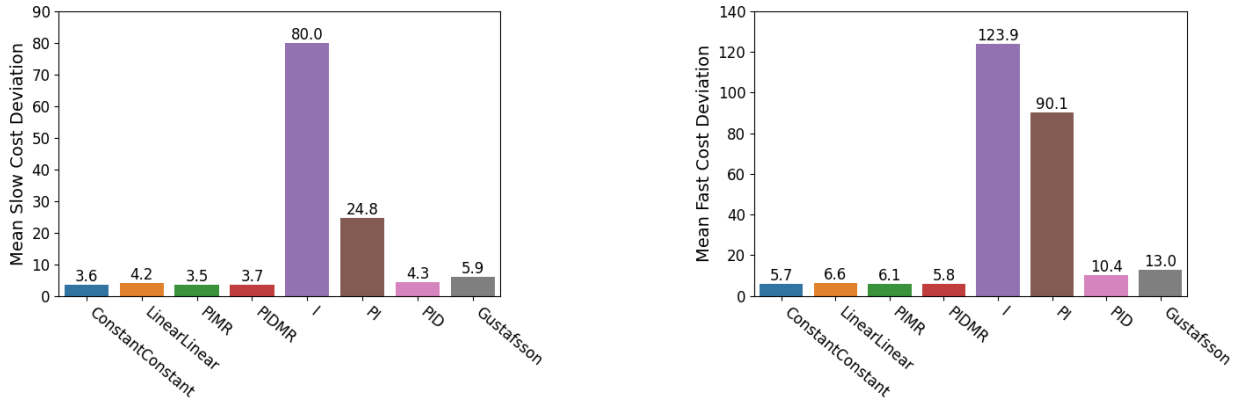


Figure 3.6: Mean Error Deviation over test suite for each controller.

In Figure 3.6, we plot the average Error Deviation (3.16) for each controller. We see that all controllers again lead to solutions with Error Deviation below zero, implying the achieved errors for all approaches achieve their target tolerances with a slight bias toward over-solving the problem. However, we note that although the differences are small, the Linear-Linear controller provides solutions with errors furthest from TOL *of all of our proposed controllers*, with a mean Error Deviation of -0.20, indicating that the solution had error approximately 6×10^{-4} when $\text{TOL} = 10^{-3}$, or 6×10^{-8} when $\text{TOL} = 10^{-7}$, which are still well within range of the target tolerance. We note that both the PID and Gustafsson single-rate controllers achieved solutions that were considerably more accurate than requested, although even those were within a reasonable range of the tolerance.



(a) Slow Cost Deviation

(b) Fast Cost Deviation

Figure 3.7: Mean Slow and Fast Cost Deviation over test suite by each controller.

In Figure 3.7, we plot the average cost deviation for each controller, over all of the combinations of test problems, methods, and tolerances. The proposed multirate controllers demonstrated comparable computational cost across our ODE test suite, with differences between the best and worst controllers of only 20% in terms of the average Slow Cost Deviation and only 16% in terms of the average Fast Cost Deviation. Meanwhile, the proposed multirate controllers uniformly outperformed their single-rate counterparts. Of those, the PID controller had the best performance which was on-par in average Slow Cost Deviation

but 58% worse in average Fast Cost Deviation than the worst-performing multirate controller (Linear-Linear). Gustafsson's controller had a slightly worse performance, and the I and PI controller's were dramatically worse. Additionally, we note that the I controller failed all runs on the Forced Van der Pol problem when using MRIGARKIRK21a and MRIGARKESDIRK34a, and so we did not include those runs in these I controller averages.

Based on these results, it is clear that the proposed multirate controllers all show excellent performance, with no single method outperforming another. Thus, for relatively simple IVPs we recommend the Constant-Constant controller due to its simplicity, whereas for more complex IVPs we recommend testing with each multirate controller.

3.4.8. Multirate controller performance deep dive

The previous results focused on averaged controller performance over a wide range of problems on which the controller parameters had already been optimized. In this section we instead compare the performance of our proposed controllers on a new test problem for which our adaptivity controllers have not been optimized, and that should thoroughly exercise their ability to adapt step sizes at both the fast and slow time scales. Thus this should provide an unbiased challenge problem on which we may compare controller performance, while also allowing a deeper dive into controller behavior.

We adapt the stiff Brusselator example from Section 3.4.1.2 to a 1D reaction-diffusion setting with time-varying coefficients,

$$\partial_t u = d(t)\,\partial_{xx} u + r(t)\left(a - (w+1)u + u^2 v\right),$$

$$\partial_t v = d(t)\,\partial_{xx} v + r(t)\left(uw - u^2 v\right),$$

$$\partial_t w = d(t)\,\partial_{xx} w + r(t)\left(\frac{b-w}{\epsilon} - uw\right),$$

for $(t, x) \in (0, 2) \times (0, 1)$, with initial conditions

$$u(0) = 1.2 + 0.1\sin(\pi x),\ v(0) = 3.1 + 0.1\sin(\pi x),\ w(0) = 3 + 0.1\sin(\pi x),$$

stationary boundary conditions,

$$\partial_t u(t, 0) = \partial_t u(t, 1) = \partial_t v(t, 0) = \partial_t v(t, 1) = \partial_t w(t, 0) = \partial_t w(t, 1) = 0,$$

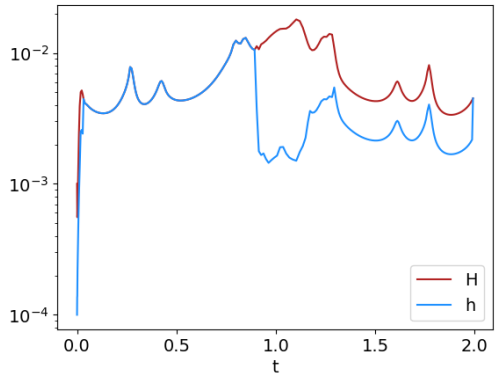
time-varying coefficient functions

$$d(t) = 0.006 + 0.005\cos(\pi t), \qquad r(t) = 0.6 + 0.5\cos(4\pi t),$$

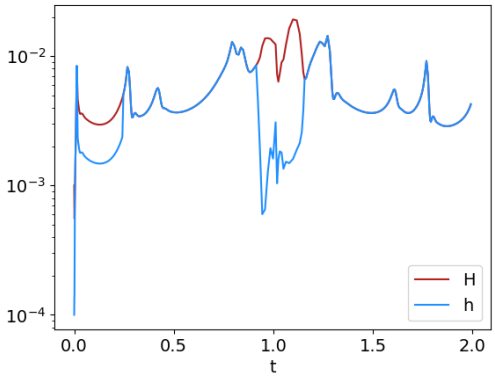
and parameters $a = 1$, $b = 3.5$. Here, we increase the stiffness of the problem by setting $\varepsilon = 0.001$ (previously this was 0.01). We partition this problem such that $f^{\{S\}}$ corresponds to the diffusion terms, while $f^{\{F\}}$ corresponds to the reaction terms. We note that for the above values, the diffusion coefficient $d(t)$ lies within $(0.001, 0.011)$ and the reaction coefficient $r(t)$ lies within $(0.1, 1.1)$, but that the frequencies of these oscillations differ, leading to coefficient ratios $d(t)/r(t)$ that range from approximately 10^{-3} to 10^{-1}. We thus expect that each of our adaptivity controllers will need to vary both H and M to accurately track the multirate solutions.

In lieu of averaging performance values across a multitude of methods, we focus on only MRIGARKERK45a here, although we note that the results are similar when using other multirate methods.

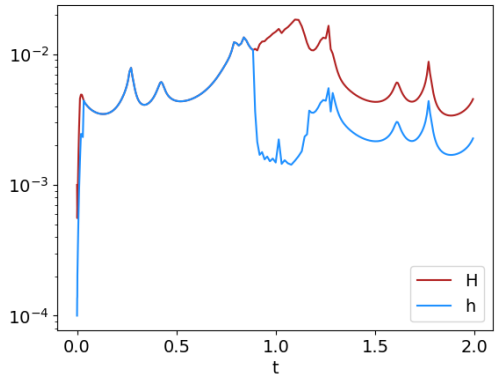
In Figure 3.8 we plot the time step sizes H_n and h_n over time for each of our controllers with a tolerance of 10^{-4}. We can see that for the first half of each simulation the problem did not exhibit multirate behavior, so all controllers varied H similarly and set $M = 1$ (except for Linear-Linear, that showed a brief initial period with $M > 1$). At approximately $t = 0.9$, some stiffness arises in the reaction network and the controllers all respond by increasing M



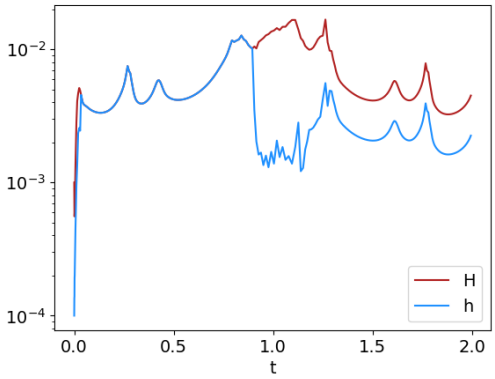
(a) Constant-Constant



(b) Linear-Linear



(c) PIMR



(d) PIDMR

Figure 3.8: H_n and h_n over time for each multirate controller with a tolerance of 10^{-4} on the 1D stiff Brusselator problem.

and thus decreasing h. We can see the effect of some failed steps at $t = 1.0$ in the Linear-Linear controller, where it decreases H and rapidly adjusts the value of M, while the other controllers more smoothly adjust H with some higher-frequency changes to M. Once the period of stiffness ends (around $t = 1.25$), the Linear-Linear controller resets M to 1, while the other controllers maintain a small value of $M = 2$.

In Figure 3.9 we examine the performance of each controller as the tolerance is varied, with plots of the Error Deviation, the total slow function evaluations, and the total fast function evaluations for each controller. All multirate controllers over-solved the problem, providing solutions two to four orders of magnitude more accurate than the chosen TOL, how-

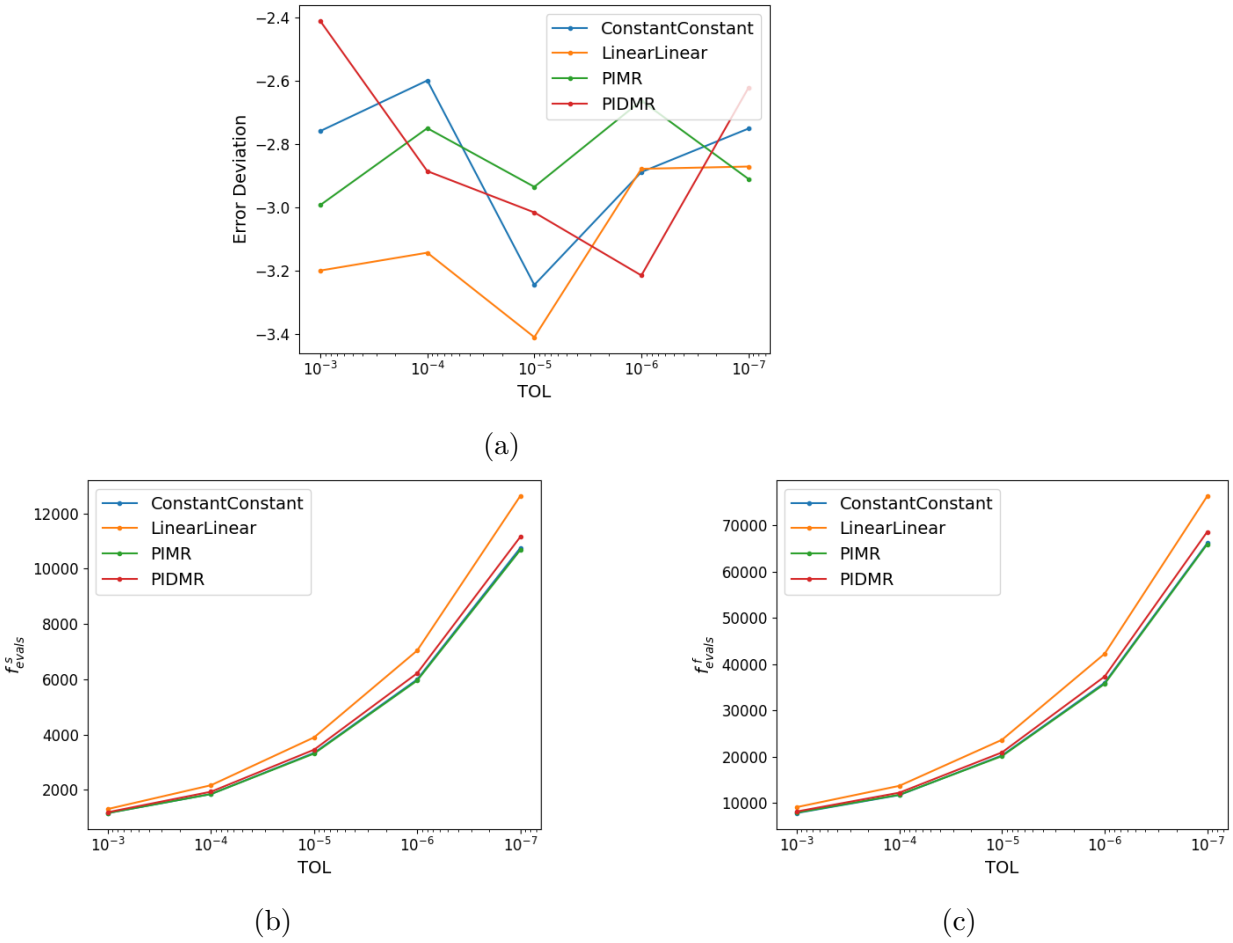


Figure 3.9: (a) Error Deviation, (b) Slow function evaluations, and (c) Fast function evaluations vs. TOL for each multirate controller on the 1D stiff Brusselator problem.

ever there was no consistent pattern as to which controller performed the best for loose/tight TOL values. Similarly, the controllers show comparable performance in terms of computational cost. Only the Linear-Linear controller had noticeably more function evaluations than the others, with approximately 13% more slow and 11% more fast function evaluations than the other controllers.

3.5. Conclusions

We followed the technique of Gustafsson [16] to develop controllers that approximate the fast and slow principal error functions for multirate infinitesimal methods. To this end, we

developed piecewise constant and linear approximations for each principal error function. We then combined these approximations using pairs of piecewise polynomial approximations to the principal error functions with like degree to construct Constant-Constant and Linear-Linear controllers for both the slow time step size H and the multirate ratio M, within multirate infinitesimal methods.

To assess the reliability and to measure the efficiency of these proposed controllers, we devised a large testing suite encompassing seven multirate test problems, four MRI methods and three accuracy tolerances. In order to measure method efficiency, we developed an algorithm to determine the best-case pair of H_n and M_n values for each testing combination.

In our initial tests, however, we found that controllers with polynomial approximations of degree two or larger to the principal error function tended to constrain step size changes too tightly, leading to a large number of solver failures. To address this issue, we introduced the PIMR controller, formed by taking the Linear-Linear controller and removing dependence on H and M terms prior to H_n and M_n, and the PIDMR controller, an extension to the PIMR controller with an increased error history. These controllers were developed to have similar structures to the existing single-rate PI and PID controllers. Through these modifications, the PIMR and PIDMR controllers can react more swiftly to a problem's influences on the multirate step size(s).

While estimation of the slow error $\varepsilon^{\{S\}}$ is straightforward for multirate methods with embeddings, we developed multiple strategies to estimate the fast error value $\varepsilon^{\{F\}}$, including the Full-Step, Stage-Aggregate, and Local-Accumulation-Stage-Aggregate strategies. These strategies trade off differing levels of computational effort with the expected accuracy in their estimation. However, when examining the performance of these approaches over our testing suite, we found that the Local-Accumulation-Stage-Aggregate strategy with mean aggregation offered an ideal combination of low cost and reliable accuracy to the target tolerance, and we therefore recommend it for practitioners.

We then evaluated the performance of each controller over our testing set, finding that the Constant-Constant, Linear-Linear, PIMR, and PIDMR controllers each tended to achieve solutions close to the target tolerance. While our proposed controllers perform similarly, they greatly outperform existing single-rate I, PI, Gustafsson's controllers, and slightly outperform the PID controller, in terms of both slow and fast function evaluations on average.

Finally, we evaluated our controllers on a stiffer, PDE version of the Brusselator problem. We saw that the controllers adjust both H and M in response to a period of increased stiffness, and readjusted after the period had ended. The controllers had a roughly equivalent computational cost in solving this problem, with the Linear-Linear controller consistently experiencing a slightly higher cost. Each controller tended to over-solve the problem, giving solutions with errors two to four orders of magnitude lower than the chosen value of TOL.

Significant work remains in the area of temporal adaptivity for multirate methods. MRI-GARK is the only infinitesimal method family we have found that includes embeddings. Thus, embedded methods from other multirate infinitesimal families need to be derived, along with a greater ecosystem of embedded MRI-GARK methods that focus more specifically on performance in a temporally adaptive context.

We additionally note that controllers which update H and M for each slow multirate step may not be the most efficient choice. We focused on H-M controllers, as those give H values that are an integer multiple of h values and result in a simpler method implementation; however, controllers may be created instead for H and h by following the steps outlined in this paper, replacing $\frac{H}{M}$ with h in the early steps. Perhaps the increased flexibility arising from real-valued h could lead to efficiency improvements over the integer-valued M approaches here.

Chapter 4

Implicit-explicit multirate infinitesimal stage-restart methods

The contents of this chapter have been submitted for publication with the title "Implicit-explicit multirate infinitesimal stage-restart methods" in collaboration with Dr. Daniel Reynolds and Dr. Steven Roberts. Background theory Sections 2.3, 2.4, 2.5, 2.6, and optionally 2.1 are useful in understanding this work.

4.1. Introduction

Flexible time integration methods for solving systems of initial-value problems (IVPs) have seen growing interest in recent years, largely due to their ability to provide highly accurate approximations of the IVP solution with increased computational efficiency. These integrators strive to reduce computational costs by partitioning the IVP into different components, and then treating each using different step sizes or numerical methods. Some of the primary families of flexible methods include implicit-explicit (IMEX) partitioning [3, 23, 25, 46], linear-nonlinear partitioning [20, 31, 32, 30, 35], and multirate partitioning [14, 45, 47, 61].

IMEX time integration methods solve IVPs in which the right-hand side function $f(t, y(t))$ is additively split into stiff $\{I\}$ and nonstiff $\{E\}$ processes,

$$\begin{aligned} y'(t) &= f(t,y) := f^{\{I\}}(t,y) + f^{\{E\}}(t,y), \quad t \geq t_0, \\ y(t_0) &= y_0. \end{aligned} \tag{4.1}$$

IMEX methods then couple two different numerical methods to treat these components: $f^{\{I\}}$ typically uses a stiff but computationally expensive solver, whereas $f^{\{E\}}$ may use a cheaper but nonstiff solver. For example, additive Runge–Kutta (ARK) methods typically combine

an A-stable diagonally-implicit Runge–Kutta (DIRK) method with an explicit Runge–Kutta (RK) method.

Similarly, multirate methods solve IVPs in which the right-hand side is additively partitioned into rapidly and slowly evolving dynamics, $\{F\}$ and $\{S\}$,

$$\begin{aligned} y'(t) &= f(t,y) := f^{\{F\}}(t,y) + f^{\{S\}}(t,y), \quad t \geq t_0, \\ y(t_0) &= y_0. \end{aligned} \tag{4.2}$$

Multirate methods then apply numerical methods with different step sizes for each component to save on computation time while retaining a desired level of accuracy.

In this work we combine the above approaches to consider a three-way additively partitioned IVP, wherein the slow partition $f^{\{S\}}$ from (4.2) is split in an IMEX fashion,

$$\begin{aligned} y'(t) &= f^{\{F\}}(t,y) + f^{\{I\}}(t,y) + f^{\{E\}}(t,y), \quad t \geq t_0, \\ y(t_0) &= y_0. \end{aligned} \tag{4.3}$$

In particular, we add a new class of methods to the ever-growing family of multirate infinitesimal (MRI) methods. These approximate the solution to (4.2) or (4.3) through solving a sequence of "fast" IVPs,

$$\begin{aligned} v_i'(\theta) &= f^{\{F\}}(\theta, v_i) + g_i(\theta), \quad \theta \in [\theta_{0,i}, \theta_{f,i}], \\ v(\theta_{0,i}) &= v_{0,i}. \end{aligned} \tag{4.4}$$

The forcing functions $g_i(\theta)$ incorporate information from the slow dynamics defined by $f^{\{I\}}$ and $f^{\{E\}}$ in a manner defined by the method. MRI methods assume that these fast IVPs (4.4) are solved exactly, but in practice these are approximated using an additional "inner" numerical method with a smaller step size than the multirate method. This inner method

can itself further decompose the problem through IMEX, linear-nonlinear, or multirate approaches.

Our proposed class of methods is called *Implicit-Explicit Multirate Infinitesimal Stage-Restart* (IMEX-MRI-SR) methods. Each stage of an IMEX-MRI-SR method consists of evolving a fast IVP followed by an implicit solve. This allows derivation of IMEX-MRI-SR methods by extending a base ARK method. We discuss the role of base methods further in Section 4.2.1, particularly focusing on their role in satisfying order conditions.

IMEX-MRI-SR methods are defined by n_Ω coefficient matrices $\Omega^{\{k\}} \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}}$, $k = 0, ..., n_\Omega - 1$, a coefficient matrix $\Gamma \in \mathbb{R}^{s^{\{S\}} \times s^{\{S\}}}$, and an abscissae vector $c^{\{S\}} \in \mathbb{R}^{s^{\{S\}}}$. Embedded versions of these methods include additional coefficient vectors $\hat{\omega}^{\{k\}} \in \mathbb{R}^{s^{\{S\}}}$ and $\hat{\gamma} \in \mathbb{R}^{s^{\{S\}}}$. The algorithm for evolving a solution to an IVP of the form (4.3) is defined as follows.

Definition 4.1.1 (IMEX-MRI-SR methods for additively partitioned systems)**.** *An IMEX-MRI-SR method evolves the solution to the problem* (4.3) *from* t_n *to* $t_n + H$ *according to the following algorithm.*

$$\text{Let: } Y_1^{\{S\}} := y_n \tag{4.5a}$$

$$\text{For: } i = 2, ..., s^{\{S\}}$$

$$\begin{cases} \text{Let:} & v_i(0) := y_n, \\ \text{Solve:} & v_i'(\theta) = f^{\{F\}}(t_n + \theta, v_i(\theta)) + g_i(\theta), \text{ for } \theta \in [0, c_i^{\{S\}} H] \\ & \text{where } g_i(\theta) = \dfrac{1}{c_i^{\{S\}}} \displaystyle\sum_{j=1}^{i-1} \omega_{i,j} \left(\dfrac{\theta}{c_i^{\{S\}} H} \right) \left(f_j^{\{E\}} + f_j^{\{I\}} \right) \\ \text{Solve:} & Y_i^{\{S\}} = v_i \left(c_i^{\{S\}} H \right) + H \displaystyle\sum_{j=1}^{i} \gamma_{i,j} f_j^{\{I\}}, \end{cases} \tag{4.5b}$$

$$\text{Let: } y_{n+1} := Y_{s^{\{S\}}}^{\{S\}}, \tag{4.5c}$$

$$\begin{cases} \text{Let:} & \hat{v}(0) := y_n, \\ \text{Solve:} & \hat{v}(\theta) = f^{\{F\}}(t_n + \theta, \hat{v}(\theta)) + \hat{g}(\theta), \text{ for } \theta \in [0, H] \\ & \text{where } \hat{g}(\theta) = \sum_{j=1}^{s^{\{S\}}-1} \hat{\omega}_j \left(\frac{\theta}{H}\right) \left(f_j^{\{E\}} + f_j^{\{I\}}\right) \\ \text{Solve:} & \hat{y}_{n+1} = \hat{v}\,(H) + H \sum_{j=1}^{s^{\{S\}}-1} \hat{\gamma}_j f_j^{\{I\}} + H\hat{\gamma}_{s^{\{S\}}} f^{\{I\}}(t_n + H, \hat{y}_{n+1}), \end{cases} \tag{4.5d}$$

where $f_j^{\{E\}} := f^{\{E\}}\left(t_n + c_j^{\{S\}} H, Y_j^{\{S\}}\right)$ and $f_j^{\{I\}} := f^{\{I\}}\left(t_n + c_j^{\{S\}} H, Y_j^{\{S\}}\right)$. Here y_{n+1} is the time-evolved approximation to $y(t_n + H)$, and $\hat{y}_{n+1}$ is an embedded solution used for temporal error esimation. If temporal error estimation is not needed, then step (4.5d) may be omitted.

Definition 4.1.2 (Slow tendency coefficients)**.** *The coefficients $\omega_{i,j}$ in* (4.5b) *are defined as in [5]*

$$\omega_{i,j}(\tau) := \sum_{k=0}^{n_\Omega - 1} \omega_{i,j}^{\{k\}} \tau^k, \quad \overline{\omega}_{i,j} := \int_0^1 \omega_{i,j}(\tau) \mathrm{d}\tau = \sum_{k=0}^{n_\Omega - 1} \frac{\omega_{i,j}^{\{k\}}}{k+1}, \tag{4.6}$$

and we refer to $\Omega^{\{k\}}$, $\overline{\Omega}$, and Γ as the $s^{\{S\}} \times s^{\{S\}}$ matrices containing the coefficients $\{\omega_{i,j}^{\{k\}}\}$, $\{\overline{\omega}_{i,j}\}$, and $\{\gamma_{i,j}\}$, respectively. As in [5] we assume the first row of the coefficient matrices are identically zero, and that $\Omega^{\{k\}}$ and $\overline{\Omega}$ are strictly lower-triangular. The embedding functions $\hat{\omega}_j(\tau)$ are defined similarly to (4.6)*, with vectors of coefficients $\hat{\omega}^{\{k\}}$.*

The rest of this paper is structured as follows. In the next subsection, we present related methods to IMEX-MRI-SR, discussing both their similarities and their limitations that are improved upon by the proposed methods. In Section 4.2 we prove order conditions for IMEX-MRI-SR methods up to order four, and in Section 4.3 we examine their linear stability. In Section 4.4 we provide embedded IMEX-MRI-SR methods of orders 2 through 4. In Section 4.5 we show that a previous class of multirate methods may be reformulated as IMEX-MRI-SR methods, and we use our convergence theory to show previously unproven features of those methods. We then present numerical results in Section 4.6 to validate our convergence theory, and to compare the efficiency of IMEX-MRI-SR methods against existing methods for

problems of the form (4.3). Finally, in Section 4.7 we conclude this article with a summary of our contributions, and an outlook toward future work.

4.1.1. Related methods

To our knowledge, there exist three MRI algorithms that allow both IMEX partitioning of the slow dynamics and infinitesimal treatment of the fast dynamics. These are the first-order accurate "Lie–Trotter" [8, 42] and the second-order accurate "Strang–Marchuk" [36, 58] operator splitting methods, and the recent fourth-order IMEX-MRI-GARK [5] method, an IMEX variation of MRI-GARK [45] methods. Both Lie–Trotter and Strang–Marchuk operate by sequentially applying distinct solvers to each component, only communicating with one another through the initial conditions applied within each sub-solve. Variations of both classes of methods for the IMEX multirate splitting (4.3) are shown in [5]. However, due to their weak "initial condition" coupling, Lie–Trotter and Strang–Marchuk are limited to at most first and second order accuracy in time, regardless of the order of accuracy of each component solver.

IMEX-MRI-GARK methods are organized similarly to IMEX-MRI-SR, in that they advance the solution by alternating between evolving fast IVPs and solving implicit algebraic equations involving $f^{\{I\}}$, and they use the result from each stage to provide a contribution to $g_i(\theta)$ for later stages. However, unlike IMEX-MRI-SR methods, IMEX-MRI-GARK methods evolve each fast IVP over an interval $[c_{i-1}H, c_iH]$, with an initial condition given as the result of the previous stage. In each stage, either a fast evolution or implicit solve may occur, with this choice dictated by the abscissae: when $\Delta c_i := c_i - c_{i-1} > 0$ a fast evolution occurs, but when $\Delta c_i = 0$ an algebraic system must be solved. In all existing implicit MRI-GARK and IMEX-MRI-GARK methods, authors have derived schemes by beginning with a base DIRK or ARK method, and then introduced additional internal stages to ensure an alternating pattern of $\Delta c_i \neq 0$ followed by $\Delta c_{i+1} = 0$, thereby ensuring an appropriate structure [5, 45]. However, these methods inherently require abscissae vectors $c^{\{S\}}$ that are

non-decreasing. This is generally an uncommon feature in RK methods, especially for orders of accuracy higher than two [3, 23, 24, 25, 39, 41, 46], so there are relatively few base DIRK and ARK methods available for deriving new MRI-GARK and IMEX-MRI-GARK methods. Additionally, the "padding" process for adding internal stages to ensure $\Delta c_i = 0$ is not obvious, and frequently results in an overly complicated trial-and-error process to decide where to insert stages. As a result of these challenges, no authors have successfully created IMEX-MRI-GARK methods with embeddings for temporal error estimation.

A second class of methods that are closely related to IMEX-MRI-SR are multirate exponential Runge–Kutta (MERK) methods [31]. While these do not support implicitness at the slow time scale (i.e., $f^{\{I\}} = 0$), and they assume that the fast partition is linear (i.e., $f^{\{F\}}(t,y) = \mathcal{L}y$), their structure matches (4.5). Each internal stage is computed through evolving a fast IVP over an interval $[0, c_i H]$, using a forcing function that is determined through the values of $f^{\{E\}}$ at previous slow stages.

4.2. IMEX-MRI-SR Order Conditions

Similar to IMEX-MRI-GARK methods, we derive order conditions for IMEX-MRI-SR methods by first representing the algorithm in GARK form. Due to the 3-component partitioning (4.3), we must identify GARK coefficients $\mathbf{A}^{\{\sigma,\nu\}}$, $\mathbf{b}^{\{\nu\}}$, and $\mathbf{c}^{\{\nu\}}$ for $\sigma \in \{S, F\}$ and $\nu \in \{I, E, F\}$. We refer to $\mathbf{A}^{\{F,\nu\}}$, $\nu \in \{I, E, F\}$ as the fast GARK tables, and to $\mathbf{A}^{\{S,\nu\}}$, $\nu \in \{I, E, F\}$ as the slow GARK tables. We assume the fast IVP (4.5b) is solved with one step of a sufficiently accurate Runge–Kutta method having $s^{\{F\}}$ stages, and defined by the Butcher table ($A^{\{F\}}$, $b^{\{F\}}$, $c^{\{F\}}$). The j^{th} sub-stage in computing the solution to the fast IVP $v_i'(\theta)$ is given by

$$V_{i,j} = y_n + c_i^{\{S\}} H \sum_{k=1}^{s^{\{F\}}} a_{j,k}^{\{F\}} f_{i,k}^{\{F\}} + H \sum_{k=1}^{s^{\{F\}}} a_{j,k}^{\{F\}} \sum_{\ell=1}^{i-1} \omega_{i,\ell}(c_k^{\{F\}}) \left(f_\ell^{\{E\}} + f_\ell^{\{I\}} \right) \tag{4.7}$$

for $i = 1, ..., s^{\{S\}}$, and $j = 1, ..., s^{\{F\}}$, where $f_{i,k}^{\{F\}} := f^{\{F\}}(t_n + c_k^{\{F\}} H, V_{i,k})$, $f_\ell^{\{E\}} := f^{\{E\}}(t_n + c_\ell^{\{S\}} H, Y_\ell^{\{S\}})$ and $f_\ell^{\{I\}} := f^{\{I\}}(t_n + c_\ell^{\{S\}} H, Y_\ell^{\{S\}})$. The IMEX-MRI-SR method's slow stages $Y_i^{\{S\}}$, $i = 1, \ldots, s^{\{S\}}$, are then computed by

$$\begin{aligned} Y_i^{\{S\}} &= y_n + c_i^{\{S\}} H \sum_{j=1}^{s^{\{F\}}} b_j^{\{F\}} f_{i,j}^{\{F\}} \\ &\quad + H \sum_{j=1}^{s^{\{F\}}} b_j^{\{F\}} \sum_{\ell=1}^{i-1} \omega_{i,\ell} \left(c_j^{\{F\}}\right) \left(f_\ell^{\{E\}} + f_\ell^{\{I\}}\right) + H \sum_{\ell=1}^{i} \gamma_{i,\ell} f_\ell^{\{I\}}. \end{aligned} \tag{4.8}$$

We leverage GARK theory [46] to construct order conditions as in [5, 45]. Since the slow partitions share the same stages, $Y_i^{\{S\}}$, these methods have six GARK matrices, $\mathbf{A}^{\{F,F\}}$, $\mathbf{A}^{\{F,E\}}$, $\mathbf{A}^{\{F,I\}}$, $\mathbf{A}^{\{S,F\}}$, $\mathbf{A}^{\{S,E\}}$, and $\mathbf{A}^{\{S,I\}}$ and three GARK vectors $\mathbf{b}^{\{F\}}$, $\mathbf{b}^{\{E\}}$, and $\mathbf{b}^{\{I\}}$,

$$\begin{array}{c|c|c} \mathbf{A}^{\{F,F\}} & \mathbf{A}^{\{F,E\}} & \mathbf{A}^{\{F,I\}} \\ \hline \mathbf{A}^{\{S,F\}} & \mathbf{A}^{\{S,E\}} & \mathbf{A}^{\{S,I\}} \\ \hline \mathbf{b}^{\{F\},T} & \mathbf{b}^{\{E\},T} & \mathbf{b}^{\{I\},T} \end{array}.$$

Here, $\mathbf{A}^{\{F,F\}}$ is a square $s^{\{SF\}} \times s^{\{SF\}}$ matrix (where we define $s^{\{SF\}} = s^{\{S\}} \cdot s^{\{F\}}$), containing the coefficients relating the fast stages $\{V_{i,j}\}$ to each other. It is block-diagonal because $V_{i,j}$ depends only on $V_{i,k}$, $k = 1, ..., s^{\{F\}}$ through the $A^{\{F\}}$ coefficients, and never on $V_{\ell j}$, $\ell \neq i$. These $s^{\{F\}} \times s^{\{F\}}$ block-diagonal elements are named $A^{\{F,F,i\}}$.

$\mathbf{A}^{\{F,E\}}$ and $\mathbf{A}^{\{F,I\}}$ are tall $s^{\{SF\}} \times s^{\{S\}}$ matrices relating the fast stages $\{V_{i,j}\}$ to the explicit and implicit function evaluations of the slow stages $\{Y_i^{\{S\}}\}$, comprised of $s^{\{S\}}$ blocks named $A^{\{F,E,i\}}$.

$\mathbf{A}^{\{S,F\}}$ is a wide $s^{\{S\}} \times s^{\{SF\}}$ matrix, containing the coefficients relating the slow stages $\{Y_i^{\{S\}}\}$ to the fast stages $\{V_{i,j}\}$. We name these $s^{\{S\}}$ total $s^{\{S\}} \times s^{\{F\}}$ blocks that comprise

$\mathbf{A}^{\{S,F\}}$ as $A^{\{S,F,i\}}$. Each $A^{\{S,F,i\}}$ contains at most one row of non-zero entries, located in row i, because $Y_i^{\{S\}}$ depends only on $V_{i,j}$, $j = 1, ..., s^{\{F\}}$, and never on $V_{\ell,j}$, $\ell \neq i$.

$\mathbf{A}^{\{S,E\}}$ and $\mathbf{A}^{\{S,I\}}$ are square $s^{\{S\}} \times s^{\{S\}}$ matrices relating the slow stages $\{Y_i^{\{S\}}\}$ to the explicit and implicit function evaluations of those slow stages.

The vectors $\mathbf{b}^{\{\sigma\}}$ equal the last rows of $\mathbf{A}^{\{S,\sigma\}}$, $\sigma \in \{F, E, I\}$, because IMEX-MRI-SR methods have the first-same-as-last (FSAL) property, where the last stage is used as the solution to the step.

When an embedding (4.5d) is included, it will correspond to three additional GARK vectors, $\hat{\mathbf{b}}^{\{F\}}$ $\hat{\mathbf{b}}^{\{E\}}$ and $\hat{\mathbf{b}}^{\{I\}}$. Due to the structural similarity of (4.5d) to the last stage $Y_{s^{\{S\}}}^{\{S\}}$, the contents of these vectors will only differ from $\mathbf{b}^{\{F\}}$, $\mathbf{b}^{\{E\}}$ and $\mathbf{b}^{\{I\}}$ through their dependence on $\hat{\omega}_j^{\{k\}}$ and $\hat{\gamma}_j$ instead of $\omega_{s^{\{S\}},j}^{\{k\}}$ and $\gamma_{s^{\{S\}},j}$. Thus the order conditions that follow for the primary GARK matrices and vectors can be applied to the embedding as well.

With the above simplifications, the GARK tableau can be expressed in block-matrix form as

$$
\begin{array}{ccc|c|c}
A^{\{F,F,1\}} & & & A^{\{F,E,1\}} & A^{\{F,I,1\}} \\
& \ddots & & \vdots & \vdots \\
& & A^{\{F,F,s^{\{S\}}\}} & A^{\{F,E,s^{\{S\}}\}} & A^{\{F,I,s^{\{S\}}\}} \\
\hline
A^{\{S,F,1\}} & \cdots & A^{\{S,F,s^{\{S\}}\}} & \mathbf{A}^{\{S,E\}} & \mathbf{A}^{\{S,I\}} \\
\hline
\multicolumn{3}{c|}{\mathbf{b}^{\{F\},T}} & \mathbf{b}^{\{E\},T} & \mathbf{b}^{\{I\},T}
\end{array}
.
$$

A GARK method with this tabular structure has stage update formulas

$$V_{i,j} = y_n + H \sum_{k=1}^{s^{\{F\}}} a_{j,k}^{\{F,F,i\}} f_{i,j}^{\{F\}} + H \sum_{k=1}^{s^{\{F\}}} a_{j,k}^{\{F,E,i\}} f_j^{\{E\}} + H \sum_{k=1}^{s^{\{F\}}} a_{j,k}^{\{F,I,i\}} f_j^{\{I\}}, \tag{4.9a}$$

$$Y_i^{\{S\}} = y_n + H \sum_{j=1}^{s^{\{F\}}} a_{i,j}^{\{S,F,i\}} f_{ij}^{\{F\}} + H \sum_{j=1}^{s^{\{F\}}} \mathbf{A}_{i,j}^{\{S,E\}} f_j^{\{E\}} + H \sum_{j=1}^{s^{\{F\}}} \mathbf{A}_{i,j}^{\{S,I\}} f_j^{\{I\}}. \tag{4.9b}$$

By matching coefficients in (4.7) and (4.9a), we identify the fast GARK coefficients,

$$a_{j,k}^{\{F,F,i\}} = c_i^{\{S\}} a_{j,k}^{\{F\}}, \tag{4.10a}$$

$$a_{j,\ell}^{\{F,E,i\}} = \sum_{l=1}^{s^{\{F\}}} a_{j,k}^{\{F\}} \omega_{i,\ell}(c_l^{\{F\}}) = \sum_{l=1}^{s^{\{F\}}} \sum_{k=0}^{n_\Omega - 1} \omega_{i,\ell}^{\{k\}} a_{j,l}^{\{F\}} c_l^{\{F\}\times k}, \tag{4.10b}$$

$$a_{j,\ell}^{\{F,I,i\}} = a_{j,\ell}^{\{F,E,i\}} = \sum_{k=1}^{s^{\{F\}}} b_{j,k}^{\{F\}} \omega_{i,\ell}(c_k^{\{F\}}), \tag{4.10c}$$

where the superscript $\times k$ denotes element-wise exponentiation of a vector by k. Converting these to matrix form, we have the fast GARK tables,

$$\mathbf{A}^{\{F,F\}} = C^{\{S\}} \otimes A^{\{F\}} \in \mathbb{R}^{s^{\{SF\}} \times s^{\{SF\}}}, \tag{4.11a}$$

$$\mathbf{A}^{\{F,E\}} = \sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\}\times k} \in \mathbb{R}^{s^{\{SF\}} \times s^{\{S\}}}, \tag{4.11b}$$

$$\mathbf{A}^{\{F,I\}} = \mathbf{A}^{\{F,E\}} = \sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\}\times k} \in \mathbb{R}^{s^{\{SF\}} \times s^{\{S\}}}, \tag{4.11c}$$

where $\otimes$ denotes the Kronecker product and $C^{\{\sigma\}} = \mathrm{diag}(c^{\{\sigma\}})$.

We similarly find the slow GARK table coefficients by comparing (4.8) and (4.9b),

$$a_{i,j}^{\{S,F,i\}} = c_i^{\{S\}} b_j^{\{F\}}, \tag{4.12a}$$

$$a_{i,\ell}^{\{S,E\}} = \sum_{k=1}^{s^{\{F\}}} b_k^{\{F\}} \omega_{i,\ell}(c_k^{\{F\}}) = \sum_{k=0}^{n_\Omega - 1} \omega_{i,\ell}^{\{k\}} b^{\{F\},T} c^{\{F\}\times k}, \tag{4.12b}$$

$$a_{i,\ell}^{\{S,I\}} = \sum_{k=1}^{s^{\{F\}}} b_k^{\{F\}} \omega_{i,\ell}\left(c_j^{\{F\}}\right) + \gamma_{i,\ell} = \sum_{k=0}^{n_\Omega - 1} \omega_{i,\ell}^{\{k\}} b^{\{F\},T} c^{\{F\}\times k} + \gamma_{i,\ell}. \tag{4.12c}$$

Due to the infinitesimal nature of the fast method, we assume that it satisfies all bushy-tree order conditions,

$$b^{\{F\},T}c^{\{F\}\times k} = \frac{1}{k+1}, \quad k = 0, ..., n_\Omega - 1.$$

Leveraging this, and examining (4.12), the slow GARK tables in matrix-form are

$$\mathbf{A}^{\{S,F\}} = C^{\{S\}} \otimes b^{\{F\},T} \in \mathbb{R}^{s^{\{S\}}\times s^{\{SF\}}}, \tag{4.13a}$$

$$\mathbf{A}^{\{S,E\}} = \sum_{k=0}^{n_\Omega-1} \Omega^{\{k\}} \frac{1}{k+1} = \overline{\Omega} \in \mathbb{R}^{s^{\{S\}}\times s^{\{S\}}}, \tag{4.13b}$$

$$\mathbf{A}^{\{S,I\}} = \sum_{k=0}^{n_\Omega-1} \Omega^{\{k\}} \frac{1}{k+1} + \Gamma = \overline{\Omega} + \Gamma \in \mathbb{R}^{s^{\{S\}}\times s^{\{S\}}}. \tag{4.13c}$$

We additionally define the following variables, knowing that an IMEX-MRI-SR method has the FSAL property with respect to the slow stages $Y^{\{S\}}$,

$$\mathbf{b}^{\{F\},T} = e^T_{s^{\{S\}}}\mathbf{A}^{\{S,F\}} = (C^{\{S\}}e_{s^{\{S\}}} \otimes b^{\{F\},T}) = (e_{s^{\{S\}}} \otimes b^{\{F\}})^T, \tag{4.14a}$$

$$\mathbf{b}^{\{E\},T} = e^T_{s^{\{S\}}}\mathbf{A}^{\{S,E\}} = e^T_{s^{\{S\}}}\overline{\Omega}, \tag{4.14b}$$

$$\mathbf{b}^{\{I\},T} = e^T_{s^{\{S\}}}\mathbf{A}^{\{S,I\}} = e^T_{s^{\{S\}}}(\overline{\Omega} + \Gamma), \tag{4.14c}$$

where $e_{s^{\{S\}}}$ is an $s^{\{S\}}$-length vector of all zeroes except a one in the last position.

4.2.1. Base Consistency

If $f^{\{F\}}(t, y) = 0$, then an IMEX-MRI-SR method reduces to a simple ARK method defined by slow explicit and implicit base methods,

$$(A^{\{E\}}, b^{\{E\}}, c^{\{E\}}) = (\mathbf{A}^{\{S,E\}}, \mathbf{b}^{\{E\}}, \mathbf{A}^{\{S,E\}}\mathbb{1}^{s^{\{S\}}}),$$

$$(A^{\{I\}}, b^{\{I\}}, c^{\{I\}}) = (\mathbf{A}^{\{S,I\}}, \mathbf{b}^{\{I\}}, \mathbf{A}^{\{S,I\}}\mathbb{1}^{s^{\{S\}}}),$$

where $\mathbb{1}^{s^{\{S\}}}$ is a vector of ones with length $s^{\{S\}}$. This ARK method has stages

$$Y_i^{\{S\}} = y_n + H \sum_{j=1}^{i-1} a_{i,j}^{\{E\}} f^{\{E\}}(t_n + c_j^{\{E\}} H, Y_j^{\{S\}})$$

$$+ H \sum_{j=1}^{i} a_{i,j}^{\{I\}} f^{\{I\}}(t_n + c_j^{\{I\}} H, Y_j^{\{S\}}),$$

with the last stage being equal to the solution for the step. We will refer to this ARK method as the slow base method throughout the derivation of the order conditions.

Remark 4.2.1 (First order conditions)**.** *As long as the slow base method and the arbitrary fast method are order one or higher, there are no additional first-order coupling conditions for a GARK method. Thus, the effective first-order IMEX-MRI-SR condition is that* $\mathbf{A}^{\{S,E\}} = \overline{\Omega}$ *and* $\mathbf{A}^{\{S,I\}} = \overline{\Omega} + \Gamma$ *form an order one ARK method, which can be simply achieved if both* $\mathbf{A}^{\{S,E\}}$ *and* $\mathbf{A}^{\{S,I\}}$ *have first-order accuracy.*

Remark 4.2.2 (Deriving IMEX-MRI-SR methods from existing ARK methods)**.** *Due to the IMEX-MRI-SR structure, a base ARK method should be stiffly-accurate; otherwise it must first be converted to stiffly-accurate form by appending the b vectors to the bottom and pad a column of zeros to the right of the ARK's A matrices.*

4.2.2. Kronecker Product Identities

In the ensuing derivations we leverage the following identities:

$$(A \otimes B)^T = A^T \otimes B^T, \tag{4.15a}$$

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD), \tag{4.15b}$$

$$(A \otimes v)\mathbb{1}^{\{r(A)\}} = (A\mathbb{1}^{\{r(A)\}}) \otimes v, \tag{4.15c}$$

$$(v \otimes A)\mathbb{1}^{\{r(A)\}} = v \otimes (A\mathbb{1}^{\{r(A)\}}), \tag{4.15d}$$

where A, B, C, D are arbitrary matrices with compatible dimemsions, v is an arbitrary vector, and $\mathbb{1}^{\{r(A)\}}$ is a vector of ones with length equal to the number of rows of A. Identities (4.15a) and (4.15b) can be found in [29]. Identities (4.15c) and (4.15d) can be shown through elementary computation.

4.2.3. GARK Internal Consistency

Theorem 4.2.1 (GARK Internal Consistency)**.** *An IMEX-MRI-SR method satisfies the GARK internal consistency conditions,*

$$\mathbf{c}^{\{F,F\}} = \mathbf{c}^{\{F,E\}} = \mathbf{c}^{\{F,I\}}, \tag{4.16a}$$

$$\mathbf{c}^{\{S,F\}} = \mathbf{c}^{\{S,E\}} = \mathbf{c}^{\{S,I\}}, \tag{4.16b}$$

where $c^{\{F,\nu\}} = A^{\{F,\nu\}}\mathbb{1}^{s^{\{SF\}}}$, $c^{\{S,\nu\}} = \mathbf{A}^{\{S,\nu\}}\mathbb{1}^{s^{\{S\}}}$, $\nu \in \{I, E, F\}$, *if the following conditions hold:*

$$\Omega^{\{0\}}\mathbb{1}^{s^{\{S\}}} = c^{\{S\}}, \tag{4.17a}$$

$$\Omega^{\{k\}}\mathbb{1}^{s^{\{S\}}} = 0^{s^{\{S\}}},\ k = 1, ..., n_\Omega - 1, \tag{4.17b}$$

$$\Gamma\mathbb{1}^{s^{\{S\}}} = 0^{s^{\{S\}}}, \tag{4.17c}$$

where $0^{s^{\{S\}}}$ *is a vector of zeros with length* $s^{\{S\}}$.

Proof. Computing the fast GARK abscissae from the respective fast GARK tables,

$$\mathbf{c}^{\{F,F\}} = \mathbf{A}^{\{F,F\}}\mathbb{1}^{s^{\{SF\}}} = (C^{\{S\}} \otimes A^{\{F\}})\mathbb{1}^{s^{\{SF\}}} = c^{\{S\}} \otimes c^{\{F\}},$$

$$\mathbf{c}^{\{F,E\}} = \mathbf{A}^{\{F,E\}}\mathbb{1}^{s^{\{S\}}} = \sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\}\times k}\mathbb{1}^{s^{\{S\}}}$$

$$= (\Omega^{\{0\}}\mathbb{1}^{s^{\{S\}}}) \otimes c^{\{F\}} + \sum_{k=1}^{n_\Omega - 1} (\Omega^{\{k\}}\mathbb{1}^{s^{\{S\}}}) \otimes A^{\{F\}} c^{\{F\}\times k},$$

$$\mathbf{c}^{\{F,I\}} = \mathbf{A}^{\{F,I\}}\mathbb{1}^{s^{\{S\}}} = \mathbf{A}^{\{F,E\}}\mathbb{1}^{s^{\{S\}}} = \mathbf{c}^{\{F,E\}},$$

thus $\mathbf{c}^{\{F,F\}} = \mathbf{c}^{\{F,I\}} = \mathbf{c}^{\{F,E\}}$ when $\Omega^0\mathbb{1}^{s^{\{S\}}} = c^{\{S\}}$ and $\Omega^{\{k\}}\mathbb{1}^{s^{\{S\}}} = 0^{s^{\{S\}}},\ k = 1, ..., n_\Omega - 1$. Computing the slow GARK abscissae from the slow GARK tables,

$$\mathbf{c}^{\{S,F\}} = \mathbf{A}^{\{S,F\}}\mathbb{1}^{s^{\{SF\}}} = (C^{\{S\}} \otimes b^{\{F\}})\mathbb{1}^{s^{\{SF\}}} = c^{\{S\}},$$

$$\mathbf{c}^{\{S,E\}} = \mathbf{A}^{\{S,E\}}\mathbb{1}^{s^{\{S\}}} = \overline{\Omega}\mathbb{1}^{s^{\{S\}}} = \sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}}\mathbb{1}^{s^{\{S\}}} \frac{1}{k+1},$$

$$\mathbf{c}^{\{S,E\}} = \mathbf{A}^{\{S,I\}}\mathbb{1}^{s^{\{S\}}} = (\overline{\Omega} + \Gamma)\mathbb{1}^{s^{\{S\}}} = \sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}}\mathbb{1}^{s^{\{S\}}} \frac{1}{k+1} + \Gamma\mathbb{1}^{s^{\{S\}}},$$

and thus $\mathbf{c}^{\{S,F\}} = \mathbf{c}^{\{S,I\}} = \mathbf{c}^{\{S,E\}}$ is satisfied using the same conditions as above, with the additional constraint that $\Gamma\mathbb{1}^{s^{\{S\}}} = 0^{s^{\{S\}}}$. □

Remark 4.2.3 (Second order conditions)**.** *When the tables that comprise a GARK method are each at least second-order accurate and internal consistency holds, there are no additional second-order coupling conditions. Thus, the internal consistency conditions act as second-order conditions when the slow base method and arbitrary fast method are at least second-order accurate.*

4.2.4. Higher Order Conditions

Theorem 4.2.2 (Third Order Conditions)**.** *An internally-consistent IMEX-MRI-SR method with third-order accurate slow base method and with fast method of order* $\max(3, n_\Omega + 1)$ *accurate is third-order accurate if the following condition holds:*

$$e_{s^{\{S\}}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) c^{\{S\}} = \frac{1}{6}. \tag{4.18}$$

Proof. From [5], a internally consistent GARK method of this structure with a third-order accurate slow base method and an order $\max(3, n_\Omega + 1)$ fast method has four third-order coupling conditions,

$$\mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{S,F\}}\mathbf{c}^{\{F\}} = \frac{1}{6}, \tag{4.19a}$$

$$\mathbf{b}^{\{F\},T}\mathbf{A}^{\{S,\sigma\}}\mathbf{c}^{\{S\}} = \frac{1}{6}, \tag{4.19b}$$

for $\sigma \in \{I, E\}$. An internally-consistent IMEX-MRI-SR method has

$$\mathbf{c}^{\{F\}} = \mathbf{c}^{\{F,F\}} = \mathbf{c}^{\{F,E\}} = \mathbf{c}^{\{F,I\}} = c^{\{S\}} \otimes c^{\{F\}}, \tag{4.20a}$$

$$\mathbf{c}^{\{S\}} = \mathbf{c}^{\{S,F\}} = \mathbf{c}^{\{S,E\}} = \mathbf{c}^{\{S,I\}} = c^{\{E\}} = c^{\{I\}} = c^{\{S\}}. \tag{4.20b}$$

Conditions (4.19a) are automatically satisfied for both values of σ:

$$\begin{aligned}\frac{1}{6} &= \mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{S,F\}}\mathbf{c}^{\{F\}} \\ &= b^{\{\sigma\},T}(C^{\{S\}} \otimes b^{\{F\},T})(c^{\{S\}} \otimes c^{\{F\}}) = \frac{1}{3} \cdot \frac{1}{2}\end{aligned}$$

Conditions (4.19b) reduce to the single condition (4.18) because $\mathbf{A}^{\{F,E\}} = \mathbf{A}^{\{F,I\}}$ for an IMEX-MRI-SR method:

$$\begin{aligned}\frac{1}{6} = \mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,\sigma\}}\mathbf{c}^{\{S\}} &= (e_{s^{\{S\}}}^T \otimes b^{\{F\},T}) \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\} \times k} \right) c^{\{S\}} \\ &= e_{s^{\{S\}}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) c^{\{S\}}.\end{aligned}$$

□

Theorem 4.2.3 (Fourth Order Conditions)**.** *An IMEX-MRI-SR method satisfying Theorem 4.2.2 is fourth-order accurate if the slow base method is fourth-order accurate, the arbitrary fast method is order* $\max(4, n_\Omega + 2)$ *accurate, and the following conditions hold:*

$$e_{s\{S\}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+3)} \right) c^{\{S\}} = \frac{1}{8}, \tag{4.21a}$$

$$e_{s\{S\}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) C^{\{S\}} c^{\{S\}} = \frac{1}{12}, \tag{4.21b}$$

$$e_{s\{S\}}^T \Gamma C^{\{S\}} \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) c^{\{S\}} = 0, \tag{4.21c}$$

$$e_{s\{S\}}^T \overline{\Omega} C^{\{S\}} \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) c^{\{S\}} = \frac{1}{24}, \tag{4.21d}$$

$$e_{s\{S\}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) \overline{\Omega} c^{\{S\}} = \frac{1}{24}, \tag{4.21e}$$

$$e_{s\{S\}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) \Gamma c^{\{S\}} = 0. \tag{4.21f}$$

Proof. From [5], there are 26 fourth-order coupling conditions for a third-order GARK method with this structure; this further reduces to 21 when $\mathbf{A}^{\{F,E\}} = \mathbf{A}^{\{F,I\}}$. These GARK coupling conditions are:

$$\mathbf{b}^{\{\sigma\},T} \mathbf{C}^{\{S\}} \mathbf{A}^{\{S,F\}} \mathbf{c}^{\{F\}} = \frac{1}{8} \tag{4.22a}$$

$$\mathbf{b}^{\{\sigma\},T} \mathbf{A}^{\{S,\nu\}} \mathbf{A}^{\{S,F\}} \mathbf{c}^{\{F\}} = \frac{1}{24} \tag{4.22b}$$

$$\mathbf{b}^{\{\sigma\},T} \mathbf{A}^{\{S,F\}} \mathbf{C}^{\{F\}} \mathbf{c}^{\{F\}} = \frac{1}{12} \tag{4.22c}$$

$$\mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{S,F\}}\mathbf{A}^{\{F,F\}}\mathbf{c}^{\{F\}} = \frac{1}{24} \tag{4.22d}$$

$$\mathbf{b}^{\{F\},T}\mathbf{C}^{\{F\}}\mathbf{A}^{\{F,\sigma\}}\mathbf{c}^{\{S\}} = \frac{1}{8} \tag{4.22e}$$

$$\mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,\sigma\}}\mathbf{C}^{\{S\}}\mathbf{c}^{\{S\}} = \frac{1}{12} \tag{4.22f}$$

$$\mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{S,F\}}\mathbf{A}^{\{F,\nu\}}\mathbf{c}^{\{S\}} = \frac{1}{24} \tag{4.22g}$$

$$\mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,F\}}\mathbf{A}^{\{F,\sigma\}}\mathbf{c}^{\{S\}} = \frac{1}{24} \tag{4.22h}$$

$$\mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,\sigma\}}\mathbf{A}^{\{S,\nu\}}\mathbf{c}^{\{S\}} = \frac{1}{24} \tag{4.22i}$$

$$\mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,\sigma\}}\mathbf{A}^{\{S,F\}}\mathbf{c}^{\{F\}} = \frac{1}{24} \tag{4.22j}$$

for $\sigma, \nu \in \{I, E\}$. $\mathbf{c}^{\{F\}}$ and $\mathbf{c}^{\{S\}}$ are defined as in (4.20a) and (4.20b), respectively. We arrive at the conditions (4.21) by checking each of the conditions (4.22) in turn.

The first four conditions (4.22a)-(4.22d) are automatically satisfied:

$$\begin{aligned}
\frac{1}{8} &= \mathbf{b}^{\{\sigma\},T}\mathbf{C}^{\{S\}}\mathbf{A}^{\{S,F\}}\mathbf{c}^{\{F\}} \\
&= b^{\{\sigma\},T}C^{\{S\}}(C^{\{S\}} \otimes b^{\{F\},T})(c^{\{S\}} \otimes c^{\{F\}}) \\
&= b^{\{\sigma\},T}C^{\{S\}}C^{\{S\}}c^{\{S\}}\frac{1}{2} = \frac{1}{4}\cdot\frac{1}{2}, \\
\frac{1}{24} &= \mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{S,\nu\}}\mathbf{A}^{\{S,F\}}\mathbf{c}^{\{F\}} \\
&= b^{\{\sigma\},T}A^{\{\nu\}}(C^{\{S\}} \otimes b^{\{F\},T})(c^{\{S\}} \otimes c^{\{F\}}) \\
&= b^{\{\sigma\},T}A^{\{\nu\}}C^{\{S\}}c^{\{S\}}\frac{1}{2} = \frac{1}{12}\cdot\frac{1}{2}, \\
\frac{1}{12} &= \mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{S,F\}}\mathbf{C}^{\{F\}}\mathbf{c}^{\{F\}}
\end{aligned}$$

$$= b^{\{\sigma\},T}(C^{\{S\}} \otimes b^{\{F\},T}) \operatorname{diag}(c^{\{S\}} \otimes c^{\{F\}})(c^{\{S\}} \otimes c^{\{F\}})$$

$$= b^{\{\sigma\},T} C^{\{S\}} C^{\{S\}} c^{\{S\}} \frac{1}{3} = \frac{1}{4} \cdot \frac{1}{3},$$

$$\frac{1}{24} = \mathbf{b}^{\{\sigma\},T} \mathbf{A}^{\{S,F\}} \mathbf{A}^{\{F,F\}} \mathbf{c}^{\{F\}}$$

$$= b^{\{\sigma\},T}(C^{\{S\}} \otimes b^{\{F\},T})(C^{\{S\}} \otimes A^{\{F\}})(c^{\{S\}} \otimes c^{\{F\}})$$

$$= b^{\{\sigma\},T} C^{\{S\}} C^{\{S\}} c^{\{S\}} \frac{1}{6} = \frac{1}{4} \cdot \frac{1}{6}.$$

Condition (4.22e) is not automatically satisfied and simplifies to (4.21a):

$$\frac{1}{8} = \mathbf{b}^{\{F\},T} \mathbf{C}^{\{F\}} \mathbf{A}^{\{F,\sigma\}} \mathbf{c}^{\{S\}}$$

$$= (e_{s^{\{S\}}} \otimes b^{\{F\}})^T \operatorname{diag}(c^{\{S\}} \otimes c^{\{F\}}) \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\} \times k} \right) c^{\{S\}}$$

$$= e_{s^{\{S\}}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+3)} \right) c^{\{S\}}.$$

Condition (4.22f) is not automatically satisfied and simplifies to (4.21b).

$$\frac{1}{12} = \mathbf{b}^{\{F\},T} \mathbf{A}^{\{F,\sigma\}} \mathbf{C}^{\{S\}} \mathbf{c}^{\{S\}}$$

$$= (e_{s^{\{S\}}} \otimes b^{\{F\}})^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\} \times k} \right) C^{\{S\}} c^{\{S\}}$$

$$= e_{s^{\{S\}}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) C^{\{S\}} c^{\{S\}}.$$

Condition (4.22g) is not automatically satisfied and reduces to (4.21c) and (4.21d) when $\sigma = I, E$, respectively, and both conditions are enforced simultaneously,

$$\begin{aligned}\frac{1}{24} &= \mathbf{b}^{\{\sigma\},T}\mathbf{A}^{\{S,F\}}\mathbf{A}^{\{F,\nu\}}\mathbf{c}^{\{S\}} \\ &= b^{\{\sigma\},T}(C^{\{S\}} \otimes b^{\{F\}}) \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\}\times k} \right) c^{\{S\}} \\ &= b^{\{\sigma\},T} C^{\{S\}} \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) c^{\{S\}}.\end{aligned}$$

Condition (4.22h) is not automatically satisfied but simplifies to (4.18) minus (4.21a),

$$\begin{aligned}\frac{1}{24} &= \mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,F\}}\mathbf{A}^{\{F,\sigma\}}\mathbf{c}^{\{S\}} \\ &= (e_{s^{\{S\}}} \otimes b^{\{F\}})^T (C^{\{S\}} \otimes A^{\{F\}}) \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\}\times k} \right) c^{\{S\}} \\ &= e_{s^{\{S\}}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)(k+3)} \right) c^{\{S\}}.\end{aligned}$$

Condition (4.22i) is not automatically satisfied and reduces to (4.21e) and (4.21f) when $\nu = I, E$, respectively, and both conditions are enforced simultaneously:

$$\begin{aligned}\frac{1}{24} &= \mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,\sigma\}}\mathbf{A}^{\{S,\nu\}}\mathbf{c}^{\{S\}} \\ &= (e_{s^{\{S\}}} \otimes b^{\{F\}})^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\}\times k} \right) \mathbf{A}^{\{S,\nu\}} c^{\{S\}} \\ &= e_{s^{\{S\}}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) \mathbf{A}^{\{S,\nu\}} c^{\{S\}}.\end{aligned}$$

Finally, condition (4.22j) simplifies to the same condition as (4.21b),

$$\begin{aligned}\frac{1}{24} &= \mathbf{b}^{\{F\},T}\mathbf{A}^{\{F,\sigma\}}\mathbf{A}^{\{S,F\}}\mathbf{c}^{\{F\}} \\ &= (e_{s^{\{S\}}} \otimes b^{\{F\}}) \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \otimes A^{\{F\}} c^{\{F\} \times k} \right) (C^{\{S\}} \otimes b^{\{F\}})(c^{\{S\}} \otimes c^{\{F\}}) \\ &= e_{s^{\{S\}}}^T \left(\sum_{k=0}^{n_\Omega - 1} \Omega^{\{k\}} \frac{1}{(k+1)(k+2)} \right) C^{\{S\}} c^{\{S\}} \frac{1}{2}.\end{aligned}$$

□

Theorem 4.2.4 (Minimum n_Ω for third-order accuracy)**.** *An IMEX-MRI-SR requires at least $n_\Omega = 2$ for third-order accuracy.*

Proof. An IMEX-MRI-SR method with one Ω matrix, $\Omega^{\{0\}}$, has $\mathbf{A}^{\{S,E\}} = \overline{\Omega} = \Omega^0$ and, from the FSAL property, $\mathbf{b}^{\{E\},T} = e_{s^{\{S\}}}^T \overline{\Omega} = e_{s^{\{S\}}}^T \Omega^{\{0\}}$. For an IMEX-MRI-SR method of order p, we assume the slow-explicit base method also satisfies all standard Runge–Kutta order conditions up to order p. Thus, a third-order IMEX-MRI-SR slow-explicit base method with one Ω satisfies the second order condition $b^{\{E\},T} c^{\{S\}} = \frac{1}{2}$, which simplifies to

$$\frac{1}{2} = \sum_{j=1}^{s^{\{S\}}} \omega_{s^{\{S\}},j}^{\{0\}} c_j^{\{S\}}.$$

However, the third-order IMEX-MRI-SR coupling condition (4.18) simplifies to

$$\frac{1}{3} = \sum_{j=1}^{s^{\{S\}}} \omega_{s^{\{S\}},j}^{\{0\}} c_j^{\{S\}}.$$

Since these cannot hold simultaneously, a third-order IMEX-MRI-SR method with one Ω matrix is not possible. There are no such mutually exclusive conditions for $n_\Omega > 1$, and in Section 4.4 we introduce third and a fourth order methods with $n_\Omega = 2$. □

4.3. Linear Stability

As in [5, 45], we analyze the stability of IMEX-MRI-SR methods when applied to the linear, scalar test problem

$$y'(t) = \lambda^{\{F\}} y + \lambda^{\{E\}} y + \lambda^{\{I\}} y, \quad t \geq 0, \quad y(0) = 1, \tag{4.23}$$

with $\lambda^{\{F\}}, \lambda^{\{E\}}, \lambda^{\{I\}} \in \mathbb{C}^-$. For convenience, we additionally define $z^{\{F\}} = H\lambda^{\{F\}}$, $z^{\{E\}} = H\lambda^{\{E\}}$, and $z^{\{I\}} = H\lambda^{\{I\}}$. When applied to (4.23), the IVP (4.5b) becomes

$$\begin{aligned} v_i'(\theta) &= \lambda^{\{F\}} v_i(\theta) + \frac{1}{c_i^{\{S\}}} \sum_{j=1}^{i-1} \omega_{i,j} \left(\frac{\theta}{c_i^{\{S\}} H} \right) (\lambda^{\{E\}} Y_j^{\{S\}} + \lambda^{\{I\}} Y_j^{\{S\}}) \\ &= \lambda^{\{F\}} v_i(\theta) + \frac{1}{c_i^{\{S\}}} \sum_{j=1}^{i-1} \sum_{k=0}^{n_\Omega - 1} \omega_{i,j}^{\{k\}} \left(\frac{\theta}{c_i^{\{S\}} H} \right)^k (\lambda^{\{E\}} Y_j^{\{S\}} + \lambda^{\{I\}} Y_j^{\{S\}}) \end{aligned}$$

for $i = 2, ..., s^{\{S\}}$, with $\theta \in [0, c_i^{\{S\}} H]$ and $v_i(0) = y_n$. The solution to this at $\theta = c_i^{\{S\}} H$ is

$$\begin{aligned} v_i(c_i^{\{S\}} H) &= e^{c_i^{\{S\}} z^{\{F\}}} y_n + (z^{\{E\}} + z^{\{I\}}) \sum_{j=1}^{i-1} \sum_{k=0}^{n_\Omega - 1} \omega_{i,j}^{\{k\}} \left(\int_0^1 e^{c_i^{\{S\}} z^{\{F\}} (1-t)} t^k \mathrm{d}t \right) Y_j^{\{S\}} \\ &= \varphi_0(c_i^{\{S\}} z^{\{F\}}) y_n + (z^{\{E\}} + z^{\{I\}}) \sum_{j=1}^{i-1} \eta_{i,j}(z^{\{F\}}) Y_j^{\{S\}}. \end{aligned}$$

Here we define η as in [5] as a function of the fast eigenvalue $z^{\{F\}}$,

$$\eta_{i,j}(z^{\{F\}}) = \sum_{k=0}^{n_\Omega - 1} \omega_{i,j}^{\{k\}} \varphi_{k+1}(c_i^{\{S\}} z^{\{F\}}), \tag{4.24}$$

where the family of analytical functions $\{\varphi_k\}$ are given by [45],

$$\varphi_0(z) = e^z, \quad \varphi_k(z) = \int_0^1 e^{z(1-t)} t^{k-1} \mathrm{d}t, \quad k \geq 1.$$

The i^{th} IMEX-MRI-SR stage for the stability problem (4.23) then becomes

$$Y_i^{\{S\}} = \varphi_0(c_i^{\{S\}} z^{\{F\}}) y_n + (z^{\{E\}} + z^{\{I\}}) \sum_{j=1}^{i-1} \eta_{i,j}(z^{\{F\}}) Y_j^{\{S\}} + z^{\{I\}} \sum_{j=1}^{i} \gamma_{i,j} Y_j^{\{S\}}. \tag{4.25}$$

Concatenating $Y = [Y_1^{\{S\}} \ \cdots \ Y_{s^{\{S\}}}^{\{S\}}]^T$ and writing (4.25) in matrix form,

$$\begin{aligned} Y &= \varphi_0(c^{\{S\}} z^{\{F\}}) y_n + (z^{\{E\}} + z^{\{I\}}) \eta(z^{\{F\}}) Y + z^{\{I\}} \Gamma Y \\ &= \left(I - (z^{\{E\}} + z^{\{I\}}) \eta(z^{\{F\}}) - z^{\{I\}} \Gamma\right)^{-1} \varphi_0(c^{\{S\}} z^{\{F\}}) y_n \end{aligned}$$

where

$$\eta(z^{\{F\}}) = \sum_{k=0}^{n_\Omega - 1} \operatorname{diag}(\varphi_{k+1}(c^{\{S\}} z^{\{F\}})) \Omega^{\{k\}}.$$

Thus the stability function for an IMEX-MRI-SR method applied to (4.23) is

$$\begin{aligned} R(z^{\{F\}}, z^{\{E\}}, z^{\{I\}}) = \\ e_{s^{\{S\}}}^T \left(I - (z^{\{E\}} + z^{\{I\}}) \eta(z^{\{F\}}) - z^{\{I\}} \Gamma\right)^{-1} \varphi_0(c^{\{S\}} z^{\{F\}}). \end{aligned} \tag{4.26}$$

We consider a few definitions of joint stability for IMEX-MRI-SR methods. As with IMEX-MRI-GARK methods, we consider a region that incorporates all three z,

$$\mathcal{J}_{\alpha,\rho,\beta,\xi} = \{z^{\{E\}} \in \mathbb{C}^- : |R(z^{\{F\}}, z^{\{E\}}, z^{\{I\}})| \leq 1, \ \forall z^{\{F\}} \in \mathcal{S}_{\alpha,\rho}^{\{F\}}, \ \forall z^{\{I\}} \in \mathcal{S}_{\beta,\xi}^{\{I\}}\}, \tag{4.27}$$

where $\mathcal{S}_{\alpha,\rho}^{\{\sigma\}} = \{z^{\{\sigma\}} \in \mathbb{C}^- : |\arg(z^{\{\sigma\}}) - \pi| \leq \alpha, \ |z^{\{\sigma\}}| \leq \rho\}$. We note that this definition of joint stability may be overly-restrictive, as it demands the implicit and fast parts of the method to be A(β)- and A(α)-stable, respectively, for any joint stability region to exist. Thus, we also analyze the implicit and explicit stability regions of IMEX-MRI-SR methods

independently, with stability regions defined as:

$$\mathcal{J}^{\{I\}}_{\alpha,\rho} = \{z^{\{I\}} \in \mathbb{C}^- : |R(z^{\{F\}}, 0, z^{\{I\}})| \leq 1,\ \forall z^{\{F\}} \in \mathcal{S}^{\{F\}}_{\alpha,\rho}\}, \tag{4.28}$$

$$\mathcal{J}^{\{E\}}_{\alpha,\rho} = \{z^{\{E\}} \in \mathbb{C}^- : |R(z^{\{F\}}, z^{\{E\}}, 0)| \leq 1,\ \forall z^{\{F\}} \in \mathcal{S}^{\{F\}}_{\alpha,\rho}\}. \tag{4.29}$$

These are more consistent with standard stability analyses of ARK methods, wherein explicit and implicit stability are considered separately.

4.4. Example Methods

We introduce three IMEX-MRI-SR methods of orders 2, 3 and 4, each of which includes an embedding with accuracy one order lower (1, 2 and 3, resp.) for temporal error estimation. We note that we designed the embedding coefficients with efficiency in mind, so that computation of the embedded solutions do not require an additional implicit solve at the slow time scale (i.e., $\hat{\gamma}_{s^{\{S\}}} = 0$).

When presenting the coefficients for each method we use the notation

$$\Omega^{\{k\}} = \left[\begin{array}{c} \boldsymbol{\Omega}^{\{\mathbf{k}\}} \\ \hline \hat{\omega}^{\{k\}} \end{array}\right],\ \Gamma = \left[\begin{array}{c} \boldsymbol{\Gamma} \\ \hline \hat{\gamma} \end{array}\right],$$

where $\boldsymbol{\Omega}^{\{\mathbf{k}\}}$ and $\boldsymbol{\Gamma}$ are the matrices of coefficients defining the primary method, and $\hat{\omega}^{\{k\}}$ and $\hat{\gamma}$ are the embedding row of coefficients.

4.4.1. IMEX-MRI-SR2(1)

The first method is second-order with a first-order embedding. It has 4 stages, $n_\Omega = 1$, and requires 3 slow nonlinear solves per step. The coefficients can be found in Appending B.1.

To create this method, we used the free coefficients of the primary method to maximize the size of the stability region defined by (4.27). We used the free coefficients of the embedded method to minimize the value of the C-statistic from Prince and Dormand [38],

$$C^{(p+1)} = \frac{\left\|\hat{\tau}^{(p+1)} - \tau^{(p+1)}\right\|_2}{\left\|\hat{\tau}^{(p)}\right\|_2}, \tag{4.30}$$

where $\tau^{(p)}$ is the vector of order condition residuals at order p for the primary method, $\hat{\tau}^{(p)}$ is the vector of order condition residuals at order p for the embedded method, and p is the order of the primary method (in this case, $p = 2$). The C-statistic gives an estimate of how much the error stemming from unsatisfied $(p + 1)$-order conditions of the primary method corrupts the error estimate provided by the embedded method.

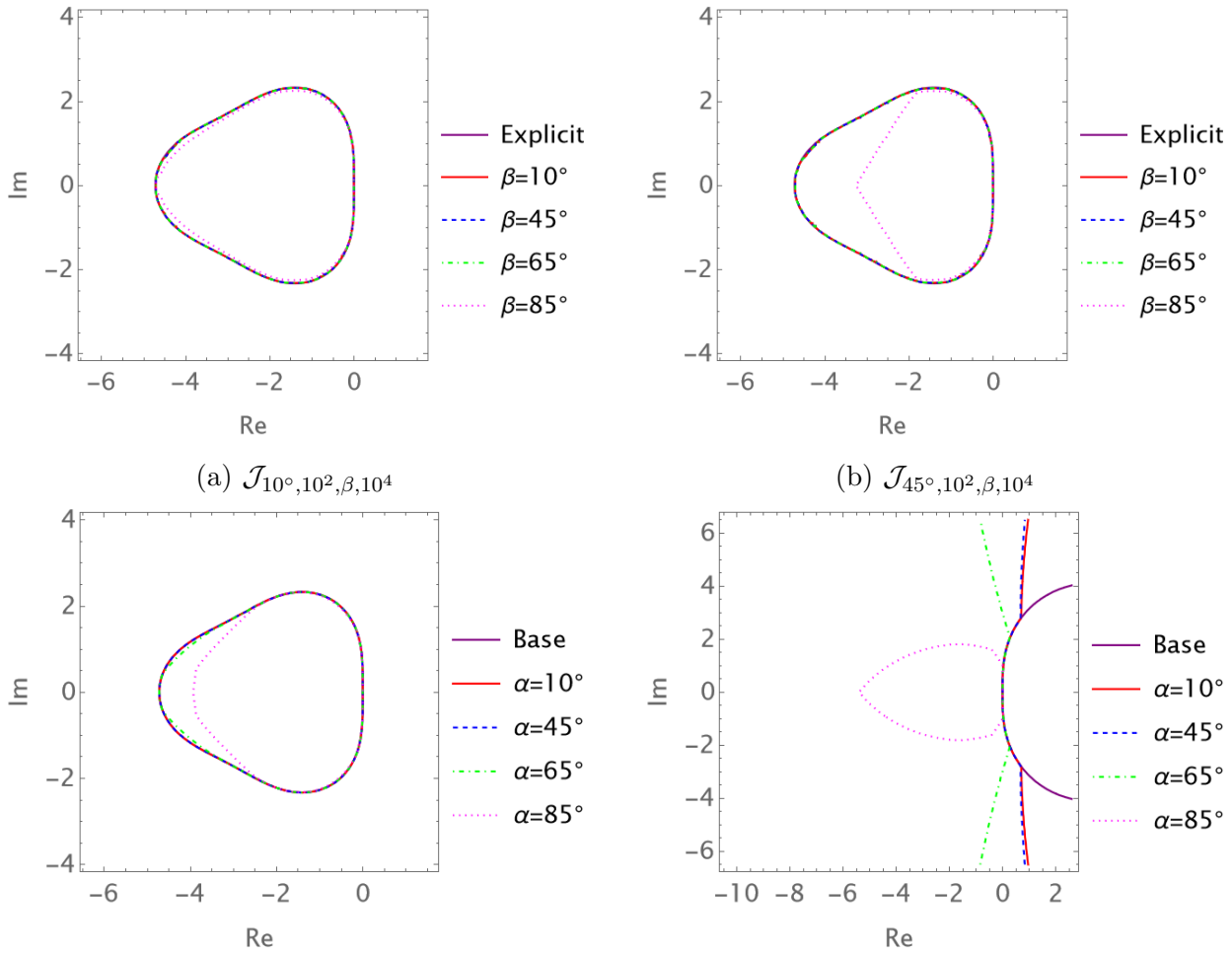
This method has large, robust stability regions. Figures 4.1a and 4.1b show $\mathcal{J}_{\alpha,10^2,\beta,10^4}$ for $\alpha \in \{10°, 45°\}$ and varying β, along with the explicit slow base method's stability region. We see that the multirate method has a stability region essentially identical to the explicit slow base method when $\alpha = 10°$ for any value of $z^{\{I\}}$. When α grows to 45° we can see a decay of the stability region associated with the $\beta = 85°$, while the regions associated with smaller β values experience negligible decay, if any.

Figure 4.1c shows $\mathcal{J}^{\{E\}}_{\alpha,10^2}$ for varying α. We see that when $z^{\{I\}} = 0$, the stability region is again large, with only a slight decay in area when $\alpha = 85°$.

Figure 4.1c shows $\mathcal{J}^{\{I\}}_{\alpha,10^2}$ for varying α. When $z^{\{E\}} = 0$, the multirate method is A-stable for $0° \leq \alpha \leq 45°$. When α grows further to 65°, the region decays slightly but is still approximately A(80°)-stable. When $\alpha = 85°$, the stability region decays to an enclosed bubble similar to the regions in Figure 4.1c.

4.4.2. IMEX-MRI-SR3(2)

Our second method is third-order with a second-order embedding. It has 5 stages, $n_\Omega = 2$, and requires 4 nonlinear solves per step. The coefficients can be found in Appendix B.2.



(a) $\mathcal{J}_{10^\circ,10^2,\beta,10^4}$ (b) $\mathcal{J}_{45^\circ,10^2,\beta,10^4}$

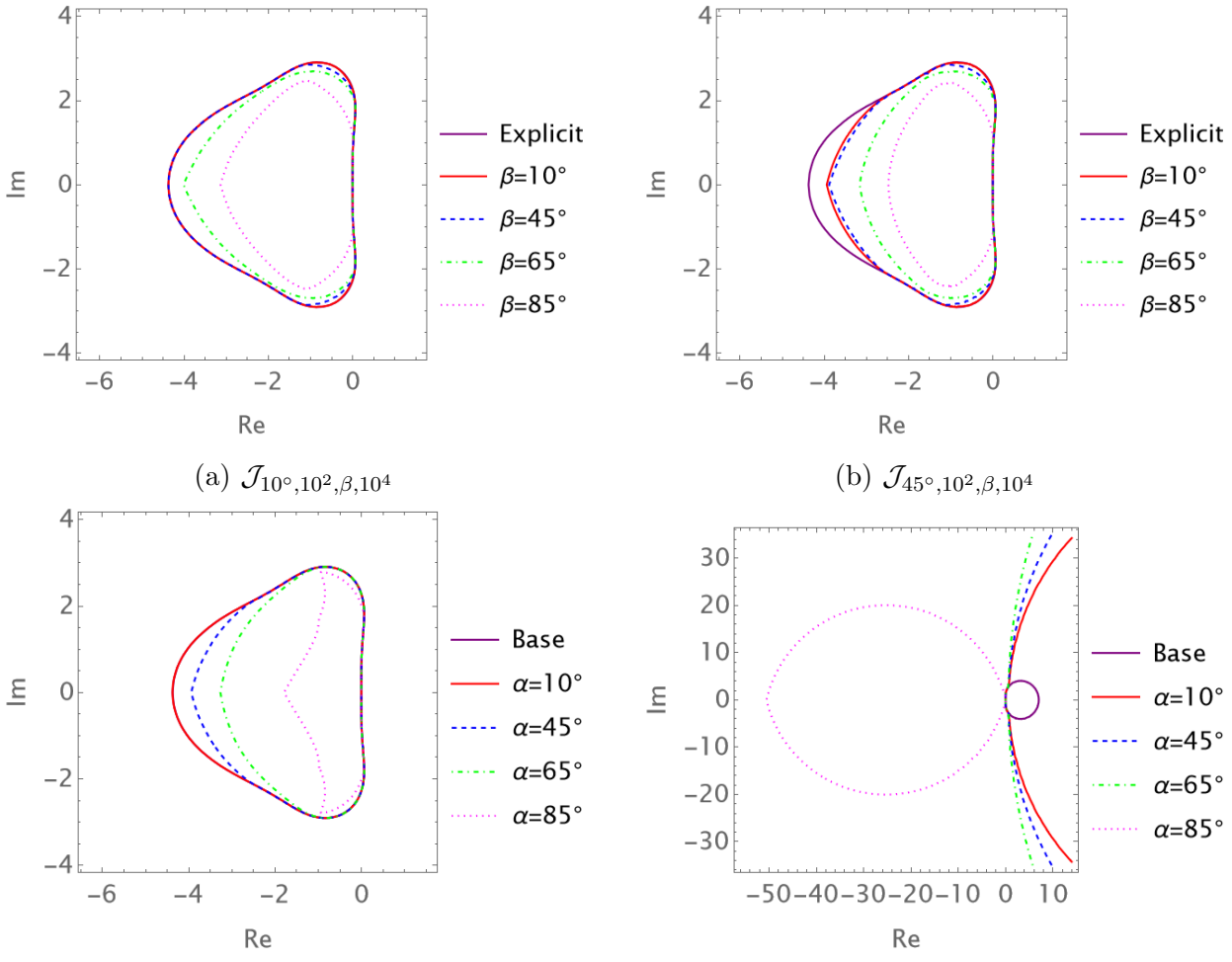
(c) $\mathcal{J}^{\{E\}}_{\alpha,10^2}$ (d) $\mathcal{J}^{\{I\}}_{\alpha,10^2}$

Figure 4.1: Joint Stability Regions for IMEX-MRI-SR2(1)

To create this method, we used the free coefficients for both the method and embedding as described in Section 4.4.1, this time using the C-statistic (4.30) with $p = 3$.

Figures 4.2a and 4.2b show the relatively large joint stability regions $\mathcal{J}_{\alpha,10^2,\beta,10^4}$ for $\alpha = \{10^\circ, 45^\circ\}$, respectively, along with the explicit slow base method's stability region. Again, at $\alpha = 10^\circ$ and lower values of β, the method is essentially as stable as the explicit slow base method. The areas of these stability regions decrease for higher values of β. When α grows to 45° in Figure 4.2b, the stability regions shrink slightly in comparison with $\alpha = 10^\circ$ from Figure 4.2a.

Figures 4.2c and 4.2d show $\mathcal{J}^{\{E\}}_{\alpha,10^2}$ and $\mathcal{J}^{\{I\}}_{\alpha,10^2}$ for varying α, respectively. We see that when $z^{\{I\}} = 0$, the stability region is reasonably large for smaller value of α, but the region shrinks as α grows; however, even for $\alpha = 85°$, the region retains a good extent along the imaginary axis. Similar to the second-order method, when $z^{\{E\}} = 0$, the stability region is A-stable for most values of α, only losing A-stability for $\alpha = 85°$, where the region decays to a rather large, yet enclosed, bubble.



(a) $\mathcal{J}_{10°,10^2,\beta,10^4}$

(b) $\mathcal{J}_{45°,10^2,\beta,10^4}$

(c) $\mathcal{J}^{\{E\}}_{\alpha,10^2}$

(d) $\mathcal{J}^{\{I\}}_{\alpha,10^2}$

Figure 4.2: Joint Stability Regions for IMEX-MRI-SR3(2)

4.4.3. IMEX-MRI-SR4(3)

Our final method is fourth-order with a third-order embedding, has 7 stages, $n_\Omega = 2$, and requires 5 nonlinear solves per step. The coefficients can be found in Appendix B.3.

Due to the number of order conditions involved when simultaneously solving all IMEX-MRI-SR coupling conditions and base ARK order conditions at fourth-order, we based this method off of an existing 4(3) ARK method, LIRK4 [3]. Since this was not stiffly accurate, then as discussed in Remark 4.2.2 we padded the $A^{\{E\}}$ and $A^{\{I\}}$ matrices with the b vector, i.e.,

$$\mathbf{A}^{\{S,E\}} = \begin{bmatrix} A^{\{E\}} & 0^{\{6\}} \\ b^T & 0 \end{bmatrix}, \quad \mathbf{A}^{\{S,I\}} = \begin{bmatrix} A^{\{I\}} & 0^{\{6\}} \\ b^T & 0 \end{bmatrix}, \tag{4.31}$$

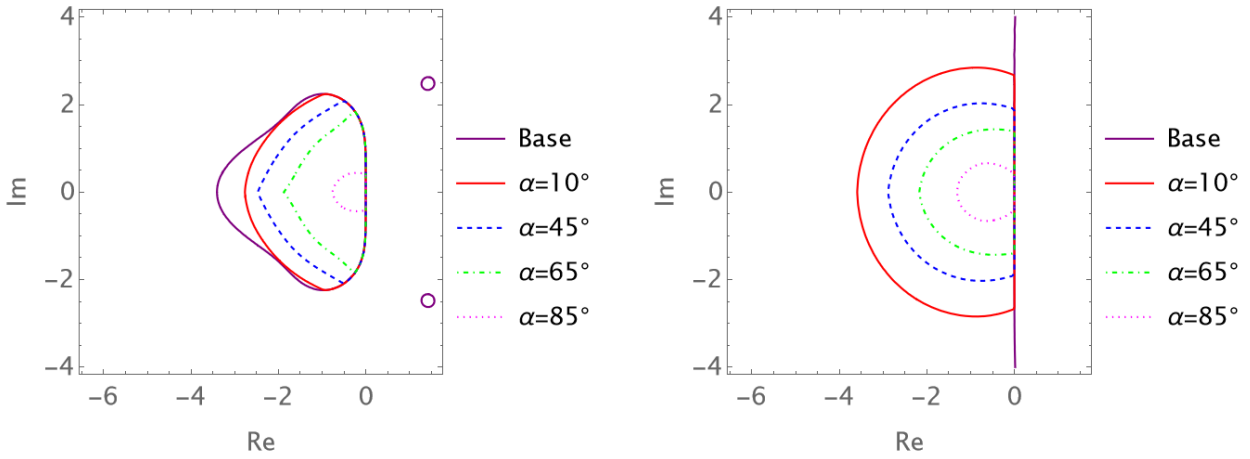
where $0^{\{6\}} \in \mathbb{R}^6$ is all zero.

Since the last row of $\mathbf{A}^{\{S,E\}}$ and $\mathbf{A}^{\{S,I\}}$ both equal b^T, the last row of Γ equals zero, as follows from the definitions (4.13b) and (4.13c). When $\mathbf{A}^{\{S,I\}}$ has a zero in the bottom right entry (and therefore Γ has a zero in the bottom right entry, from (4.13c)), there is no nonlinear solve required to compute this last stage and therefore the updated time step solution. We believe that this negatively affects stability, as we will show in Figure 4.3.

As before, we used the free variables of the IMEX-MRI-SR method to optimize stability. This method had an empty joint stability region defined by (4.27) for all of our attempts to choose or optimize values of the free variables, so we instead optimized the size of the stability region defined by (4.29). To optimize the embedded method, we minimized the 2-norm of the fourth-order condition residuals, $\|\hat{\tau}^{(4)}\|_2$, to reduce the overall error in the embedded solution. We note that our previous approach of minimizing the C-statistic (4.30) was not possible since we do not yet have the fifth-order IMEX-MRI-SR coupling conditions.

Figure 4.3a shows the stability regions $\mathcal{J}_{\alpha,1}^{\{E\}}$ for varying α. Unlike the lower-order methods, this method's explicit stability region never fully matches that of the explicit base method. Similarly to the other methods, as α grows the stability region shrinks.

Figure 4.3b shows the stability regions $\mathcal{J}_{\alpha,1}^{\{I\}}$ for varying α. Again unlike the lower-order methods, these regions are not A-stable for $\alpha \neq 0$. Because $\mathcal{J}_{\alpha,\rho}^{\{I\}}$ is never A-stable for this method, the stability region (4.27) is always empty. We suspect that this is primarily due to the lack of an implicit solve in the last stage of the method. In future work we plan to investigate this issue in more detail to better understand the factors that contribute to IMEX-MRI-SR joint stability.



(a) $\mathcal{J}_{\alpha,10^2}^{\{E\}}$ (b) $\mathcal{J}_{\alpha,10^2}^{\{I\}}$

Figure 4.3: Joint Stability Regions for IMEX-MRI-SR4(3)

4.5. MERK Methods as Explicit IMEX-MRI-SR Methods

In [31], Luan et al. define MERK methods with orders of accuracy spanning two through five by explicitly defining the abscissae $c_i^{\{S\}}$ and the forcing functions $g_i(\theta)$. Due to their similar structure to IMEX-MRI-SR methods, we may analyze MERK methods using our theory from Section 4.2. Because MERK methods are always explicit, their IMEX-MRI-SR Γ matrices will be all zero. We recall that MERK methods are defined under an assumption that the fast function is linear, $f^{\{F\}}(t,y) = \mathcal{L}y$, but that the slow function can be arbitrary. It is thus natural to assume that MERK methods might only satisfy a subset of the IMEX-MRI-SR order conditions, potentially failing those that handle nonlinearity in $f^{\{F\}}$.

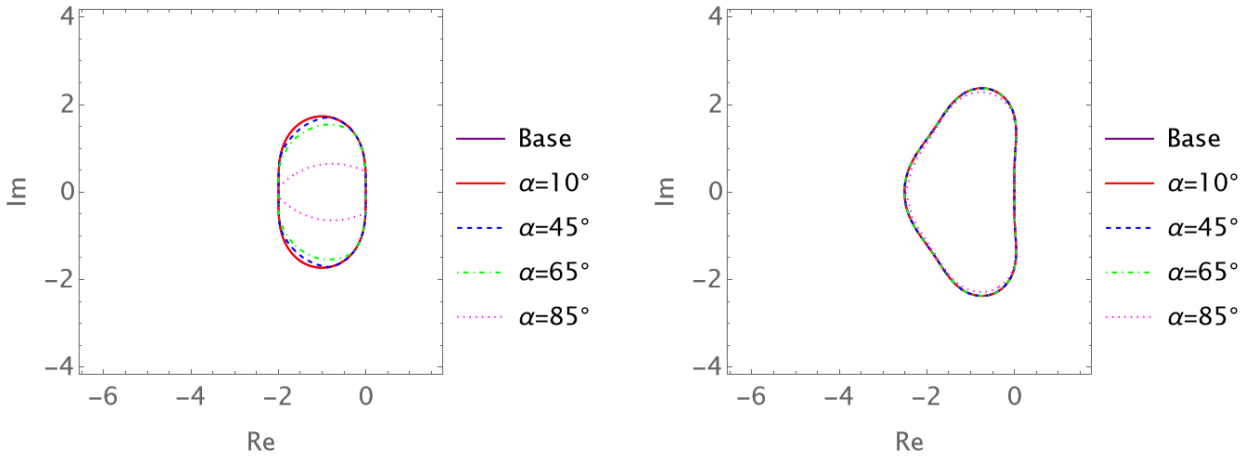
4.5.1. MERK2

Converting the three-stage second-order accurate MERK2 method from [31] into IMEX-MRI-SR form, we have

$$\Omega^{\{0\}} = \begin{bmatrix} 0 & 0 & 0 \\ c_2^{\{S\}} & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \Omega^{\{1\}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1/c_2^{\{S\}} & 1/c_2^{\{S\}} & 0 \end{bmatrix}. \tag{4.32}$$

Interestingly, these coefficients (along with $\Gamma = 0$) satisfy *all* coupling conditions up to third order, and the slow base method determined by $\overline{\Omega}$ satisfies all conditions up to order two. Thus, we expect MERK2 to have second-order accuracy, *even for nonlinear* $f^{\{F\}}$. We confirm this with numerical tests involving nonlinear $f^{\{F\}}$ in Section 4.6, where we use $c_2^{\{S\}} = \frac{1}{2}$ since it was unspecified in [31].

Figure 4.4a shows the stability regions for MERK2. Because this method is explicit, we only plot $\mathcal{J}^{\{E\}}_{\alpha,10^2}$. Notably, these regions nearly match that of the base explicit method for most values of α examined.



(a) MERK2 (b) MERK3

Figure 4.4: $\mathcal{J}^{\{E\}}_{\alpha,10^2}$ Regions for MERK2 and MERK3

4.5.2. MERK3

Converting the four-stage, third-order accurate, MERK3 from [31] to an IMEX-MRI-SR method, we have

$$\Omega^{\{0\}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ c_2^{\{S\}} & 0 & 0 & 0 \\ \frac{2}{3} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \Omega^{\{1\}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{2}{3c_2^{\{S\}}} & \frac{2}{3c_2^{\{S\}}} & 0 & 0 \\ -\frac{3}{2} & 0 & \frac{3}{2} & 0 \end{bmatrix}. \tag{4.33}$$

These satisfy all coupling conditions up to third order, and the corresponding slow base method satisfies all conditions up to order three. Thus similarly to MERK2, we expect it to show third-order accuracy on problems with nonlinear $f^{\{F\}}$. We again confirm this result on numerical tests in Section 4.6 using $c_2^{\{S\}} = 1/2$.

Figure 4.4b shows the stability regions for MERK3, $\mathcal{J}^{\{E\}}_{\alpha,10^2}$. Notably, these regions show no degradation of stability as α is increased.

4.5.3. MERK4

We may express MERK4 as an IMEX-MRI-SR method with 7 stages and $n_\Omega = 3$. The corresponding coefficients are provided in Appendix B.4.

We find that this method satisfies all IMEX-MRI-SR coupling conditions through order four, and its slow base method satisfies all order conditions up through fourth order, so long as $c_6^{\{S\}} = (3 - 4c_5^{\{S\}})/(4 - 6c_5^{\{S\}})$. This restriction on $c_6^{\{S\}}$ is satisfied by the choices in [31] of $c_2^{\{S\}} = 1/2$, $c_3^{\{S\}} = 1/2$, $c_4^{\{S\}} = 1/3$, $c_5^{\{S\}} = 5/6$, and $c_6^{\{S\}} = 1/3$. Thus like before, we expect this method to demonstrate fourth-order accuracy for nonlinear $f^{\{F\}}$, which we confirm numerically in Section 4.6. Interestingly, the joint stability regions $\mathcal{J}^{\{E\}}_{\alpha,10^2}$ for MERK4 are empty. However, given the reliability of this method in [31] and our own results from

Section 4.6, we believe that these empty regions more strongly indicate a deficiency in these definitions of joint stability than any actual issues with MERK4 itself.

4.5.4. MERK5

The IMEX-MRI-SR method that corresponds with MERK5 has 11 stages and $n_\Omega = 4$. These coefficients are given in Appendix B.5. When analyzing this method, we find that when

$$c_9^{\{S\}} = \frac{12 - 15c_{10}^{\{S\}} - 15c_8^{\{S\}} + 20c_{10}^{\{S\}}c_8^{\{S\}}}{15 - 20c_{10}^{\{S\}} - 20c_8^{\{S\}} + 30c_{10}^{\{S\}}c_8^{\{S\}}}$$

the method satisfies all IMEX-MRI-SR coupling conditions up through fourth-order, and its slow base method satisfies all order conditions up through fifth-order. This condition is satisfied by the choice in [31] of $c_2^{\{S\}} = 1/2$, $c_3^{\{S\}} = 1/2$, $c_4^{\{S\}} = 1/3$, $c_5^{\{S\}} = 1/2$, $c_6^{\{S\}} = 1/3$, $c_7^{\{S\}} = 1/4$, $c_8^{\{S\}} = 7/10$, $c_9^{\{S\}} = 1/2$, and $c_{10}^{\{S\}} = 2/3$, therefore we expect it to have at least fourth-order accuracy without restriction on the linearity of $f^{\{F\}}$. As we do not have fifth-order IMEX-MRI-SR order conditions, we cannot check these for MERK5; however, our numerical tests in Section 4.6 indeed show fifth-order convergence for nonlinear $f^{\{F\}}$. As with MERK4, the MERK5 joint stability regions $\mathcal{J}^{\{E\}}_{\alpha,10^2}$ are empty.

4.6. Numerical Results

In this section we examine the convergence rates for our newly-proposed IMEX-MRI-SR methods from Section 4.4, as well as for MERK methods applied to problems with nonlinear $f^{\{F\}}$. We also compare the efficiency of our IMEX-MRI-SR methods against both IMEX-MRI-GARK and legacy Lie-Trotter and Strang-Marchuk methods from [5].

4.6.1. KPR

The Kværnø-Prothero-Robinson problem is a coupled system of IVPs which has been widely used for testing multirate algorithms, since it is nonlinear, non-autonomous, includes

stiffness and multirate tuning parameters, and has an analytical solution. We use the same formulation and partitioning as in [5],

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \Lambda \begin{bmatrix} \frac{-3+u^2-\cos(\beta t)}{2u} \\ \frac{-2+u^2-\cos(t)}{2v} \end{bmatrix} - \begin{bmatrix} \frac{\beta \sin(\beta t)}{2u} \\ \frac{\sin(t)}{2v} \end{bmatrix}, \quad \begin{bmatrix} u(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} 2 \\ \sqrt{3} \end{bmatrix}$$
$$\Lambda = \begin{bmatrix} \lambda^{\{F\}} & \frac{1-\varepsilon}{\alpha}(\lambda^{\{F\}} - \lambda^{\{S\}}) \\ -\alpha\varepsilon(\lambda^{\{F\}} - \lambda^{\{S\}}) & \lambda^{\{S\}} \end{bmatrix} \tag{4.34}$$

for $t \in [0, 5\pi/2]$, with parameters $\lambda^{\{F\}} = -10$, $\lambda^{\{S\}} = -1$, $\varepsilon = 0.1$, $\alpha = 1$, and $\beta = 20$. This problem has solution

$$u(t) = \sqrt{3 + \cos(\beta t)}, \quad v(t) = \sqrt{2 + \cos(t)}. \tag{4.35}$$

We partition the problem as

$$f^{\{E\}} = \begin{bmatrix} 0 \\ \frac{\sin(t)}{2v} \end{bmatrix}, \quad f^{\{I\}} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \Lambda \begin{bmatrix} \frac{-3+u^2-\cos(\beta t)}{2u} \\ \frac{-2+u^2-\cos(t)}{2v} \end{bmatrix}$$
$$f^{\{F\}} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \Lambda \begin{bmatrix} \frac{-3+u^2-\cos(\beta t)}{2u} \\ \frac{-2+u^2-\cos(t)}{2v} \end{bmatrix}. \tag{4.36}$$

In Figure 4.5, we plot the convergence as H is refined for MERK methods and implicit-explicit methods, including all provided IMEX-MRI-SR methods from Section 4.4, IMEX-MRI-GARK3(a,b), IMEX-MRI-GARK4, Lie-Trotter and Strang-Marchuk. We see that all

methods converge at their expected rates. Notably, as expected from Section 4.5, the MERK methods show no convergence issues even though $f^{\{F\}}$ in (4.36) is nonlinear.

In these tests, we combined multirate methods with explicit inner Runge–Kutta methods of the same order, with the only exception of pairing a second-order inner solver with the first-order Lie-Trotter method. Each of Lie-Trotter, Strang-Marchuk, and IMEX-MRI-SR2(1) used the second-order Heun method given by the Butcher table

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}$$

. The IMEX-MRI-SR3(2) and IMEX-MRI-GARK3(a,b) methods used the third-order method by Bogacki and Shampine [2]. The IMEX-MRI-SR4(3) and IMEX-MRI-GARK4 methods used the fourth-order method by Zonneveld [64].

We measured error at 10 equally-spaced points in the time interval. The estimated convergence rates for each method, using a least squares fit of log(Max Error) versus $\log(H)$, are in the legend parentheses. For the MERK convergence tests, we used $H = \pi/2^k$, $k = 2, ..., 9$ and for the implicit-explicit method convergence tests, we used $H = \pi/2^k$, $k = 4, ..., 11$. For all tests we used fast time step size $h = H/10$. The implicit-explicit methods used a standard Newton-Raphson method with a banded linear solver for the implicit solves.

4.6.2. Stiff Brusselator

The stiff brusselator problem is an advection-reaction-diffusion system of nonlinear partial differential equations. It is a modification to the standard brusselator [19] used in [5], from which we use the same formulation and partitioning:

$$u_t = \alpha_u u_{xx} + \rho_u u_x + r_u(a - (w-1)u + u^2 v),$$

$$v_t = \alpha_v v_{xx} + \rho_v v_x + r_v(wu - u^2 v),$$

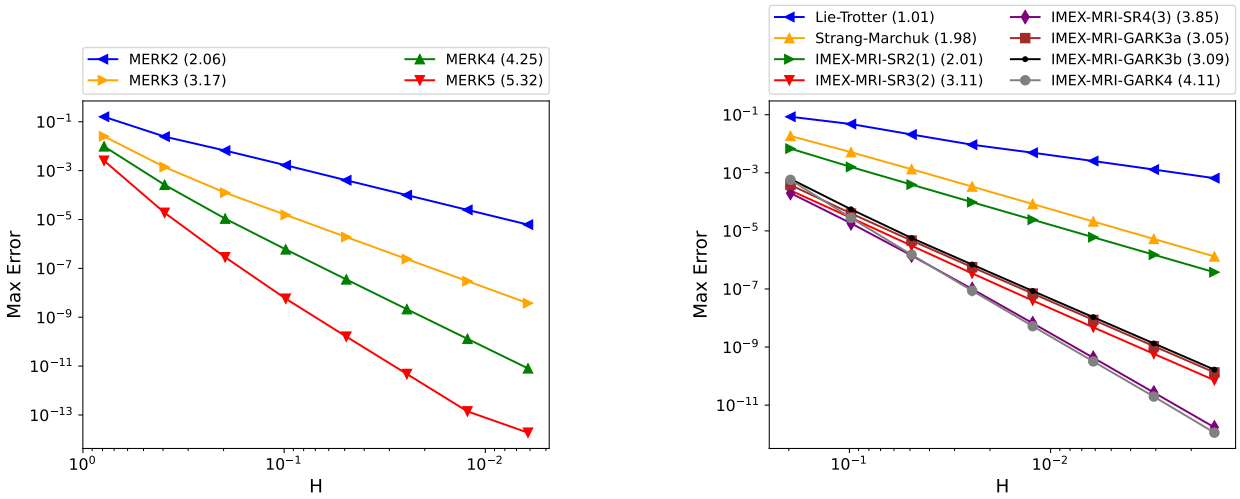


Figure 4.5: Convergence for the KPR test problem (4.34) for MERK methods (left) and implicit-explicit methods (right) using the partitioning (4.36). All methods converge at the expected theoretical rates (with measured convergence rates in parentheses), including MERK methods using the given nonlinear fast partition.

$$w_t = \alpha_w w_{xx} + \rho_w w_x + r_w(\frac{b-w}{\varepsilon} - wu),$$

for $t \in [0,3]$ and $x \in [0,1]$, with stationary boundary conditions

$$u_t(t,0) = u_t(t,1) = v_t(t,0) = v_t(t,1) = w_t(t,0) = w_t(t,1),$$

and initial conditions

$$u(0,x) = a + 0.1\sin(\pi x),$$

$$v(0,x) = b/a + 0.1\sin(\pi x),$$

$$w(0,x) = b + 0.1\sin(\pi x).$$

We partition the problem as

$$f^{\{I\}} = \begin{bmatrix} \alpha_u u_{xx} \\ \alpha_v v_{xx} \\ \alpha_w w_{xx} \end{bmatrix}, \quad f^{\{E\}} = \begin{bmatrix} \rho_u u_x \\ \rho_v v_x \\ \rho_w w_x \end{bmatrix}, \quad f^{\{F\}} = \begin{bmatrix} r_u(a - (w-1)u + u^2 v) \\ r_v(wu - u^2 v) \\ r_w(\frac{b-w}{\varepsilon} - wu) \end{bmatrix},$$

and use second order centered difference approximations for all spatial derivative operators. This problem has no analytical solution, so we used MATLAB's *ode15s* with tight tolerances $AbsTol = 10^{-14}$ and $RelTol = 2.5 \times 10^{-14}$ to generate reference solutions.

4.6.2.1. Fixed Time Step

In this section, we compare runtime efficiency of the splitting, IMEX-MRI-SR, and IMEX-MRI-GARK methods using fixed time step sizes. We use the same fixed parameters $\alpha_u = \alpha_v = \alpha_w = 10^{-2}$, $\rho_u = \rho_v = \rho_w = 10^{-3}$, $r_u = r_v = r_w = 1$, $a = 0.6$, $b = 2$, and $\varepsilon = 10^{-2}$ with initial conditions

$$u(0) = a + 0.1\sin(\pi x),\ v(0) = b/a + 0.1\sin(\pi x),\ w(0) = b + 0.1\sin(\pi x),$$

for 201 and 801 grid points as in [5]. All methods used fast time steps of $h = H/10$ and all methods used the same inner methods and implicit algebraic solvers as in Section 4.6.1.

In Figure 4.6 we plot the observed maximum solution error over ten equally spaced points in the time interval using step sizes of $H = 0.1 \cdot 2^{-k}$, $k = 0, ..., 10$. Both splitting methods and both fourth-order methods experience significant order reduction, with IMEX-MRI-SR4(3) taking the biggest hit in reducing an entire order of accuracy for the 201 grid, and two orders of accuracy for the 801 grid. The second- and third-order IMEX-MRI-SR and IMEX-MRI-GARK methods all achieve their expected order for the 201 grid. The third-order accurate methods experience only slight order reduction for the 801 grid while IMEX-MRI-SR2(1)

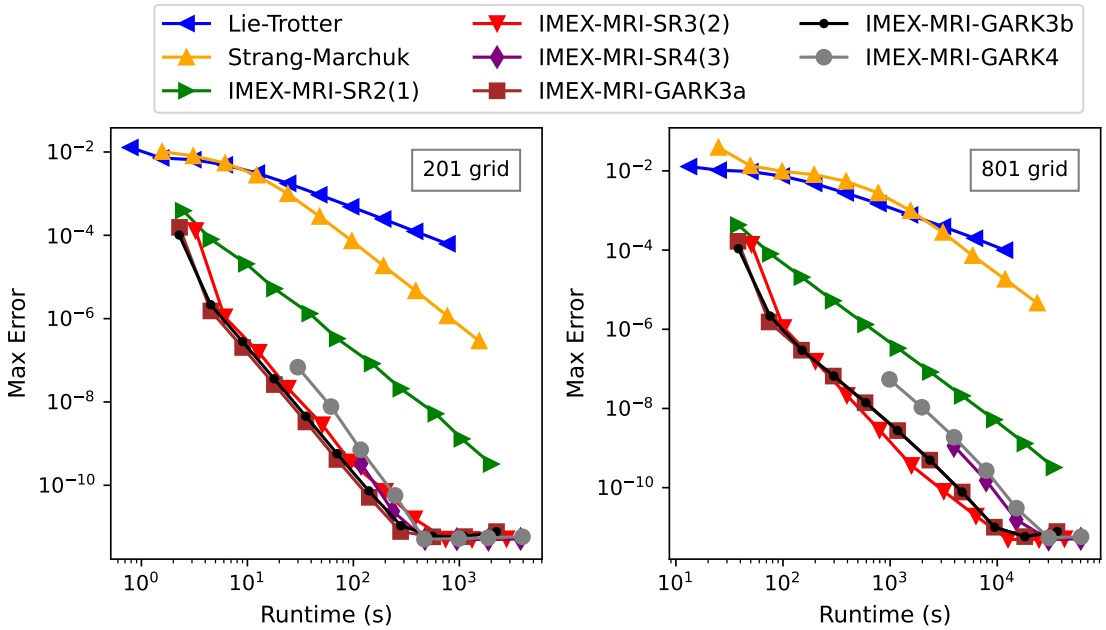


Figure 4.6: Efficiency for stiff brusselator problem using 201 grid points (left) and 801 grid points (right). Estimated least-squares convergence rates before settling at the error-floor are (0.76,1.59,2.00,3.09,3.00,3.36,3.25,3.41) and (0.72,1.24,2.01,2.90,1.90,2.71,2.69,2.42) for the 201 and 801 grids, respectively, for Lie-Trotter, Strang-Marchuk, IMEX-MRI-SR2(1), IMEX-MRI-SR3(2), IMEX-MRI-SR4(3), IMEX-MRI-GARK3a, IMEX-MRI-GARK3b, IMEX-MRI-GARK4.

remains steady at its expected order. All methods exhibit an error floor of approximately 10^{-11}, that is likely caused by the accuracy of the reference solution.

The stiffness of this problem highlights the stability limitations of the fourth order methods, IMEX-MRI-SR4(3) and IMEX-MRI-GARK4, which was observed in [5]. For the 201 grid, IMEX-MRI-SR4(3) and IMEX-MRI-GARK4 were unstable for step sizes greater than 1/320 and 1/80 respectively. For the 801 grid, IMEX-MRI-SR4(3) and IMEX-MRI-GARK4 were unstable for step sizes greater than 1/640 and 1/160, respectively.

We can see that IMEX-MRI-SR2(1) is *far* more efficient than the first- and second-order splitting methods, providing errors two to three orders of magnitude smaller for the same runtimes. It also has a steady rate of error decrease as runtime increases, while the splitting methods show periods of stagnation at larger step sizes.

The third-order and fourth-order methods tend to have similar efficiency on this problem, attaining similar error for similar runtimes. The third-order IMEX-MRI-GARK methods have a slight edge for the 201 grid and for some error ranges in the 801 grid, but IMEX-MRI-SR3(2) becomes the most efficient method for the 801 grid for errors between approximately 10^{-7} and 10^{-11}. The third-order methods are all more efficient than the fourth-order methods, where IMEX-MRI-SR4(3) maintains a slight but consistent edge over IMEX-MRI-GARK4.

4.6.2.2. Adaptive Time Step

In this section, we compare work-precision efficiency for the IMEX-MRI-SR methods in the adaptive time step context. Because the splitting and IMEX-MRI-GARK methods do not have embeddings, we omit them from these tests. We use the Constant-Constant controller from [9] with the recommended parameters $k_1 = 0.42$, $k_2 = 0.44$, and the recommended fast error measurement strategy, LASA-mean. This controller adapts H and M, such that the inner time step size $h = H/M$, in a similar manner to a standard I-controller.

We use time-varying parameters $\alpha_u = \alpha_v = \alpha_w = \rho_u = \rho_v = \rho_w = 6 \times 10^{-5} + 5 \times 10^{-5}\cos(\pi t)$, $r_u = r_v = r_w = 0.6 + 0.5\cos(4\pi t)$ adapted from [9] with the same fixed parameters $a = 1$, $b = 3.5$, $\varepsilon = 10^{-3}$ and initial conditions

$$u(0) = 1.2 + 0.1\sin(\pi x),\ v(0) = 3.1 + 0.1\sin(\pi x),\ w(0) = 3 + 0.1\sin(\pi x),$$

with 101 grid points. All IMEX-MRI-SR methods used the same inner methods and implicit algebraic solvers as in Section 4.6.1.

In Figure 4.7 we plot the number of fast function evaluations and the number of implicit solves, good indicators of overall cost at the fast and slow timescales, versus the observed maximum solution error over ten equally spaced points in the time interval when running with controller tolerance values of $\text{TOL} = 10^{-k}$, $k = 1, ..., 9$. We use $\text{TOL}^{\{S\}} = \text{TOL}^{\{F\}} = \frac{1}{2}\text{TOL}$ in our tests for simplicity. We note that IMEX-MRI-SR3(2) failed the tests with

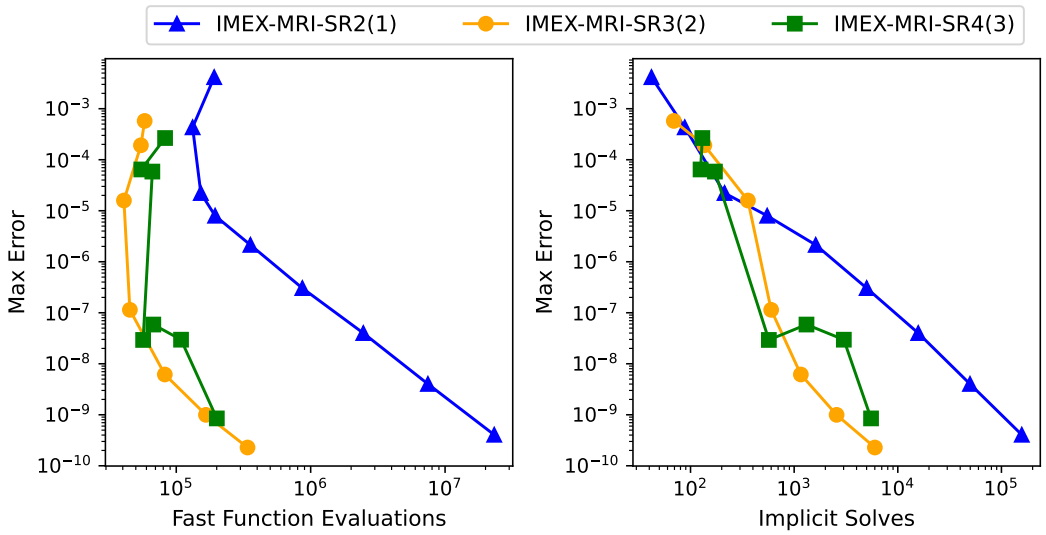


Figure 4.7: Fast function evaluations (left) and total implicit solves (right) versus the observed maximum error for the stiff brusselator problem.

TOL $= 10^{-3}, 10^{-5}$ and IMEX-MRI-SR4(3) failed the tests with TOL $= 10^{-4}, 10^{-5}$ due to getting stuck in oscillations between successful and failed steps.

We can see that IMEX-MRI-SR2(1) is much less efficient than the higher order methods in terms of fast function evaluations (farthest from the bottom-left corner), where IMEX-MRI-SR3(2) is generally the most efficient (closest to the bottom-left corner) across the range of errors. IMEX-MRI-SR4(3) is generally comparable to IMEX-MRI-SR3(2) in terms of fast function evaluations but occasionally gets "stuck," providing approximately the same error value for different total fast function evaluations, depending on the value of TOL. This is likely an indication that IMEX-MRI-SR4(3) has a low quality embedding which provides inaccurate error estimates, possibly stemming from the embedding coefficients being defined by minimizing $\|\hat{\tau}^{(4)}\|_2$ rather than the C-statistic.

The total number of implicit solves is comparable across all three methods for errors larger than approximately 10^{-5}. Below that, IMEX-MRI-SR3(2) and IMEX-MRI-SR4(3) are again comparable with IMEX-MRI-SR2(1) falling further behind. We see the same phenomenon

of IMEX-MRI-SR4(3) getting "stuck" at certain error values, achieving the same error for varying total implicit solves.

4.7. Conclusions

We introduce a new class of multirate time integration methods which builds off of previous work in IMEX-MRI-GARK and MERK methods, serving to improve various aspects of each. The proposed class of IMEX-MRI-SR methods are flexible, allowing IMEX treatment of the slow time scale while allowing the use of any viable IVP solver for the fast time scale. These methods remove the sorted abscissae requirement of IMEX-MRI-GARK methods since they start each internal stage at the beginning of the time step, thereby dramatically simplifying their order conditions and allowing introduction of embeddings. IMEX-MRI-SR methods can also be viewed as an extension to MERK methods, in that these allow implicitness at the slow time scale and nonlinearity at the fast time scale.

The convergence theory of IMEX-MRI-SR methods leverages GARK theory [46], through which we established order conditions for methods of orders one through four. Due to their structural similarity to IMEX-MRI-SR methods, we leveraged these new order conditions to analyze the previously-proposed MERK methods without their restriction to linear fast partitions. Using this analytical framework, we provided the first theoretical justification that these MERK methods (at least up through fourth order) should retain their high orders of accuracy even on problems with nonlinear fast partitions.

We analyzed joint stability as in [5], as well as in simplified implicit- or explicit-only senses of joint stability. With these reduced definitions of stability, we gain some insight into what happens when joint stability regions break down.

Using this theoretical framework, we provided three new IMEX-MRI-SR methods. These included a second-order method with a first-order embedding, and a third-order method with a second-order embedding, both derived from scratch by solving all order conditions simul-

taneously. We additionally provided a fourth-order method with a third-order embedding, that we derived from the existing LIRK4 ARK method [3].

We experimentally examined convergence for the three new methods and the existing MERK methods on the KPR test problem, finding that they all converge at their expected orders of accuracy. We also provided experimental efficiency results on the stiff brusselator problem, a nonlinear advection-reaction-diffusion system of PDEs, where we found that our methods are competitive with IMEX-MRI-GARK methods and even surpass them in some cases. We also found that IMEX-MRI-SR2(1) is vastly more efficient than the similarly second-order Strang-Marchuk operator splitting method. We found that in the context of adaptive time stepping, the third- and fourth-order IMEX-MRI-SR methods were comparable in work required for achieving a given error, while the second-order method lagged behind.

More work remains to be done on IMEX-MRI-SR methods. Higher order conditions can be derived which, while tedious due to the number of conditions at higher orders, is tractable due to the use of Kronecker product identities. Further analysis should be done on the factors that most strongly affect joint stability and what conditions, if any, can be enforced to ensure a non-empty joint stability region. Additionally, more methods should be derived, in particular an embedded fourth-order method with improved joint stability over the method provided here.

Chapter 5

New IMEX-MRI-GARK methods with embeddings

In this chapter, we propose two new IMEX-MRI-GARK methods. We propose a second-order method with a first-order embedding, and a third-order method with a second-order embedding. Not only is this the first proposed second-order IMEX-MRI-GARK method, but these are also the first IMEX-MRI-GARK methods with embeddings. We discuss the stability properties of these methods and evaluate their performance on the same test problems as in Chapter 4. We note that the background theory from Section 2.4 is useful in understanding this chapter.

5.1. Second-order method

We propose a second-order accurate method with a first-order accurate embedding which we name IMEX-MRI-GARK2(1). It has 4 stages, $n_\Gamma = n_\Omega = 2$, requires 2 nonlinear solves per step, and satisfies all order conditions analytically. Its coefficients are defined as follows.

$$c^{\{S\},T} = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \tag{5.1a}$$

$$\Gamma^{\{0\}} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -\frac{567}{290} & 0 & \frac{567}{290} & 0 \\ -\frac{119623}{80910} & 0 & -\frac{133}{279} & \frac{567}{290} \\ \hline \frac{1678}{735} & 0 & 0 & 0 \end{array}\right], \; \Gamma^{\{1\}} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{567}{290} & 0 & \frac{567}{290} & 0 \\ \frac{126583}{16182} & 0 & -\frac{395554}{40455} & \frac{567}{290} \\ \hline \frac{121}{106} & 0 & -\frac{444671}{77910} & 0 \end{array}\right] \tag{5.1b}$$

$$\Omega^{\{0\}} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{80}{311} & 0 & -\frac{80}{311} & 0 \\ \hline -\frac{17}{269} & 0 & 0 & 0 \end{array}\right], \; \Omega^{\{1\}} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{471}{311} & 0 & \frac{471}{311} & 0 \\ \hline -\frac{11}{348} & 0 & \frac{14791}{93612} & 0 \end{array}\right] \tag{5.1c}$$

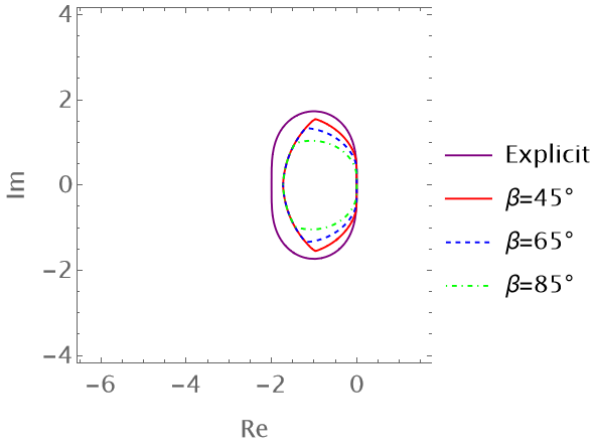
A step defined by this method consists of a fast integration over the full interval $[0, H]$, followed by two ARK-like implicit solves. The second implicit solve was needed so that we could add a sufficient number of degrees of freedom to solve the order conditions.

The last row of coefficients (below the horizontal line) define the alternate last stage, the embedding. The embedding stage is purely explicit and requires no new function evaluations.

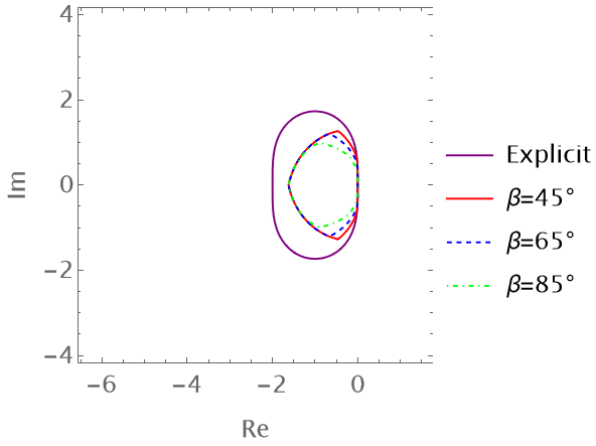
The primary set of coefficients were optimized to maximize the size of the stability region, and the embedding coefficients were optimized to minimize the norm of the residuals of the second-order conditions, $\|\hat{\tau}^{(2)}\|$, discussed in Section 4.4.1.

Figure 5.1a shows the stability regions for IMEX-MRI-GARK2(1) defined by $\mathcal{J}_{10^\circ,1,\beta,10^4}$, discussed in Section 4.3 and using the IMEX-MRI-GARK stability function defined by Chinomona [5]. Similarly, Figure 5.1b shows the stability regions defined by $\mathcal{J}_{45^\circ,1,\beta,10^4}$. We can

see that, even with quite restrictive parameters on the value of $z^{\{F\}}$ (a maximum magnitude of 1 and no more than 10° above the negative real axis), the joint stability region never reaches its theoretical maximum extent, given by the explicit component of the base ARK method, and the area of the region decays slowly as β grows.



(a) $\mathcal{J}_{10^\circ,1,\beta,10^4}$



(b) $\mathcal{J}_{45^\circ,1,\beta,10^4}$

5.2. Third-order method

We also propose a third-order accurate method with a second-order accurate embedding which we name IMEX-MRI-GARK3(2). It has 8 stages, $n_\Gamma = n_\Omega = 1$, requires 4 nonlinear solves per step, and satisfies all order conditions analytically. Its coefficients are defined as follows.

$$c^{\{S\},T} = \begin{bmatrix} 0 & \frac{3}{7} & \frac{3}{7} & \frac{8}{15} & \frac{8}{15} & 1 & 1 & 1 \end{bmatrix} \tag{5.2a}$$

$$\Gamma^{\{0\}} = \left[\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{4}{105} & 0 & \frac{1}{7} & 0 & 0 & 0 & 0 & 0 \\ \frac{388}{315} & 0 & -\frac{703}{315} & 0 & 1 & 0 & 0 & 0 \\ -\frac{33997}{92610} & 0 & \frac{6178}{9261} & 0 & \frac{1}{6} & 0 & 0 & 0 \\ \frac{43461623}{23245110} & 0 & -\frac{38315719}{11622555} & 0 & \frac{643}{1506} & 0 & 1 & 0 \\ -\frac{14243}{7530} & 0 & \frac{99701}{30120} & 0 & -\frac{1835}{3514} & 0 & -\frac{531}{280} & 1 \\ \hline \frac{2569}{4518} & 0 & -\frac{2569}{4518} & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{5.2b}$$

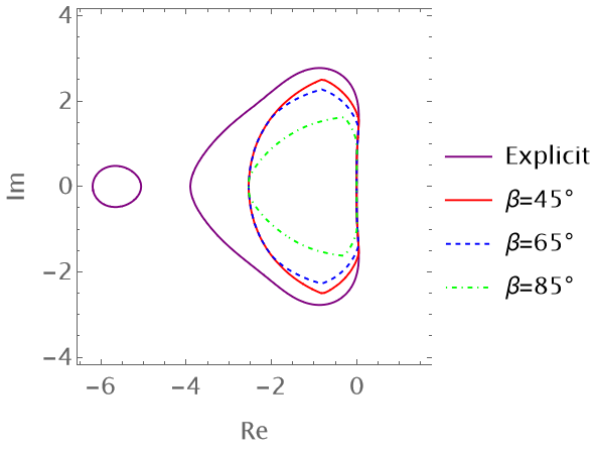
$$\Omega^{\{0\}} = \left[\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{3}{7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{73}{105} & 0 & \frac{4}{5} & 0 & 0 & 0 & 0 & 0 \\
\frac{3119}{7425} & 0 & -\frac{3119}{7425} & 0 & 0 & 0 & 0 & 0 \\
-\frac{225086}{1091475} & 0 & \frac{491891}{1091475} & 0 & \frac{2}{9} & 0 & 0 & 0 \\
-\frac{9450719}{91320075} & 0 & -\frac{32058406}{91320075} & 0 & \frac{5}{11} & 0 & 0 & 0 \\
\frac{680411548}{2132416935} & 0 & -\frac{401996371}{1550848680} & 0 & -\frac{295928}{1321551} & 0 & \frac{87599}{533960} & 0 \\
\hline
\frac{2569}{16566} & 0 & -\frac{2569}{16566} & 0 & 0 & 0 & 0 & 0
\end{array}\right] \tag{5.2c}$$

A step defined by this method consists of three pairs of fast integrations followed by ARK-like implicit solves, all followed by one additional ARK-like implicit solve. The fast integrations are over the intervals $[0, \frac{3}{7}H]$, $[\frac{3}{7}H, \frac{8}{15}H]$, and $[\frac{8}{15}H, H]$, respectively. As with IMEX-MRI-GARK2(1), the last implicit solve was needed so that sufficient degrees of freedom were present to solve the order conditions.

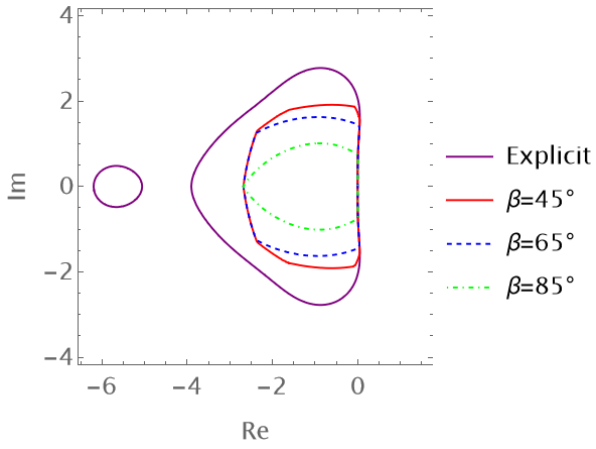
The primary set of coefficients were optimized to maximize the size of the stability region, and the embedding coefficients were optimized to minimize the norm of the residuals of the third-order conditions, $\|\hat{\tau}^{(3)}\|$.

Figure 5.2a shows the stability regions for IMEX-MRI-GARK3(2) defined by $\mathcal{J}_{10^\circ,1,\beta,10^4}$. Figure 5.2b shows the stability regions defined by $\mathcal{J}_{45^\circ,1,\beta,10^4}$. While this method's joint stability regions are larger than those of IMEX-MRI-GARK2(1), they suffer from similar problems. Specifically, the regions never reach the theoretical maximum defined by the

explicit component of the base ARK methods, and the regions decay more quickly as β grows than IMEX-MRI-GARK2(1)'s regions.



(a) $\mathcal{J}_{10^\circ,1,\beta,10^4}$



(b) $\mathcal{J}_{45^\circ,1,\beta,10^4}$

5.3. Lack of fourth-order method

Although we attempted to create an embedded fourth-order IMEX-MRI-GARK method for multiple months, we found that the structure of these methods and their corresponding order conditions were overly complex for advanced computer algebra systems to handle. If using a base method computed strictly from the IMEX-MRI-GARK coefficients, as done with IMEX-MRI-GARK2(1) and IMEX-MRI-GARK3(2), fourth-order coupling and base method conditions are prohibitively computationally expensive to solve. Mathematica [22], arguably the strongest computer algebra system available, would spend weeks computing a single `Solve` command of the order conditions. When attempting the alternate strategy of starting with an existing base method, thereby automatically satisfying the base method's fourth-order conditions and greatly simplifying the coupling conditions, one must first find a base ARK method with an embedding and with a non-decreasing abcissae vector c. As noted in [5], we were unable to find any record of any such base method in the publication literature, even without an embedding. Chinomona and Reynolds derived a new fourth-order

base ARK method with non-decreasing abcissae, but the method did not have an associated embedding.

5.4. Numerical results

In this section we compare the performance of these proposed IMEX-MRI-GARK methods against the previously published IMEX-MRI-GARK methods [5] and the IMEX-MRI-SR methods introduced in Chapter 4. For all numerical tests, IMEX-MRI-GARK2(1) uses the explicit Heun-Euler method, and IMEX-MRI-GARK3(2) uses the third-order method from Bogacki and Shampine [2]. All parameters, inner solvers, nonlinear solvers, etc., are the same as those used in Chapter 4. We omit the evaluation of splitting methods in this section to reduce clutter in the plots.

5.4.1. KPR

We evaluate the new methods' fixed-step convergence rates on the KPR problem defined in Section 4.6.1. Figure 5.3 shows these convergence rates. We can see that the two new methods achieve their expected rates of convergence. IMEX-MRI-GARK2(1) has a consistently higher error than IMEX-MRI-SR2(1) by a small margin, while IMEX-MRI-GARK3(2) has essentially identical performance to IMEX-MRI-GARK3b.

5.4.2. Stiff brusselator

We evaluate the new methods' performance efficiency as in Section 4.6.2, on the same versions of the stiff brusselator problem defined therein.

5.4.2.1. Fixed time step

Figure 5.4 shows the runtime-precision efficiency for the stiff brusselator problem defined in 4.6.2.1. We might expect IMEX-MRI-GARK3a and b to be more efficient than IMEX-MRI-SR3(2) due to evolving the fast dynamics over a shorter duration and requiring

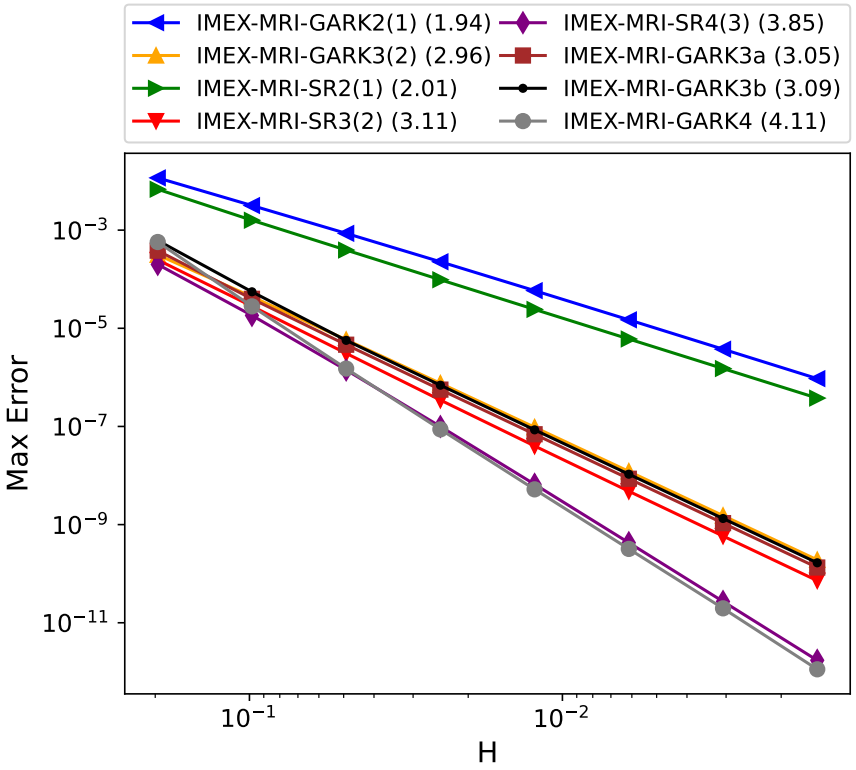


Figure 5.3: Convergence for the KPR test problem (4.34) for implicit-explicit multirate infinitesimal methods using the partitioning (4.36). All methods converge at the expected theoretical rates, with measured convergence rates in parentheses.

3 nonlinear solves per step versus 4 for IMEX-MRI-SR3(2), but this is not observed in the experimental results. IMEX-MRI-GARK3a and b and IMEX-MRI-SR3(2) have approximately the same efficiency, hinting that IMEX-MRI-GARK methods have large coefficients in their Taylor expansions. These coefficients don't affect the convergence rate but do provide a large multiplicative constant on the error observed. This effect is also observed in the proposed IMEX-MRI-GARK2(1) and IMEX-MRI-GARK3(2). Not only do the proposed methods have large constant error factors, but they require the same amount of nonlinear solves as the same-order IMEX-MRI-SR counterparts, the driving factor in runtime. When IMEX-MRI-GARK3a and b require *less* work than IMEX-MRI-SR3(2) to achieve the same efficiency, it follows that IMEX-MRI-GARK methods requiring roughly equivalent work to same-order IMEX-MRI-SR methods would have worse efficiency.

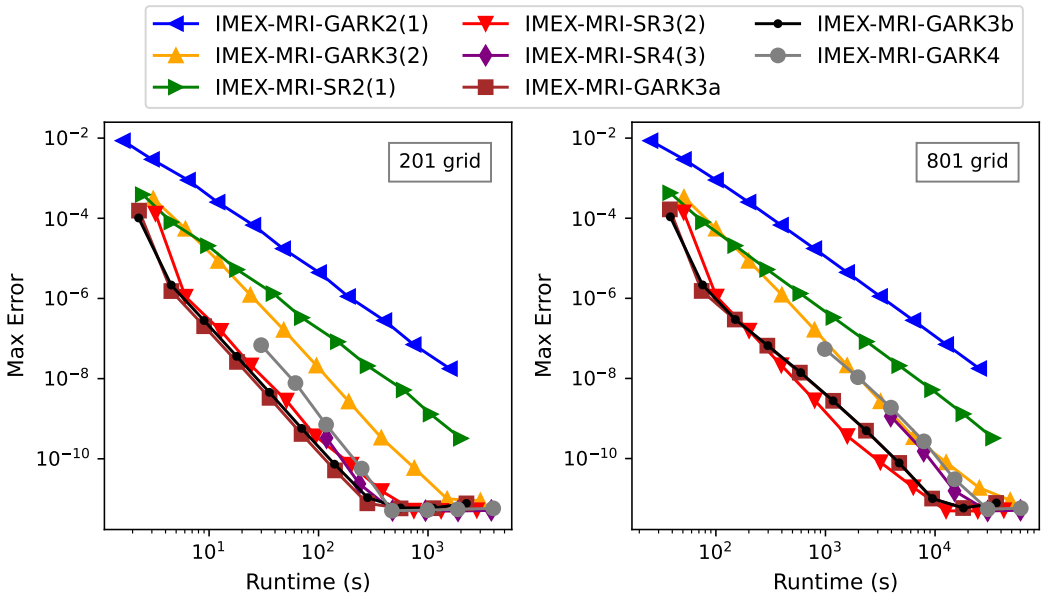


Figure 5.4: Efficiency for stiff brusselator problem using 201 grid points (left) and 801 grid points (right).

5.4.2.2. Adaptive time step

Figure 5.5 shows the work-precision efficiency of the stiff brusselator problem defined in Section 4.6.2.2. We can see that, while IMEX-MRI-GARK3(2) is far more efficient in terms of total fast function evaluations, it is outclassed by IMEX-MRI-SR3(2) and IMEX-MRI-SR4(3) in terms of total implicit solves, which are often the main driver of computational cost. The IMEX-MRI-GARK methods' improvements upon their same-order IMEX-MRI-SR counterparts in terms of fast function evaluations is expected because, as noted in Section 2.4.1, their total fast IVP integration duration is H which is less than that of IMEX-MRI-SR methods.

We note that IMEX-MRI-GARK3(2) gives error values approximately 100 times smaller than desired for the lax tolerances of $10^{-k}, k = 3, 4, 5$. Additionally, we note that IMEX-MRI-GARK2(1) failed its solves for tolerances of $10^{-k}, k = 1, 2, 9$, and IMEX-MRI-GARK3(2) failed its solves for tolerances of $10^{-k}, k = 1, 2$.

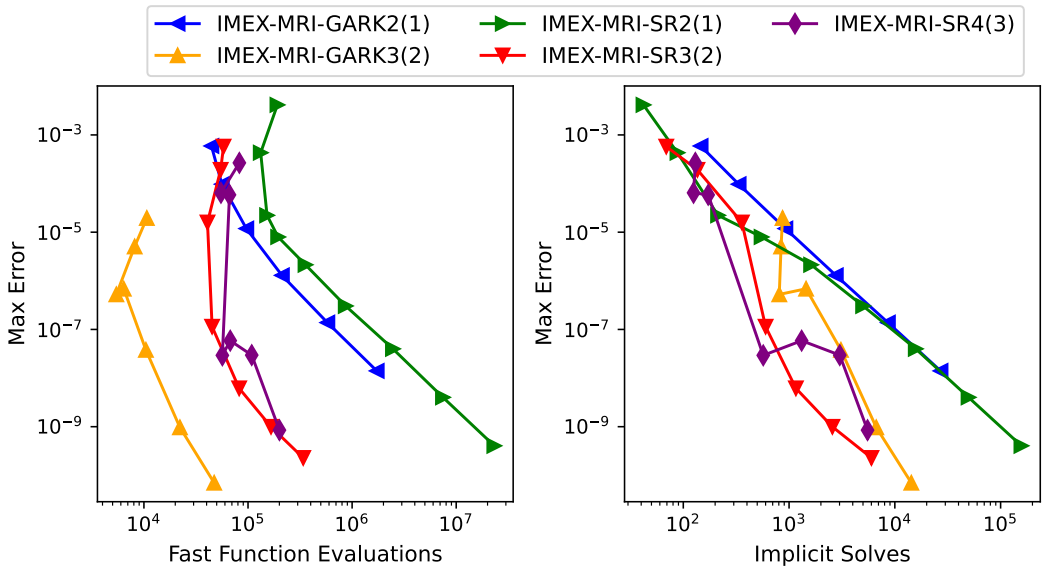


Figure 5.5: Fast function evaluations (left) and total implicit solves (right) versus the observed maximum error for the stiff brusselator problem.

5.5. Conclusions

We proposed two new IMEX-MRI-GARK methods, a second-order accurate method with a first-order embedding, IMEX-MRI-GARK2(1), and a third-order method with a second-order embedding, IMEX-MRI-GARK3(2). IMEX-MRI-GARK2(1) is the first proposed second-order IMEX-MRI-GARK method, and these are the first proposed IMEX-MRI-GARK methods with embeddings. We discussed their construction and their stability properties. We also discussed the challenges in deriving a fourth-order IMEX-MRI-GARK method with an embedding.

We then evaluated the new methods' performance against existing IMEX-MRI methods in a variety of metrics, including convergence, runtime efficiency, and work-precision efficiency. The new methods performed in line with convergence rate expectations without surprises. However, the new methods demonstrated worse efficiency than the other same-order methods in a fixed-step setting, due to large constant error factors and work required per step. Lastly, we found that the new methods had mixed performance in the adaptive step setting. IMEX-MRI-GARK3(2) had clearly the best performance in terms of total fast function evaluations

versus the achieved error but was outclassed by IMEX-MRI-SR3(2) and IMEX-MRI-SR4(3) in terms of total implicit solves versus the achieved error. IMEX-MRI-GARK2(1) performed similarly to IMEX-MRI-SR2(1) in both adaptive step metrics. Based on these results, we generally recommend using IMEX-MRI-SR methods over IMEX-MRI-GARK methods.

Chapter 6

Conclusion

In this chapter we summarize the contributions made in this thesis and discuss potential fruitful areas of future research.

6.1. Overall contributions

In this thesis we explored and developed the idea of adaptivity in multirate infinitesimal methods. We proposed the first time step controllers designed specifically for multirate infinitesimal methods, which are derived from ideas in control theory. We additionally introduced a new class of implicit-explicit multirate infinitesimal IVP-solving methods which improve upon several existing classes of multirate infinitesimal methods.

In Chapter 3 we proposed the first time step controllers for multirate infinitesimal methods. We used a control theoretic approach, inspired by Gustafsson [16] in their derivations. We discussed several techniques of measuring error from the fast dynamics and weighed their costs versus their expected accuracy. We tested these fast error measurement strategies and controllers across a range of multirate IVPs, using a range of both implicit and explicit MRI-GARK [45] methods. We found that the cheapest fast error measurement strategy was the most efficient and that all of our multirate controllers outperform classical controllers with a fixed multirate ratio M on average across the test suite.

In Chapter 4 we proposed a new class of multirate infinitesimal methods. These methods, implicit-explicit multirate infinitesimal stage-restart (IMEX-MRI-SR) methods, improve upon IMEX-MRI-GARK [5] methods by including an implicit solve with every stage, thus removing the uncertainty present in designing IMEX-MRI-GARK methods, deriving simpler order conditions, and allowing for non-decreasing abcissae vector $c^{\{S\}}$ (thereby allowing for a

wider variety of base methods to be used). These methods also extend MERK [32] methods to arbitrary nonlinearity of the fast dynamics and to include implicit-correction terms.

In Chapter 5 we proposed the first IMEX-MRI-GARK methods with embeddings. We discussed their construction, explored their stability problems, and discussed the challenges in deriving a fourth-order methods. We evaluated their performance in a variety of numerical tests and found that they behave in line with expectations in terms of convergence rates but are out-shone by IMEX-MRI-SR methods in terms of both fixed-step and adaptive performance. We recommend using IMEX-MRI-SR methods in both cases.

6.2. Future work

The area of multirate infinitesimal adaptivity is still fruitful. There are a variety of extensions to this work which can be immediately useful.

While the multirate controllers developed work well, they cannot be used with "nested adaptivity." That is, the method that solves the fast IVPs must use the fixed step size determined by H and M from the controller. A new family of controllers can be developed which allow for adaptivity of the fast IVP solvers–perhaps a simultaneous time step and tolerance controller, where the controller dictates the desired tolerance with which stage should be solved to produce an overall accurate step.

A wider set of IMEX-MRI-SR methods can be produced which target certain problem types or optimize their free coefficients in different ways.

New test problems which exhibit more strongly multirate behavior without being overly computationally expensive would benefit the quality of testing of new controllers and multirate methods.

Finally, a more thorough understanding of joint stability in multirate methods should be explored, including any possible algebraic conditions on the coefficients of the methods which can guarantee forms of stability.

Appendix A

A.1. Optimal Performance Estimation Algorithms

In order to compare the performance of our proposed adaptive controllers and error estimation algorithms, we create a baseline set of "optimal" cost values. We note that in an practice the most computationally efficient values of H_n and M_n will depend on a number of factors, including: the IVP itself, the multirate method under consideration, the cost of any implicit solvers at either time scale, the desired solution accuracy, and even the relative computational cost of evaluating the slow and fast right-hand side functions, $f^{\{S\}}$ and $f^{\{F\}}$. Furthermore, even these optimal values of H_n and M_n will vary as functions of time throughout the simulation, particularly for nontrivial multirate problems.

In this work, we define the optimal cost as the minimal number of $f^{\{S\}}$ and $f^{\{F\}}$ evaluations required to reach the end of the time interval, where each step results in local error estimates that achieve the chosen tolerance, and with each step locally optimal with respect to a prescribed computational efficiency measurement. For the sake of simplicity, we define this efficiency measurement as

$$\text{efficiency} = \frac{H_n}{\text{cost}}, \tag{A.1}$$

$$\text{cost} = \text{slowWeight} \cdot f_{\text{evals}}^{\{S\}} + f_{\text{evals}}^{\{F\}}, \tag{A.2}$$

where $f_{\text{evals}}^{\{S\}}$ and $f_{\text{evals}}^{\{F\}}$ are the total number of $f^{\{S\}}$ and $f^{\{F\}}$ evaluations for the multirate time step, respectively. Here, "slowWeight" provides a problem-specific factor that encodes the relative costs of $f^{\{S\}}$ and $f^{\{F\}}$. We note that for any given simulation, this value could itself depend on numerous un-modeled factors, such as the IVP under consideration, its numerical implementation, and even the computational hardware. However, irrespective of

the "slowWeight" value used, for a given slow step size H_n, a method that results in a smaller overall "cost" corresponds with increased efficiency.

We chose the definitions (A.1)-(A.2) because, in the goal of achieving the cheapest possible solve of a given IVP to a given tolerance, we want as large of step sizes as possible, and as small of costs as possible. If a step is rather expensive, e.g. if the value of M_n is high, the step can still achieve a high efficiency if the step size was large. Eventually, once the errors arising from the fast time scale are sufficiently small for a given method or problem, additional increases to M_n will not improve the overall accuracy and will thereby lead to decreased efficiency. Similarly, H_n will be bounded from above due to accuracy considerations, and although decreasing H_n below this bound may allow for a smaller M_n, the overall efficiency could decrease.

With these definitions in place, our approach to find the optimal set of H-M pairs is shown in pseudocode representation in Algorithms 1 and 2. This is essentially a brute-force mechanism to rigorously determine the best-case values for multirate adaptivity algorithms. The function "ComputeReferenceSolution" is a black box that computes the reference solution at a desired time $t_i + H$ and is assumed to be more accurate than the "ComputeStep" function. The function "ComputeStep" is a black box function that takes one step with the given method from t_i to $t_i + H$ and returns the total slow and fast function calls, the error in the step's solution, and the solution itself. For a given IVP, initial condition, and initial time, the algorithm iterates over increasing values of the integer multirate ratio M_n and uses the given multirate method to find the maximal step size H_n for each M_n which gives an error close to the chosen tolerance via a binary search process, stopping when the interval width is smaller than a relative tolerance H_{tol} of the midpoint of the interval. Once the efficiency from increasing M_n decreases below some relative tolerance eff_{rtol} of the maximum so far found, the solution is moved forward based on the most efficient (H_n, M_n) pair and repeats, iterating until the algorithm reaches the end of the given time window $[t_0, t_f]$.

Algorithms 1 and 2 are rather costly and the results from one run are specific to the IVP, method, and other parameters. For consistency, we always run the algorithm with the parameters slowWeight = 10, $H_{fine} = 10^{-10}$, $H_{tol} = 10^{-5}$, $H_{interval} = 10^{-1}$, $M_{max\_iter} = 400$, $M_{min\_iter} = 10$, $\text{eff}_{rtol} = 10^{-1}$, *and a sixth order explicit RK method [60] with small time steps for reference solutions.* We further note that the resulting "optimal" total $f^{\{S\}}$ and $f^{\{F\}}$ evaluations across each most efficient step found by this algorithm achieve a cost that is nearly impossible for a time adaptivity controller to reach in practice, and should thus be considered a best possible scenario.

Result: Optimal H array H_{opt}, Optimal M array M_{opt}, Total $f^{\{S\}}$ evaluations $f_{opt}^{\{S\}}$, Total $f^{\{F\}}$ evaluations $f_{opt}^{\{F\}}$.

Given an IVP, multirate method, error tolerance tol, weight factor slowWeight, initial condition y_0, time interval $\{t_0, t_f\}$, minimum step H_{fine}, binary search H tolerance H_{tol}, binary search H interval width $H_{interval}$, M maximum M_{max_iter}, M minimum M_{min_iter}, and relative efficiency tolerance eff_{rtol}:;

$f_{opt}^{\{S\}} \leftarrow 0$, $f_{opt}^{\{F\}} \leftarrow 0$, $i \leftarrow 0$, $t \leftarrow t_0$, $y_i \leftarrow y_0$;
while $t + H_{fine} < t_f$ **do**
    empty H_{array}, M_{array}, eff_{array}, $f_{evals,array}^{\{S\}}$, $f_{evals,array}^{\{F\}}$, y_{array};
    $M \leftarrow 1$;
    while $M < M_{max\_iter}$ **do**
        H, eff, $f_{evals}^{\{S\}}$, $f_{evals}^{\{F\}}$, $y_{temp} \leftarrow$ FindH(IVP, method, tol, slowWeight, y, t, H_{fine}, M_{new}, H_{tol}, $H_{interval}$);
        if $\frac{\text{eff}_{array}.\max() - \text{eff}}{\text{eff}_{array}.\max()} > \text{eff}_{rtol}$ and $M > M_{min\_iter}$ **then**
            break;
        else
            M_{array}.append(M);
            H_{array}.append(H);
            eff_{array}.append(eff);
            $f_{evals,array}^{\{S\}}$.append($f_{evals}^{\{S\}}$);
            $f_{evals,array}^{\{F\}}$.append($f_{evals}^{\{F\}}$);
            y_{array}.append(y_{temp})
        end
    end
    $opt\_idx \leftarrow \text{eff}_{array}$.indexOf($\text{eff}_{array}$.max());
    H_{opt}.append($H_{array}[opt\_idx]$);
    M_{opt}.append($M_{array}[opt\_idx]$);
    $f_{opt}^{\{S\}} \leftarrow f_{opt}^{\{S\}} + f_{evals,array}^{\{S\}}[opt\_idx]$
    $f_{opt}^{\{F\}} \leftarrow f_{opt}^{\{F\}} + f_{evals,array}^{\{F\}}[opt\_idx]$
    $t \leftarrow t + H_{array}[opt\_idx]$;
    $y \leftarrow y_{array}[opt\_idx]$;
end

Algorithm 1: Optimal H-M Search Algorithm

Result: Maximal step size H giving error close to tol, efficiency of computation eff using step size H, number of $f^{\{S\}}$ evaluations $f^{\{S\}}_{evals}$, number of $f^{\{F\}}$ evaluations $f^{\{F\}}_{evals}$, computed solution y_{i+1} using step size H.

Given an IVP, multirate method, error tolerance tol, weight factor slowWeight, initial condition y_i, initial time t_i, minimum step H_{fine}, multirate factor M, binary search H tolerance H_{tol}, and binary search H interval width $H_{interval}$:;

$y_{ref} \leftarrow$ ComputeReferenceSolution(IVP, y_i, t_i, H_{fine});

err, $f^{\{S\}}_{evals}$, $f^{\{F\}}_{evals}$, $y_{i+1} \leftarrow$ ComputeStep(IVP, method, y_i, t_i, H_{mid}, M, y_{ref});

$cost \leftarrow$ slowWeight $\cdot f^{\{S\}}_{evals} + f^{\{F\}}_{evals}$;

eff $\leftarrow H/$cost;

if $err < tol$ **then**

  $H_{right} \leftarrow 0$;

  while $err < tol$ and $t_i + H_{right} < t_f$ **do**

    $H_{left} \leftarrow H_{right}$;

    $H_{right} \leftarrow \min(H_{right} + H_{interval},\ t_f - t_i)$;

    $H_{mid} \leftarrow \frac{1}{2}(H_{left} + H_{right})$;

    $y_{ref} \leftarrow$ ComputeReferenceSolution(IVP, y_i, t_i, H_{right});

    err, $f^{\{S\}}_{evals}$, $f^{\{F\}}_{evals}$, $y_{i+1} \leftarrow$ ComputeStep(IVP, method, y_i, t_i, H_{right}, M, y_{ref});

    $cost \leftarrow$ slowWeight $\cdot f^{\{S\}}_{evals} + f^{\{F\}}_{evals}$;

    eff $\leftarrow H/$cost;

    $n \leftarrow n + 1$;

  end

  if $err > tol$ **then**

    while $(H_{right} - H_{left})/H_{mid} > H_{tol}$ **do**

      $H_{mid} \leftarrow \frac{1}{2}(H_{left} + H_{right})$;

      $y_{ref} \leftarrow$ComputeReferenceSolution(IVP, y_i, t_i, H_{right});

      $err, f^{\{S\}}_{evals}, f^{\{F\}}_{evals}, y_{i+1} \leftarrow$ ComputeStep(IVP,method,$y_i, t_i, H_{mid}, M, y_{ref}$);

      $cost \leftarrow$ slowWeight $\cdot f^{\{S\}}_{evals} + f^{\{F\}}_{evals}$;

      eff $\leftarrow H/$cost;

      if $err \leq tol$ **then**

        $H_{left} \leftarrow H_{mid}$;

      else

        $H_{right} \leftarrow H_{mid}$;

      end

    end

    $H \leftarrow H_{left}$

  else

    $H \leftarrow H_{right}$

  end

else

  Failure (H_{fine} was insufficiently small).;

end

Algorithm 2: FindH Algorithm

Appendix B

B.1. IMEX-MRI-SR2(1) Coefficients

$$c^{\{S\}} = \begin{bmatrix} \\ 0 \\ \\ \frac{3}{5} \\ \\ \frac{4}{15} \\ \\ 1 \\ \\ \end{bmatrix}, \; \Omega^{\{0\}} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ \frac{3}{5} & 0 & 0 & 0 \\ \frac{14}{165} & \frac{2}{11} & 0 & 0 \\ -\frac{13}{54} & \frac{137}{270} & \frac{11}{15} & 0 \\ \hline -\frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 0 \end{array}\right], \; \Gamma = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ -\frac{11}{23} & \frac{11}{23} & 0 & 0 \\ -\frac{6692}{52371} & -\frac{18355}{52371} & \frac{11}{23} & 0 \\ \frac{11621}{90666} & -\frac{215249}{226665} & \frac{17287}{50370} & \frac{11}{23} \\ \hline -\frac{31}{12} & -\frac{1}{6} & \frac{11}{4} & 0 \end{array}\right]. \tag{B.1}$$

B.2. IMEX-MRI-SR3(2) Coefficients

$$c^{\{S\},T} = \begin{bmatrix} 0 & \frac{23}{34} & \frac{4}{5} & \frac{17}{15} & 1 \end{bmatrix}, \tag{B.2a}$$

$$\Omega^{\{0\}} = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ \frac{23}{34} & 0 & 0 & 0 & 0 \\ \frac{71}{70} & -\frac{3}{14} & 0 & 0 & 0 \\ \frac{124}{1155} & \frac{4}{7} & \frac{5}{11} & 0 & 0 \\ \frac{162181}{187680} & \frac{119}{1380} & \frac{11}{32} & -\frac{5}{17} & 0 \\ \hline \frac{76355}{74834} & -\frac{46}{31} & \frac{67}{34} & -\frac{36}{71} & 0 \end{array}\right], \tag{B.2b}$$

$$\Omega^{\{1\}} = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{14453}{63825} & \frac{14453}{63825} & 0 & 0 & 0 \\ -\frac{2101267877}{1206582300} & -\frac{2476735438}{301645575} & -\frac{13575085}{2098404} & 0 & 0 \\ -\frac{762580446799}{588660102960} & \frac{11083240219}{4328383110} & -\frac{211274129}{100368304} & \frac{89562055}{106641323} & 0 \\ \hline -\frac{3732974}{2278035} & \frac{13857574}{2278035} & -\frac{52}{9} & \frac{4}{3} & 0 \end{array}\right] \tag{B.2c}$$

$$\Gamma = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ -\frac{4}{7} & \frac{4}{7} & 0 & 0 & 0 \\ -\frac{2707004}{3127425} & \frac{919904}{3127425} & \frac{4}{7} & 0 & 0 \\ \frac{852879271}{703839675} & -\frac{1575000496}{703839675} & \frac{5}{11} & \frac{4}{7} & 0 \\ \frac{43136869}{2019912118} & -\frac{73810600}{1009956059} & -\frac{17653551}{87822266} & -\frac{13993902}{43911133} & \frac{4}{7} \\ \hline -\frac{179}{4140} & \frac{799}{14490} & \frac{1}{14} & -\frac{1}{12} & 0 \end{array}\right] \tag{B.2d}$$

B.3. IMEX-MRI-SR4(3) Coefficients

$$c^{\{S\},T} = \left[\begin{array}{ccccccc} 0 & \frac{1}{4} & \frac{3}{4} & \frac{11}{20} & \frac{1}{2} & 1 & 1 \end{array}\right], \tag{B.3a}$$

$$\Omega^{\{0\}} = \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{9}{8} & -\frac{3}{8} & 0 & 0 & 0 & 0 & 0 \\ \frac{187}{2340} & \frac{7}{9} & -\frac{4}{13} & 0 & 0 & 0 & 0 \\ \frac{64}{165} & \frac{1}{6} & -\frac{3}{5} & \frac{6}{11} & 0 & 0 & 0 \\ \frac{1816283}{549120} & -\frac{2}{9} & -\frac{4}{11} & -\frac{1}{6} & -\frac{2561809}{1647360} & 0 & 0 \\ 0 & \frac{7}{11} & -\frac{2203}{264} & \frac{10825}{792} & -\frac{85}{12} & \frac{841}{396} & 0 \\ \hline \frac{1}{400} & \frac{49}{12} & \frac{43}{6} & -\frac{7}{10} & -\frac{85}{12} & -\frac{2963}{1200} & 0 \end{array}\right] \tag{B.3b}$$

$$\Omega^{\{1\}} = \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{11}{4} & \frac{11}{4} & 0 & 0 & 0 & 0 & 0 \\ -\frac{1228}{2925} & -\frac{92}{225} & \frac{808}{975} & 0 & 0 & 0 & 0 \\ -\frac{2572}{2805} & \frac{167}{255} & \frac{199}{136} & -\frac{1797}{1496} & 0 & 0 & 0 \\ -\frac{1816283}{274560} & \frac{253}{36} & -\frac{23}{44} & \frac{76}{3} & -\frac{20775791}{823680} & 0 & 0 \\ 0 & \frac{107}{132} & \frac{1289}{88} & -\frac{9275}{792} & 0 & -\frac{371}{99} & 0 \\ \hline -\frac{1}{200} & -\frac{137}{24} & -\frac{235}{16} & \frac{1237}{80} & 0 & \frac{2963}{600} & 0 \end{array}\right] \tag{B.3c}$$

$$\Gamma = \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{13}{100} & -\frac{7}{30} & -\frac{11}{75} & \frac{1}{4} & 0 & 0 & 0 \\ \frac{6}{85} & -\frac{301}{1360} & -\frac{99}{544} & \frac{45}{544} & \frac{1}{4} & 0 & 0 \\ 0 & -\frac{9}{4} & -\frac{19}{48} & -\frac{75}{16} & \frac{85}{12} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{B.3d}$$

B.4. MERK4 IMEX-MRI-SR Coefficients

We list the non-zero coefficients of the MERK5 method's IMEX-MRI-SR formulation below.

$$\Omega_{i,1}^{\{0\}} = c_i^{\{S\}},\ i = 1, ..., s^{\{S\}} \tag{B.4}$$

$$\Omega_{3,1}^{\{1\}} = -\Omega_{3,2}^{\{1\}},\ \Omega_{3,2}^{\{1\}} = \frac{c_3^{\{S\}\times 2}}{c_2^{\{S\}}},\ \Omega_{4,1}^{\{1\}} = -\Omega_{4,2}^{\{1\}},\ \Omega_{4,2}^{\{1\}} = \frac{c_4^{\{S\}\times 2}}{c_2^{\{S\}}},$$

$$\begin{aligned}
\Omega_{5,1}^{\{1\}} &= -(\Omega_{5,3}^{\{1\}} + \Omega_{5,4}^{\{1\}}),\ \Omega_{5,3}^{\{1\}} = -\frac{c_4^{\{S\}} c_5^{\{S\}\times 2}}{c_3^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})},\ \Omega_{5,4}^{\{1\}} = \frac{c_3^{\{S\}} c_5^{\{S\}\times 2}}{c_4^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})} \\
\Omega_{6,1}^{\{1\}} &= -(\Omega_{6,3}^{\{1\}} + \Omega_{6,4}^{\{1\}}),\ \Omega_{6,3}^{\{1\}} = -\frac{c_4^{\{S\}} c_6^{\{S\}\times 2}}{c_3^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})},\ \Omega_{6,4}^{\{1\}} = \frac{c_3^{\{S\}} c_6^{\{S\}\times 2}}{c_4^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})} \\
\Omega_{7,1}^{\{1\}} &= -(\Omega_{7,5}^{\{1\}} + \Omega_{7,6}^{\{1\}}),\ \Omega_{7,5}^{\{1\}} = -\frac{c_6^{\{S\}}}{c_5^{\{S\}}(c_5^{\{S\}} - c_6^{\{S\}})},\ \Omega_{7,6}^{\{1\}} = \frac{c_5^{\{S\}}}{c_6^{\{S\}}(c_5^{\{S\}} - c_6^{\{S\}})}
\end{aligned} \tag{B.5}$$

$$\begin{aligned}
\Omega_{5,1}^{\{1\}} &= -(\Omega_{5,3}^{\{1\}} + \Omega_{5,4}^{\{1\}}),\ \Omega_{5,3}^{\{1\}} = \frac{1}{c_3^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})},\ \Omega_{5,4}^{\{1\}} = -\frac{1}{c_4^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})} \\
\Omega_{6,1}^{\{1\}} &= -(\Omega_{6,3}^{\{1\}} + \Omega_{6,4}^{\{1\}}),\ \Omega_{6,3}^{\{1\}} = \frac{1}{c_3^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})},\ \Omega_{6,4}^{\{1\}} = -\frac{1}{c_4^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})} \\
\Omega_{7,1}^{\{1\}} &= -(\Omega_{7,5}^{\{1\}} + \Omega_{7,6}^{\{1\}}),\ \Omega_{7,5}^{\{1\}} = \frac{1}{c_5^{\{S\}}(c_5^{\{S\}} - c_6^{\{S\}})},\ \Omega_{7,6}^{\{1\}} = -\frac{1}{c_6^{\{S\}}(c_5^{\{S\}} - c_6^{\{S\}})}
\end{aligned} \tag{B.6}$$

This method attains general fourth-order accuracy when

$$c_6^{\{S\}} = \frac{3 - 4c_5^{\{S\}}}{4 - 6c_5^{\{S\}}}. \tag{B.7}$$

B.5. MERK5 IMEX-MRI-SR Coefficients

We list the non-zero coefficients of the MERK5 method's IMEX-MRI-SR formulation below.

$$\Omega_{i,1}^{\{0\}} = c_i^{\{S\}},\ i = 1, ..., s^{\{S\}} \tag{B.8}$$

$$\begin{aligned}
&\Omega_{3,1}^{\{1\}} = -\Omega_{3,2}^{\{1\}},\ \Omega_{3,2}^{\{1\}} = c_3^{\{S\}\times 2}\alpha_2,\\
&\Omega_{4,1}^{\{1\}} = -\Omega_{4,2}^{\{1\}},\ \Omega_{4,2}^{\{1\}} = c_4^{\{S\}\times 2}\alpha_2,\\
&\Omega_{5,1}^{\{1\}} = -\left(\Omega_{5,3}^{\{1\}} + \Omega_{5,4}^{\{1\}}\right),\ \Omega_{5,3}^{\{1\}} = c_5^{\{S\}\times 2}\alpha_3,\ \Omega_{5,4}^{\{1\}} = c_5^{\{S\}\times 2}\alpha_4\\
&\Omega_{6,1}^{\{1\}} = -\left(\Omega_{6,3}^{\{1\}} + \Omega_{6,4}^{\{1\}}\right),\ \Omega_{6,3}^{\{1\}} = c_6^{\{S\}\times 2}\alpha_3,\ \Omega_{6,4}^{\{1\}} = c_6^{\{S\}\times 2}\alpha_4\\
&\Omega_{7,1}^{\{1\}} = -\left(\Omega_{7,3}^{\{1\}} + \Omega_{7,4}^{\{1\}}\right),\ \Omega_{7,3}^{\{1\}} = c_7^{\{S\}\times 2}\alpha_3,\ \Omega_{7,4}^{\{1\}} = c_7^{\{S\}\times 2}\alpha_4\\
&\Omega_{8,1}^{\{1\}} = -\left(\Omega_{8,5}^{\{1\}} + \Omega_{8,6}^{\{1\}} + \Omega_{8,7}^{\{1\}}\right),\ \Omega_{8,5}^{\{1\}} = c_8^{\{S\}\times 2}\alpha_5,\ \Omega_{8,6}^{\{1\}} = c_8^{\{S\}\times 2}\alpha_6,\ \Omega_{8,7}^{\{1\}} = c_8^{\{S\}\times 2}\alpha_7\\
&\Omega_{9,1}^{\{1\}} = -\left(\Omega_{9,5}^{\{1\}} + \Omega_{9,6}^{\{1\}} + \Omega_{9,7}^{\{1\}}\right),\ \Omega_{9,5}^{\{1\}} = c_9^{\{S\}\times 2}\alpha_5,\ \Omega_{9,6}^{\{1\}} = c_9^{\{S\}\times 2}\alpha_6,\ \Omega_{9,7}^{\{1\}} = c_9^{\{S\}\times 2}\alpha_7\\
&\Omega_{10,1}^{\{1\}} = -\left(\Omega_{10,5}^{\{1\}} + \Omega_{10,6}^{\{1\}} + \Omega_{10,7}^{\{1\}}\right),\ \Omega_{10,5}^{\{1\}} = c_{10}^{\{S\}\times 2}\alpha_5,\ \Omega_{10,6}^{\{1\}} = c_{10}^{\{S\}\times 2}\alpha_6,\ \Omega_{10,7}^{\{1\}} = c_{10}^{\{S\}\times 2}\alpha_7\\
&\Omega_{11,1}^{\{1\}} = -\left(\Omega_{11,8}^{\{1\}} + \Omega_{11,9}^{\{1\}} + \Omega_{11,10}^{\{1\}}\right),\ \Omega_{11,8}^{\{1\}} = \alpha_8,\ \Omega_{11,9}^{\{1\}} = \alpha_9,\ \Omega_{11,10}^{\{1\}} = \alpha_{10}
\end{aligned} \tag{B.9}$$

where

$$\alpha_2 = \frac{1}{c_2^{\{S\}}},\ \alpha_3 = \frac{c_3^{\{S\}}}{c_3^{\{S\}}(c_4^{\{S\}} - c_3^{\{S\}})},\ \alpha_4 = \frac{c_4^{\{S\}}}{c_4^{\{S\}}(c_3^{\{S\}} - c_4^{\{S\}})}$$

$$\alpha_5 = \frac{c_6^{\{S\}} c_7^{\{S\}}}{c_5^{\{S\}}(c_5^{\{S\}} - c_6^{\{S\}})(c_5^{\{S\}} - c_7^{\{S\}})},\ \alpha_6 = \frac{c_5^{\{S\}} c_7^{\{S\}}}{c_6^{\{S\}}(c_6^{\{S\}} - c_5^{\{S\}})(c_6^{\{S\}} - c_7^{\{S\}})},$$

$$\alpha_7 = \frac{c_5^{\{S\}} c_6^{\{S\}}}{c_7^{\{S\}}(c_7^{\{S\}} - c_5^{\{S\}})(c_7^{\{S\}} - c_6^{\{S\}})},\ \alpha_8 = \frac{c_9^{\{S\}} c_{10}^{\{S\}}}{c_8^{\{S\}}(c_8^{\{S\}} - c_9^{\{S\}})(c_8^{\{S\}} - c_{10}^{\{S\}})} \tag{B.10}$$

$$\alpha_9 = \frac{c_8^{\{S\}} c_{10}^{\{S\}}}{c_9^{\{S\}}(c_9^{\{S\}} - c_8^{\{S\}})(c_9^{\{S\}} - c_{10}^{\{S\}})},\ \alpha_{10} = \frac{c_8^{\{S\}} c_9^{\{S\}}}{c_{10}^{\{S\}}(c_{10}^{\{S\}} - c_8^{\{S\}})(c_{10}^{\{S\}} - c_9^{\{S\}})}$$

$$\Omega_{5,1}^{\{2\}} = -\left(\Omega_{5,3}^{\{2\}} + \Omega_{5,4}^{\{2\}}\right),\ \Omega_{5,3}^{\{2\}} = c_5^{\{S\}\times 2}\beta_3,\ \Omega_{5,4}^{\{2\}} = -c_5^{\{S\}\times 2}\beta_4$$

$$\Omega_{6,1}^{\{2\}} = -\left(\Omega_{6,3}^{\{2\}} + \Omega_{6,4}^{\{2\}}\right),\ \Omega_{6,3}^{\{2\}} = c_6^{\{S\}\times 2}\beta_3,\ \Omega_{6,4}^{\{2\}} = -c_6^{\{S\}\times 2}\beta_4$$

$$\Omega_{7,1}^{\{2\}} = -\left(\Omega_{7,3}^{\{2\}} + \Omega_{7,4}^{\{2\}}\right),\ \Omega_{7,3}^{\{2\}} = c_7^{\{S\}\times 2}\beta_3,\ \Omega_{7,4}^{\{2\}} = -c_7^{\{S\}\times 2}\beta_4$$

$$\Omega_{8,1}^{\{2\}} = -\left(\Omega_{8,5}^{\{2\}} + \Omega_{8,6}^{\{2\}} + \Omega_{8,7}^{\{2\}}\right),\ \Omega_{8,5}^{\{2\}} = -c_8^{\{S\}\times 2}\beta_5,\ \Omega_{8,6}^{\{2\}} = -c_8^{\{S\}\times 2}\beta_6,\ \Omega_{8,7}^{\{2\}} = -c_8^{\{S\}\times 2}\beta_7$$

$$\Omega_{9,1}^{\{2\}} = -\left(\Omega_{9,5}^{\{2\}} + \Omega_{9,6}^{\{2\}} + \Omega_{9,7}^{\{2\}}\right),\ \Omega_{9,5}^{\{2\}} = -c_9^{\{S\}\times 2}\beta_5,\ \Omega_{9,6}^{\{2\}} = -c_9^{\{S\}\times 2}\beta_6,\ \Omega_{9,7}^{\{2\}} = -c_9^{\{S\}\times 2}\beta_7$$

$$\Omega_{10,1}^{\{2\}} = -\left(\Omega_{10,5}^{\{2\}} + \Omega_{10,6}^{\{2\}} + \Omega_{10,7}^{\{2\}}\right),\ \Omega_{10,5}^{\{2\}} = -c_{10}^{\{S\}\times 2}\beta_5,\ \Omega_{10,6}^{\{2\}} = -c_{10}^{\{S\}\times 2}\beta_6,\ \Omega_{10,7}^{\{2\}} = -c_{10}^{\{S\}\times 2}\beta_7$$

$$\Omega_{11,1}^{\{2\}} = -\left(\Omega_{11,8}^{\{2\}} + \Omega_{11,9}^{\{2\}} + \Omega_{11,10}^{\{2\}}\right),\ \Omega_{11,8}^{\{2\}} = \beta_8,\ \Omega_{11,9}^{\{2\}} = \beta_9,\ \Omega_{11,10}^{\{2\}} = \beta_{10} \tag{B.11}$$

where

$$\begin{aligned}
\beta_3 &= \frac{1}{c_3^{\{S\}}(c_4^{\{S\}}-c_3^{\{S\}})},\ \beta_4 = \frac{1}{c_4^{\{S\}}(c_3^{\{S\}}-c_4^{\{S\}})} \\
\beta_5 &= \frac{c_6^{\{S\}}+c_7^{\{S\}}}{c_5^{\{S\}}(c_5^{\{S\}}-c_6^{\{S\}})(c_5^{\{S\}}-c_7^{\{S\}})},\ \beta_6 = \frac{c_5^{\{S\}}+c_7^{\{S\}}}{c_6^{\{S\}}(c_6^{\{S\}}-c_5^{\{S\}})(c_6^{\{S\}}-c_7^{\{S\}})}, \\
\beta_7 &= \frac{c_5^{\{S\}}+c_6^{\{S\}}}{c_7^{\{S\}}(c_7^{\{S\}}-c_5^{\{S\}})(c_7^{\{S\}}-c_6^{\{S\}})},\ \beta_8 = \frac{c_9^{\{S\}}+c_{10}^{\{S\}}}{c_8^{\{S\}}(c_8^{\{S\}}-c_9^{\{S\}})(c_8^{\{S\}}-c_{10}^{\{S\}})} \\
\beta_9 &= \frac{c_8^{\{S\}}+c_{10}^{\{S\}}}{c_9^{\{S\}}(c_9^{\{S\}}-c_8^{\{S\}})(c_9^{\{S\}}-c_{10}^{\{S\}})},\ \beta_{10} = \frac{c_8^{\{S\}}+c_9^{\{S\}}}{c_{10}^{\{S\}}(c_{10}^{\{S\}}-c_8^{\{S\}})(c_{10}^{\{S\}}-c_9^{\{S\}})}
\end{aligned} \tag{B.12}$$

$$\begin{aligned}
\Omega_{8,1}^{\{3\}} &= -\left(\Omega_{8,5}^{\{3\}}+\Omega_{8,6}^{\{3\}}+\Omega_{8,7}^{\{3\}}\right),\ \Omega_{8,5}^{\{3\}} = c_8^{\{S\}\times 2}\gamma_5,\ \Omega_{8,6}^{\{3\}} = c_8^{\{S\}\times 2}\gamma_6,\ \Omega_{8,7}^{\{3\}} = c_8^{\{S\}\times 2}\gamma_7 \\
\Omega_{9,1}^{\{3\}} &= -\left(\Omega_{9,5}^{\{3\}}+\Omega_{9,6}^{\{3\}}+\Omega_{9,7}^{\{3\}}\right),\ \Omega_{9,5}^{\{3\}} = c_9^{\{S\}\times 2}\gamma_5,\ \Omega_{9,6}^{\{3\}} = c_9^{\{S\}\times 2}\gamma_6,\ \Omega_{9,7}^{\{3\}} = c_9^{\{S\}\times 2}\gamma_7 \\
\Omega_{10,1}^{\{3\}} &= -\left(\Omega_{10,5}^{\{3\}}+\Omega_{10,6}^{\{3\}}+\Omega_{10,7}^{\{3\}}\right),\ \Omega_{10,5}^{\{3\}} = c_{10}^{\{S\}\times 2}\gamma_5,\ \Omega_{10,6}^{\{3\}} = c_{10}^{\{S\}\times 2}\gamma_6,\ \Omega_{10,7}^{\{3\}} = c_{10}^{\{S\}\times 2}\gamma_7 \\
\Omega_{11,1}^{\{3\}} &= -\left(\Omega_{11,8}^{\{3\}}+\Omega_{11,9}^{\{3\}}+\Omega_{11,10}^{\{3\}}\right),\ \Omega_{11,8}^{\{3\}} = \gamma_8,\ \Omega_{11,9}^{\{3\}} = \gamma_9,\ \Omega_{11,10}^{\{3\}} = \gamma_{10}
\end{aligned} \tag{B.13}$$

where

$$\begin{aligned}
\gamma_5 &= \frac{1}{c_5^{\{S\}}(c_5^{\{S\}}-c_6^{\{S\}})(c_5^{\{S\}}-c_7^{\{S\}})},\ \gamma_6 = \frac{1}{c_6^{\{S\}}(c_6^{\{S\}}-c_5^{\{S\}})(c_6^{\{S\}}-c_7^{\{S\}})}, \\
\gamma_7 &= \frac{1}{c_7^{\{S\}}(c_7^{\{S\}}-c_5^{\{S\}})(c_7^{\{S\}}-c_6^{\{S\}})},\ \gamma_8 = \frac{1}{c_8^{\{S\}}(c_8^{\{S\}}-c_9^{\{S\}})(c_8^{\{S\}}-c_{10}^{\{S\}})} \\
\gamma_9 &= \frac{1}{c_9^{\{S\}}(c_9^{\{S\}}-c_8^{\{S\}})(c_9^{\{S\}}-c_{10}^{\{S\}})},\ \gamma_{10} = \frac{1}{c_{10}^{\{S\}}(c_{10}^{\{S\}}-c_8^{\{S\}})(c_{10}^{\{S\}}-c_9^{\{S\}})}
\end{aligned} \tag{B.14}$$

This method satisfies all available coupling conditions (up through fourth-order) and its base method satisfies all order conditions up through fifth-order when

$$c_9^{\{S\}} = \frac{12 - 15c_{10}^{\{S\}} - 15c_8^{\{S\}} + 20c_{10}^{\{S\}}c_8^{\{S\}}}{15 - 20c_{10}^{\{S\}} - 20c_8^{\{S\}} + 30c_{10}^{\{S\}}c_8^{\{S\}}}. \tag{B.15}$$

REFERENCES

[1] Araujo, A. L., Murua, A., and Sanz-Serna, J. M. Symplectic methods based on decompositions. *SIAM Journal on Numerical Analysis 34*, 5 (1997), 1926–1947.

[2] Bogacki, P., and Shampine, L. A 3(2) pair of Runge–Kutta formulas. *Applied Mathematics Letters 2*, 4 (1989), 321–325.

[3] Calvo, M., De Frutos, J., and Novo, J. Linearly implicit Runge–Kutta methods for advection–reaction–diffusion equations. *Applied Numerical Mathematics 37*, 4 (2001), 535–549.

[4] Cervi, J., and Spiteri, R. J. High-Order Operator Splitting for the Bidomain and Monodomain Models. *SIAM Journal on Scientific Computing 40*, 2 (Jan. 2018), A769–A786. Publisher: Society for Industrial and Applied Mathematics.

[5] Chinomona, R., and Reynolds, D. R. Implicit-explicit multirate infinitesimal GARK methods. *SIAM Journal on Scientific Computing 43*, 5 (Jan. 2021), A3082–A3113.

[6] Cooper, G. J., and Sayfy, A. Additive Runge–Kutta methods for stiff ordinary differential equations. *Math. Comp. 40*, 161 (Jan. 1983), 207–207.

[7] Dorf, R. C., and Bishop, R. H. *Modern Control Systems*, 13th ed. Prentice-Hall, Inc., USA, 2017.

[8] Estep, D., Ginting, V., Ropp, D., Shadid, J. N., and Tavener, S. An a posteriori–a priori analysis of multiscale operator splitting. *SIAM Journal on Numerical Analysis 46*, 3 (Jan. 2008), 1116–1146.

[9] Fish, A. C., and Reynolds, D. R. Adaptive time step control for multirate infinitesimal methods, 2022.

[10] Fish, A. C., Reynolds, D. R., and Roberts, S. B. Implicit-explicit multirate infinitesimal stage-restart methods, 2023.

[11] Forest, E., and Ruth, R. D. Fourth-order symplectic integration. *Physica D: Nonlinear Phenomena 43* (5 1990), 105–117.

[12] Gassner, G., and Kopriva, D. A. A comparison of the dispersion and dissipation errors of gauss and gauss–lobatto discontinuous galerkin spectral element methods. *SIAM Journal on Scientific Computing 33*, 5 (2011), 2560–2579.

[13] Gear, C. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice–Hall, Englewood Cliffs, NJ, 1971.

[14] Günther, M., and Sandu, A. Multirate generalized additive runge kutta methods. *Numerische Mathematik 133*, 3 (2016), 497–524.

[15] Gustafsson, K. Control theoretic techniques for stepsize selection in explicit runge-kutta methods. *ACM Transactions on Mathematical Software (TOMS) 17*, 4 (1991), 533–554.

[16] Gustafsson, K. Control-theoretic techniques for stepsize selection in implicit Runge–Kutta methods. *ACM Transactions on Mathematical Software 20*, 4 (Dec. 1994), 496–517.

[17] Hairer, E., Nørsett, S. P., and Wanner, G. *Solving Ordinary Differential Equations I (2nd Revised. Ed.): Nonstiff Problems*. Springer-Verlag, Berlin, Heidelberg, 1993.

[18] Hairer, E., Nørsett, S. P., and Wanner, G. *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd rev. ed ed. No. 8 in Springer series in computational mathematics. Springer, Heidelberg ; London, 2009.

[19] Hairer, E., and Wanner, G. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, vol. 14. Springer, 1996.

[20] Hochbruck, M., and Ostermann, A. Explicit exponential Runge–Kutta methods for semilinear parabolic problems. *SIAM J. Numer. Anal. 43*, 3 (sep 2005), 1069–1090.

[21] Hochbruck, M., and Ostermann, A. Exponential Runge–Kutta methods for parabolic problems. *Applied Numerical Mathematics 53*, 2 (May 2005), 323–339.

[22] Inc., W. R. Mathematica, Version 13.2. Champaign, IL, 2022.

[23] Kennedy, C. A., and Carpenter, M. H. Additive Runge–Kutta schemes for convection–diffusion–reaction equations. *Applied Numerical Mathematics 44*, 1-2 (Jan. 2003), 139–181.

[24] Kennedy, C. A., and Carpenter, M. H. Diagonally implicit Runge–Kutta methods for stiff ODEs. *Applied Numerical Mathematics 146* (Dec. 2019), 221–244.

[25] Kennedy, C. A., and Carpenter, M. H. Higher-order additive Runge–Kutta schemes for ordinary differential equations. *Applied Numerical Mathematics 136* (Feb. 2019), 183–205.

[26] Kennedy, C. A., and Carpenter, M. H. Higher-order additive Runge–Kutta schemes for ordinary differential equations. *Applied Numerical Mathematics 136* (Feb. 2019), 183–205.

[27] Keyes, D. E., McInnes, L. C., Woodward, C., Gropp, W., Myra, E., Pernice, M., Bell, J., Brown, J., Clo, A., Connors, J., Constantinescu, E., Estep, D., Evans, K., Farhat, C., Hakim, A., Hammond, G., Hansen, G., Hill, J., Isaac, T., Jiao, X., Jordan, K., Kaushik, D., Kaxiras, E., Koniges, A., Lee, K., Lott, A., Lu, Q., Magerlein, J., Maxwell, R., McCourt, M., Mehl, M., Pawlowski, R., Randles, A. P., Reynolds, D., Rivière, B., Rüde, U., Scheibe, T., Shadid, J., Sheehan, B., Shephard, M., Siegel, A., Smith, B., Tang, X., Wilson, C., and Wohlmuth, B. Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications 27*, 1 (Feb. 2013), 4–83. Publisher: SAGE Publications Ltd STM.

[28] Knoth, O., and Wolke, R. Implicit-explicit Runge-Kutta methods for computing atmospheric reactive flows. *Applied Numerical Mathematics 28*, 2 (1998), 327 – 341.

[29] Langville, A. N., and Stewart, W. J. The kronecker product and stochastic automata networks. *Journal of Computational and Applied Mathematics 167*, 2 (June 2004), 429–447.

[30] Luan, V. T. *High-order exponential integrators*. PhD thesis, University of Innsbruck, 2014.

[31] Luan, V. T., Chinomona, R., and Reynolds, D. R. A New Class of High-Order Methods for Multirate Differential Equations. *SIAM Journal on Scientific Computing 42*, 2 (Jan. 2020), A1245–A1268.

[32] Luan, V. T., Chinomona, R., and Reynolds, D. R. Multirate exponential Rosenbrock methods, 2021.

[33] Luan, V. T., and Ostermann, A. Explicit exponential Runge–Kutta methods of high order for parabolic problems. *Journal of Computational and Applied Mathematics 256* (Jan. 2014), 168–179.

[34] Luan, V. T., and Ostermann, A. Exponential Rosenbrock methods of order five — construction, analysis and numerical comparisons. *Journal of Computational and Applied Mathematics 255* (Jan. 2014), 417–431.

[35] Luan, V. T., and Ostermann, A. Parallel exponential rosenbrock methods. *Computers & Mathematics with Applications 71*, 5 (Mar. 2016), 1137–1150.

[36] Marchuk, G. I. Some application of splitting-up methods to the solution of mathematical physics problems. *Aplikace Matematiky 13*, 2 (1968), 103–132.

[37] McLachlan, R. I., and Quispel, G. R. W. Splitting methods. *Acta Numerica 11* (Jan. 2002), 341–434.

[38] Prince, P., and Dormand, J. High order embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics 7*, 1 (1981), 67–75.

[39] Roberts, S., Loffeld, J., Sarshar, A., Woodward, C. S., and Sandu, A. Implicit multirate gark methods. *Journal of Scientific Computing 87*, 1 (2021), 1–32.

[40] Roberts, S., Popov, A. A., and Sandu, A. ODE test problems: a MATLAB suite of initial value problems. *CoRR abs/1901.04098* (2019).

[41] Roberts, S., Sarshar, A., and Sandu, A. Coupled multirate infinitesimal GARK schemes for stiff systems with multiple time scales. *SIAM Journal on Scientific Computing 42*, 3 (2020), A1609–A1638.

[42] Ropp, D. L., and Shadid, J. N. Stability of operator splitting methods for systems with indefinite operators: reaction-diffusion systems. *Journal of Computational Physics 203*, 2 (Mar. 2005), 449–466.

[43] Rota, G., Kahaner, D., and Odlyzko, A. On the foundations of combinatorial theory. VIII. Finite operator calculus. *Journal of Mathematical Analysis and Applications 42*, 3 (June 1973), 684–760.

[44] Sandu, A. A class of multirate infinitesimal GARK methods. *CoRR abs/1808.02759* (2018).

[45] Sandu, A. A class of multirate infinitesimal GARK methods. *SIAM Journal on Numerical Analysis 57*, 5 (Jan. 2019), 2300–2327.

[46] Sandu, A., and Günther, M. A generalized-structure approach to additive Runge–Kutta methods. *SIAM Journal on Numerical Analysis 53*, 1 (2015), 17–42.

[47] Sarshar, A., Roberts, S., and Sandu, A. Design of High-Order Decoupled Multirate GARK Schemes. *SIAM Journal on Scientific Computing 41*, 2 (Jan. 2019), A816–A847.

[48] Savcenco, V., Hundsdorfer, W., and Verwer, J. A multirate time stepping strategy for stiff ordinary differential equations. *BIT 47* (03 2007), 137–155.

[49] Schlegel, M., Knoth, O., Arnold, M., and Wolke, R. Multirate Runge-Kutta schemes for advection equations. *Journal of Computational and Applied Mathematics 226*, 2 (2009), 345–357.

[50] Schlegel, M., Knoth, O., Arnold, M., and Wolke, R. Multirate Runge–Kutta schemes for advection equations. *Journal of Computational and Applied Mathematics 226*, 2 (Apr. 2009), 345–357.

[51] Schlegel, M., Knoth, O., Arnold, M., and Wolke, R. Multirate implicit-explicit time integration schemes in atmospheric modelling. *AIP Conference Proceedings 1281*, 1 (2010), 1831–1834.

[52] Sexton, J. M., and Reynolds, D. R. Relaxed multirate infinitesimal step methods. *arxiv:1808.03718 [math.NA]* (2018).

[53] Skvortsov, L. Accuracy of Runge–Kutta methods applied to stiff problems. *Computational Mathematics and Mathematical Physics 43* (09 2003), 1320–1330.

[54] Söderlind, G. Automatic control and adaptive time-stepping. *Numerical Algorithms 31* (01 2002), 281–310.

[55] Söderlind, G. Digital filters in adaptive time-stepping. *ACM Trans. Math. Softw. 29*, 1 (mar 2003), 1–26.

[56] Söderlind, G. Time-step selection algorithms: Adaptivity, control, and signal processing. *Applied Numerical Mathematics 56* (2006), 488–502.

[57] Spiteri, R. J., and Wei, S. Fractional-step Runge–Kutta methods: Representation and linear stability analysis. *Journal of Computational Physics 476* (Mar. 2023), 111900.

[58] Strang, G. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis 5*, 3 (Sept. 1968), 506–517.

[59] Tokman, M. A new class of exponential propagation iterative methods of Runge–Kutta type (EPIRK). *Journal of Computational Physics 230*, 24 (Oct. 2011), 8762–8778.

[60] Verner, J. H. Explicit Runge–Kutta methods with estimates of the local truncation error. *SIAM Journal on Numerical Analysis 15*, 4 (1978), 772–790.

[61] Wensch, J., Knoth, O., and Galant, A. Multirate infinitesimal step methods for atmospheric flow simulation. *BIT Numerical Mathematics 49*, 2 (2009), 449–473.

[62] Wensch, J., Knoth, O., and Galant, A. Multirate infinitesimal step methods for atmospheric flow simulation. *BIT Numer. Math. 49*, 2 (June 2009), 449–473.

[63] Zharovsky, E., Sandu, A., and Zhang, H. A class of implicit-explicit two-step Runge–Kutta methods. *SIAM Journal on Numerical Analysis 53*, 1 (2015), 321–341.

[64] Zonneveld, J. Automatic integration of ordinary differential equations. *Stichting Mathematisch Centrum. Rekenafdeling 743*, 63 (1963).