


Fall 2017

# High-order Relaxed Multirate Infinitesimal Step Methods for Multiphysics Applications

Jean Sexton  
jmsexton@smu.edu

Follow this and additional works at: [https://scholar.smu.edu/hum\\_sci\\_mathematics\\_etds](https://scholar.smu.edu/hum_sci_mathematics_etds)

 Part of the [Numerical Analysis and Computation Commons](#), [Ordinary Differential Equations and Applied Dynamics Commons](#), [Other Physical Sciences and Mathematics Commons](#), and the [Theory and Algorithms Commons](#)

---

## Recommended Citation

Sexton, Jean, "High-order Relaxed Multirate Infinitesimal Step Methods for Multiphysics Applications" (2017). *Mathematics Theses and Dissertations*. 1.

[https://scholar.smu.edu/hum\\_sci\\_mathematics\\_etds/1](https://scholar.smu.edu/hum_sci_mathematics_etds/1)

This Dissertation is brought to you for free and open access by the Mathematics at SMU Scholar. It has been accepted for inclusion in Mathematics Theses and Dissertations by an authorized administrator of SMU Scholar. For more information, please visit <http://digitalrepository.smu.edu>.

HIGH-ORDER RELAXED MULTIRATE INFINITESIMAL STEP METHODS  
FOR MULTIPHYSICS APPLICATIONS

Approved by:

---

Dr. Daniel R. Reynolds  
Associate Professor of Mathematics

---

Dr. Usama El-Shamy  
Associate Professor of Civil and  
Environmental Engineering

---

Dr. Thomas Hagstrom  
Professor of Mathematics

---

Dr. Barry Lee  
Associate Professor of Mathematics

HIGH-ORDER RELAXED MULTIRATE INFINITESIMAL STEP METHODS  
FOR MULTIPHYSICS APPLICATIONS

A Dissertation Presented to the Graduate Faculty of the  
Dedman College

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

Jean Sexton

B.S., Mathematics, Southern Methodist University  
M.S., Computational and Applied Mathematics, Southern Methodist University

December 16, 2017

Copyright (2017)

Jean Sexton

All Rights Reserved

## ACKNOWLEDGMENTS

I would like to acknowledge generally everyone who supported me in my PhD work, both in the initial stages as well as for the last portion. I would like to thank my advisor, Dr. Daniel Reynolds, as well as my committee, Dr. Barry Lee, Dr. Thomas Hagstrom, and Dr. Usama El-Shamy. I really appreciate how our upper-level computational courses, Graduate Student Seminar, and SIAM events were impacted and informed by contributions from my fellow graduate students. Specifically, I am grateful to Jessica and Mahnprit Jutley, Fritz Juhnke, Claudia Castro-Castro, John Lagrone, and Jennifer Swenson. I would like to thank my parents, my brother, my sister, and my boyfriend for the support and the inspiration. I would like to thank everyone from the SMU Neuhoff Center for the support and community throughout my time at SMU. Thanks to all y'all for showing me that there's always something more to learn.

Sexton, Jean

B.S., Mathematics, Southern Methodist University  
M.S., Computational and Applied Mathematics, Southern Methodist University

High-order Relaxed Multirate Infinitesimal Step Methods  
for Multiphysics Applications

Advisor: Dr. Daniel R. Reynolds

Doctor of Philosophy degree conferred December 16, 2017

Dissertation completed October 27, 2017

In this work, we consider numerical methods for integrating multirate ordinary differential equations. We are interested in the development of new multirate methods with good stability properties and improved efficiency over existing methods. We discuss the development of multirate methods, particularly focusing on those that are based on Runge-Kutta theory. We introduce the theory of Generalized Additive Runge-Kutta methods proposed by Sandu and Günther [20, 44]. We also introduce the theory of Recursive Flux Splitting Multirate Methods with Sub-cycling described by Schlegel [49], as well as the Multirate Infinitesimal Step methods this work is based on. We propose a generic structure called Flexible Multirate Generalized-Structure Additively-Partitioned Runge-Kutta methods which allows for optimization and more rigorous analysis. We also propose a specific class of higher-order methods, called Relaxed Multirate Infinitesimal Step Methods. We will leverage GARK theories to develop new theory about the stability and accuracy of these new methods.

## TABLE OF CONTENTS

LIST OF FIGURES .....	viii
LIST OF TABLES .....	xi
CHAPTER	
1. Introduction .....	1
1.1. Fundamental Concepts of Multirate Ordinary Differential Equations .....	1
1.2. Review of Known Methods .....	5
1.2.1. Extrapolation Methods for Multirate ODEs .....	7
1.2.2. Kværno-Rentrop Multirate Partitioned Runge-Kutta Methods .....	8
1.2.3. Multirate Infinitesimal Step (MIS) .....	11
2. Detailed Background Theory .....	14
2.1. Generalized-Structure Additively-Partitioned Runge Kutta Methods .....	14
2.2. Multirate Generalized Additive Runge Kutta .....	18
2.3. Creating Multirate Methods using GARK theory .....	21
2.3.1. More Flexible Structure for New Methods .....	29
2.4. MIS methods as an extension of GARK theory .....	30
2.5. Linear Stability Theory for Multirate Systems .....	34
3. Framework for Relaxed Multirate Infinitesimal Step Methods .....	39
3.1. Proposed Flexible Multirate GARK (fmGARK) structure .....	39
3.2. Structure .....	39
3.3. Implementation and Memory Considerations .....	45
3.4. Order Conditions for Relaxed Framework .....	50
4. Proposed Relaxed MIS methods and comparative fmGARK methods .....	57
4.1. Same strategy: RFSMR or MIS .....	59

4.2.	Relaxed Multirate Infinitesimal Step Methods: preserves linear invariants . .	61
4.3.	Optimized methods . . . . .	65
4.4.	Comparisons . . . . .	69
5.	Numerical Tests: Observed Order . . . . .	70
5.1.	Test Problems . . . . .	70
5.1.1.	Inverter-chain Test Problem . . . . .	70
5.1.2.	Kuhn Linear Test Problem . . . . .	76
5.1.3.	Brusselator Test Problem . . . . .	78
5.2.	Test Setup . . . . .	81
5.3.	Numerical Convergence and Efficiency Results . . . . .	84
5.3.1.	Inverter-chain Results . . . . .	85
5.3.2.	Kuhn Linear Results . . . . .	89
5.3.3.	Brusselator Results . . . . .	92
6.	Stability Results: Theoretical & Numerical . . . . .	95
6.1.	Deriving a Stability Concept for GARK methods based on Kværno's stability	95
6.2.	Comparing Stability for Base Methods with High Order . . . . .	101
6.3.	Comparing Stability Optimization . . . . .	105
7.	Conclusions . . . . .	109

## APPENDIX



## LIST OF FIGURES

Figure	Page	
2.1	$\kappa = 10$ and $\kappa = 100$ for similarly constructed methods. Note that the $\xi$ axis changes for the $\kappa = 100$ plot on the right, to show that the stability area is diminished for larger $\kappa$ . The yellow area is stable by our stability concept. $\eta < 0$ shows stability parameters where the off-diagonal coupling terms have opposite signs. One explanation for the extended stability in the $\eta = -.6$ and $\xi = -.25$ on the $\kappa = 10$ plot is that the opposite signs on the off-diagonal coupling terms have a net effect of decreasing the stability requirements. ....	38
4.1	Choices of $c_2$ and $c_3$ on these lines result in the 3rd order RFMSR method, otherwise the method is 2nd order .....	62
4.2	Choices of $c_2$ and $c_3$ on these lines result in the 4th order RMIS method, otherwise the method is 3rd order .....	66
4.3	Choices of $c_2$ and $c_3$ on these lines result in the higher order method, otherwise the method is lower order .....	69
5.1	Solutions for the inverter chain problem with $n_i = 100$ : Dotted lines represent the fastest components, while thick lines represent the slowest, i.e. $\mathbf{y}_1$ is the blue dotted curve at the top-left and $\mathbf{y}_{100}$ remains at the value 2.5 at the final time, $t = 7$ . ....	74
5.2	Small differences in the initial integration solution may result in disparate solution values .....	75
5.3	Solutions for the Kuhn test problem (5.1): note that the second component reaches equilibrium much more rapidly than, and does not vary by as much as, the first component. ....	77
5.4	Solutions for the Brusselator test problem (5.6): $\mathbf{y}_3$ initially rapidly approaches $b = 2.5$ , then varies slowly thereafter.....	80

5.5	Convergence for the inverter chain problem: note that for this problem the slope for order of accuracy is consistent for all methods except Algorithm 6 w/ Base Method 4.17. The flattening of the error for this method may be due to inaccurate capturing of the initial integration window causing this method to converge to a solution which is approximately $10^{-8}$ away from the reference solution. ....	87
5.6	Efficiency for the inverter chain problem: note that for this problem the most efficient methods for larger error values are Algorithm 5 w/ Base Method 4.17 and Algorithm 6 w/ Base Method 4.17, while for smaller errors Algorithm 5 w/ Base Method 4.17 is the clear winner; all other methods perform comparably well. ....	88
5.7	The numerical order observed is close to the expected values .....	90
5.8	The higher order of accuracy methods are more efficient for almost all $h$ values tested.....	91
5.9	The optimized method and the RMIS seem to do best .....	93
5.10	Method efficiency for this problem is dependent on both the order of accuracy and on the constant multiplier on the error terms.....	94
6.1	Linear stability plots for time-scale separation and multirate-method factors $\kappa = m = 10$ using 3/8 – Rule: our RMIS method is on the left and the RFSMR method is on the right. ....	102
6.2	Linear stability plots for time-scale separation and multirate-method factors $\kappa = m = 100$ , using 3/8 – Rule: our RMIS method is on the left and the RFSMR method is on the right.....	102
6.3	Linear stability plots for time-scale separation and multirate-method factors $\kappa = m = 10$ : our RMIS method is on the left and the RFSMR method is on the right. Note that the stability areas are slightly larger than in Figure 6.1, although they have similar shape, indicating that multirate problems can potentially take a larger time-step due to the increased region of stability. The different placement of the region which is extended indicates that some multirate problems which would be stable for $\kappa = 10$ using the 3/8 – Rule would not be stable for this method, and vice versa.....	104

6.4	Linear stability plots for time-scale separation and multirate-method factors $\kappa = m = 10$ : our RMIS method is on the left and the RFSMR method is on the right. Note that the stability areas are slightly larger than in Figure 6.2, although they have similar shape, indicating that multirate problems can potentially take a larger time-step due to the increased region of stability. The different placement of the region which is extended indicates that some multirate problems which would be stable for $\kappa = 100$ using the 3/8 – Rule would not be stable for this method, and vice versa. ....	105
6.5	For most of the base methods tested, the RMIS methods had a stable area around 35.5% of the plotted area. The RFSMR methods, on the other hand, had a more varied percentage area, going from 35.2% to 35.8%. The total percentage area did not seem to favor either the RMIS or the RFSMR in terms of stability. We used these sets of base methods to pick a base method for both the RMIS and the RFSMR in order to compare the largest area of stability. ....	106
6.6	Same $\kappa = m = 10$ , best stability area RMIS on the left and best stability area RFSMR on the right .....	107
6.7	Same $\kappa = m = 100$ , best stability area RMIS on the left and best stability area RFSMR on the right .....	107

## LIST OF TABLES

Table		Page
5.1	Table of Method Efficiency Properties: The base method choice determines the number of stages, and affects the ratio of fast stages to slow stages in the overall GARK table. ....	85
5.2	Table of Inverter-Chain Convergence: The order of accuracy is expected, although our measure of observed order is limited by the inclusion of all results with error between 1 and $10^{-9}$ .....	86
5.3	Table of Kuhn Convergence: The observed order of accuracy is slightly better than expected based on theory .....	89
5.4	Table of Brusselator Convergence: The observed order of accuracy is slightly better than expected based on theory, and the observed order for the optimized method is significantly better .....	92

Informal dedication to Dr. Pepper of Dublin, TX, an inspiration to us all.

In a homogeneous solution sense, I would like to make a formal dedication to my entire family, past, present and future. In a particular solution sense, I would like to make a formal dedication to Charles Colvin, Emmeline Miles, Jessica Jutley, Jenny Rohde, Andrea Fernandez and all those who have walked these years with me day by day.

## Chapter 1

### Introduction

This thesis gives context for why higher-order multirate methods are useful, and demonstrates newly developed methods with useful properties. These methods are motivated by multiphysics, multiscale real-world application problems which are constructed by coupling physical processes with potential disparate length and time scales together. Chemical reaction networks are one example of this type of multirate problem, the brusselator test problem being one particular example. The compressible Euler equations from fluid dynamics which are used in numerical weather prediction is an example of a physical model which when combined with adaptive mesh refinement or chemistry (or other physical models) has a multirate character. Specific multirate methods have been developed to address these problems when there is both significant separation between the characteristic time scales and significant coupling between the processes. This thesis focuses on developing and testing a set of efficient, fully coupled fourth-order multirate methods with comparable stability properties to leading existing third-order multirate methods.

#### 1.1. Fundamental Concepts of Multirate Ordinary Differential Equations

One of the earliest references to multirate methods is found in a technical report from 1980 by Gear [15]. In the introduction, he defines “A multirate method for integrating ordinary differential equations is one in which different equations are integrated using different time steps.” We define a *multirate method* for integrating ordinary differential equations as any method that satisfies Gear’s definition, or any method that uses operator splitting to integrate the integrand using different time steps. These methods are most useful when the operator splitting is constructed so that the operator which re-

quires more work also requires a larger characteristic time-step size than other operators. We will be considering a specialization of a problem posed as system of ordinary differential equations  $\mathbf{y}'(t) = f(t, \mathbf{y})$ ,  $\mathbf{y}(t_0) = \mathbf{y}_0$ . An operator splitting for many rates yields  $\mathbf{y}'(t) = f(t, \mathbf{y}) = \sum_{i=1}^{i=N} f^{\{i\}}(t, \mathbf{y})$ ,  $\mathbf{y}(t_0) = \mathbf{y}_0$ . The splitting is defined so that the characteristic timescale corresponding to  $f^{\{i\}}(t, \mathbf{y})$  is non-increasing as  $i$  increases. This implies  $f^{\{1\}}(t, \mathbf{y})$  is the slowest operator since no other operator has a larger stable step size. Correspondingly,  $f^{\{N\}}(t, \mathbf{y})$  is the fastest operator since no other operator has a smaller stable step size. In most cases, a many-rate problem can be rewritten as nested two-rate problems. For instance, a method for integrating two-rates can be used on the two fastest rates as part of the integration of the overall problem.

$$\begin{aligned} \mathbf{y}'(t) &= f^{\{1\}}(t, \mathbf{y}) + f^{\{2\}}(t, \mathbf{y}) + f^{\{3\}}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0 \\ f^{\{s\}}(t, \mathbf{y}) &= f^{\{1\}}(t, \mathbf{y}) \\ f^{\{f\}}(t, \mathbf{y}) &= f^{\{2\}}(t, \mathbf{y}) + f^{\{3\}}(t, \mathbf{y}) \\ \mathbf{y}'(t) &= f^{\{s\}}(t, \mathbf{y}) + f^{\{f\}}(t, \mathbf{y}) \end{aligned}$$

This will be discussed in further detail for the methods developed in this thesis.

In a two-rate problem, we partition the function  $f(t, \mathbf{y})$  into a fast portion and a slow portion. One common way is defining some additive partition of the function  $f(t, \mathbf{y})$  into a fast function  $f^{\{f\}}$  and a slow function  $f^{\{s\}}$ . This is called the *additive problem formulation*. Another common way is defining some partition of the solution  $y$  between fast components  $\mathbf{y}^{\{f\}}$  and slow components  $\mathbf{y}^{\{s\}}$ . This is called the *partitioned problem formulation*. These formulations are mathematically defined below.

The partitioned two-rate problem formulation consists of a system of ODEs which are partitioned into slow and fast components, with their associated initial conditions,

$$\begin{aligned} \left[ \mathbf{y}^{\{s\}}(t) \right]' &= f^{\{s\}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)), \quad t \geq t_0 & \mathbf{y}^{\{s\}}(t_0) &= \mathbf{y}_0^{\{s\}} \\ \left[ \mathbf{y}^{\{f\}}(t) \right]' &= f^{\{f\}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)), \quad t \geq t_0 & \mathbf{y}^{\{f\}}(t_0) &= \mathbf{y}_0^{\{f\}}. \end{aligned}$$

This type of partitioning naturally arises when components of the solution represent different types of physical variables. For instance, many physical systems from fluid dynamics may

be defined in terms of primitive variables such as pressure, density, and temperature which are then discretized at particular positions in space to be represented as a system of ODEs. A natural splitting may be to split by components along the different types of component variables. This splitting has the benefit of being physically motivated, and not changing as the time integration progresses. Another type of component-based splitting for variables at specific physical locations can be based upon domain decomposition. This type of splitting is particularly helpful for grid-based problems with uneven grid refinement. Different grid spacing can affect the requirements for a stable time step. Finally, component-based splittings could change dynamically between time-integration steps based on where in the problem domain the quickly changing dynamics is happening.

The additive two-rate problem formulation consists of a system of ODEs which can be separated into the sum of a slow function and a fast function:

$$\mathbf{y}'(t) = f(t, \mathbf{y}), \quad f(t, \mathbf{y}) = f^{\{s\}}(t, \mathbf{y}) + f^{\{f\}}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0.$$

We note that a partitioned problem can be reformulated as an additive problem,

$$\begin{aligned} \mathbf{y}'(t) &= \begin{bmatrix} \mathbf{y}^{\{s\}} \\ \mathbf{y}^{\{f\}} \end{bmatrix}' = \begin{bmatrix} \overline{f^{\{s\}}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)) \\ \overline{f^{\{f\}}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)) \end{bmatrix} & (1.1) \\ &= \begin{bmatrix} 0 \\ \overline{f^{\{f\}}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)) \end{bmatrix} + \begin{bmatrix} \overline{f^{\{s\}}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)) \\ 0 \end{bmatrix} \\ &= f^{\{s\}}(t, \mathbf{y}) + f^{\{f\}}(t, \mathbf{y}), & (1.2) \end{aligned}$$

so the additive formulation is more general than the partitioned formulation. We note that some sources consider reformulating additive problems into partitioned problems [1], via the decomposition

$$\begin{aligned} \begin{bmatrix} \mathbf{y}^{\{s\}} \\ \mathbf{y}^{\{f\}} \end{bmatrix}' &= \begin{bmatrix} f^{\{s\}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)) \\ f^{\{f\}}(t, \mathbf{y}^{\{s\}}(t), \mathbf{y}^{\{f\}}(t)) \end{bmatrix} \\ \mathbf{y}_0 &= \mathbf{y}_0^{\{s\}} + \mathbf{y}_0^{\{f\}}. \end{aligned}$$



However, since this decomposition is not unique due to the infinitely many different choices for partitioning the initial condition, in this work we focus on additive approaches to multi-rate problems. Each of these types of splittings has a corresponding adaptation of a standard Runge-Kutta method.

Partitioned Runge-Kutta methods (PRKs) and their order conditions are discussed in detail in the book by Hairer, Nørsett and Wanner [22]. PRK methods are represented through a pair of Butcher tableau such as

$$\begin{array}{c|c} c^{\{1\}} & A^{\{1\}} \\ \hline & b^{\{1\}}\tau \end{array}$$

$$\begin{array}{c|c} c^{\{2\}} & A^{\{2\}} \\ \hline & b^{\{2\}}\tau \end{array}$$

where the number of stages in  $A^{\{1\}}$  and in  $A^{\{2\}}$  are  $s^{\{1\}}$  and  $s^{\{2\}}$  respectively. The stage and the solution updates are:

$$\mathbf{f}_i^{\{1\}} = f^{\{1\}} \left( t_n + c_i^{\{1\}}h, \mathbf{y}_n^{\{1\}} + h \sum_{j=1}^{s^{\{1\}}} a_{ij}^{\{1\}} \mathbf{k}_j^{\{1\}}, \mathbf{y}_n^{\{2\}} + h \sum_{j=1}^{s^{\{2\}}} a_{ij}^{\{2\}} \mathbf{k}_j^{\{2\}} \right), \quad i = 1, 2, \dots, s^{\{1\}},$$

$$\mathbf{k}_i^{\{1\}} = \mathbf{y}_n^{\{1\}} + h \sum_{j=1}^{s^{\{1\}}} a_{ij}^{\{1\}} \mathbf{f}_j^{\{1\}}, \quad i = 1, 2, \dots, s^{\{1\}},$$

$$\mathbf{y}_{n+1}^{\{1\}} = \mathbf{y}_n^{\{1\}} + h \sum_{i=1}^{s^{\{1\}}} b_i^{\{1\}} \mathbf{k}_i^{\{1\}},$$

$$\mathbf{f}_i^{\{2\}} = f^{\{2\}} \left( t_n + c_i^{\{2\}}h, \mathbf{y}_n^{\{1\}} + h \sum_{j=1}^{s^{\{1\}}} a_{ij}^{\{1\}} \mathbf{k}_j^{\{1\}}, \mathbf{y}_n^{\{2\}} + h \sum_{j=1}^{s^{\{2\}}} a_{ij}^{\{2\}} \mathbf{k}_j^{\{2\}} \right), \quad i = 1, 2, \dots, s^{\{2\}},$$

$$\mathbf{k}_i^{\{2\}} = \mathbf{y}_n^{\{2\}} + h \sum_{j=1}^{s^{\{2\}}} a_{ij}^{\{2\}} \mathbf{f}_j^{\{2\}}, \quad i = 1, 2, \dots, s^{\{2\}},$$

$$\mathbf{y}_{n+1}^{\{2\}} = \mathbf{y}_n^{\{2\}} + h \sum_{i=1}^{s^{\{2\}}} b_i^{\{2\}} \mathbf{k}_i^{\{2\}}.$$

Additive Runge-Kutta (ARK) methods and their order conditions are presented in detail in the 2003 paper by Kennedy and Carpenter [25]. ARK methods are represented using a pair of Butcher tableau, such as

$$\begin{array}{c|c|c|c} c^{\{1\}} & A^{\{1\}} & A^{\{2\}} & c^{\{2\}} \\ \hline & b^{\{1\}\tau} & b^{\{2\}\tau} & \end{array}$$

The stages and the solution update given  $s^{\{1\}} = s^{\{2\}} = s$  are:

$$\begin{aligned} \mathbf{k}_i &= \mathbf{y}_n + h \sum_{j=1}^s a_{ij}^{\{1\}} f^{\{1\}} \left( t_n + c_i^{\{1\}} h, \mathbf{k}_j \right) + h \sum_{j=1}^s a_{ij}^{\{2\}} f^{\{2\}} \left( t_n + c_i^{\{2\}} h, \mathbf{k}_j \right), \quad i = 1, \dots, s, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{i=1}^s b_i^{\{1\}} f^{\{1\}} \left( t_n + c_i^{\{1\}} h, \mathbf{k}_i \right) + h \sum_{i=1}^s b_i^{\{2\}} f^{\{2\}} \left( t_n + c_i^{\{2\}} h, \mathbf{k}_i \right). \end{aligned}$$

## 1.2. Review of Known Methods

The family of methods developed in this thesis are demonstrated for problems where both the fast and the slow operators are integrated explicitly. Similar techniques allow for methods where the fast operator is integrated implicitly while the slow operator is integrated explicitly. The background in this section is intended to give a general overview of the historical development of multirate methods which depend upon Runge-Kutta theory, as well as explaining the current state of the field.

In order to simplify the presentation of each method, we will consider a slow time-step of size  $dt_s = h$ . In order to derive an integer time-scale separation, we use  $\overline{dt}_f$  to indicate the problem's fast characteristic time-scale, and  $dt_f$  to indicate the fast time-step our method will use. For the sake of notation, we use  $\overline{dt}_s$  to indicate the slow characteristic time-scale, which will be equal to  $dt_s$ , the slow time-step our method will use. We may then define the *time-scale separation* between the problem's characteristic time-scales as  $\overline{m} = \overline{dt}_s / \overline{dt}_f$ . The methods will utilize this factor to determine the ratio of the slow time-step and the fast time-step sizes, i.e. by rounding this up to the nearest integer,  $\overline{m} \approx m$  and  $\overline{m} < m$ , we may define a fast step size  $dt_f = h/m$  so that it is a good approximation to  $\overline{dt}_f$ . Schlegel postulated that a more efficient method could be created by allowing the number of fast steps per slow

stage to vary within the method in 2012 [49]. Recent work by Bremicker–Trübelhorn and Ortleb has investigated variable fast step sizes for specific numerical simulations by allowing  $dt_{f,i} = h/m_i$  for each fast micro-step  $i$  [4].

The order of accuracy of a multirate method describes how the error behaves as  $h \rightarrow 0$ . In order to investigate this order concept, we must define our terms. The *fast component method* is the method which results if the multirate method is applied to a problem where the slow function is equivalent to the zero function. The *slow component method* is the method which results if the multirate method is applied to a problem where the fast function is equivalent to the zero function. The *coupling method* is the interpolation method which connects the slow component method to the fast component method. This can be further split into interpolation using slow input data and interpolation using fast input data. When  $dt_f = h/m$ , the fast component method is typically a composition of  $m$  steps of a fast base method, while the slow component method is typically one step of a slow base method. In this typical case, the multirate method order depends on the base method orders and on the coupling order.

With these definitions in place, we may precisely introduce the simplest multirate integration methods. In these methods, one integrates only a single time scale at a time, using the results from the first portion as data when integrating the second portion. Without a loss of generality, we may consider use of an explicit Euler base method for integration of both the slow step and the corresponding  $m$  fast micro-steps,

$$\mathbf{y}_{n+1}^{\{s\}} = \mathbf{y}_n^{\{s\}} + hf^{\{s\}}(t, \mathbf{y}_n^{\{s\}}, \mathbf{y}_n^{\{f\}}) \quad (1.3)$$

$$\mathbf{y}_{n+\frac{i}{m}}^{\{f\}} = \mathbf{y}_{n+\frac{i-1}{m}}^{\{f\}} + \frac{h}{m} f^{\{f\}}\left(t, \mathbf{Y}_{n+\frac{i-1}{m}}, \mathbf{y}_{n+\frac{i-1}{m}}^{\{f\}}\right) \quad i = 1, \dots, m. \quad (1.4)$$

Here  $\mathbf{Y}_{n+\frac{i-1}{m}}$  is an approximation of the slow solution,  $\mathbf{y}^{\{s\}}$ , at the fast sub-step times,  $t_{n+\frac{i-1}{m}} = t_n + (i-1)h$ . Within the *fastest-first* strategy, the fast steps are performed first, using one of

$$\mathbf{Y}_{n+\frac{i-1}{m}} = \mathbf{y}_n^{\{s\}}, \quad \text{or} \quad (1.5)$$

$$\mathbf{Y}_{n+\frac{i-1}{m}} = \frac{2m-i+1}{m} \mathbf{y}_n^{\{s\}} - \frac{m-i+1}{m} \mathbf{y}_{n-1}^{\{s\}} \quad (1.6)$$

for extrapolation of the slow solution values. Alternately, within the *slowest-first* strategy, the slow step is performed first, using one of

$$\mathbf{Y}_{n+\frac{i-1}{m}} = \mathbf{y}_{n+1}^{\{s\}}, \quad \text{or} \quad (1.7)$$

$$\mathbf{Y}_{n+\frac{i-1}{m}} = \frac{m-i+1}{m} \mathbf{y}_n^{\{s\}} + \frac{i-1}{m} \mathbf{y}_{n+1}^{\{s\}} \quad (1.8)$$

for interpolation of the slow solution values within the fast sub-steps. Gear is credited with the original concept from a manuscript titled "Automatic Multirate Methods for Ordinary Differential Equations" from 1980, and continued to publish an article in BIT with Wells from IBM in 1984, citing the earlier work [15, 16]. This definition was used in Well's thesis from 1982 [56].

Due to the decoupling of the slow and fast components, these methods are *at best* first-order accurate (technically, only when using (1.6) or (1.8)), *even if higher-order base methods than Euler are used*. This fastest-first and slowest-first terminology was used by Sandu and Constantinescu in 2009 to describe their multirate method [41]. Recently in 2015, Fok and Rosales focused on creating linearly higher-order multirate methods by improving the interpolation between the fast base method and the slow base method directly [14]. They construct a cubic interpolant by assuming a specific Runge-Kutta method, and matching Taylor series error terms for specific Runge-Kutta stages.

### 1.2.1. Extrapolation Methods for Multirate ODEs

The earliest reference to multirate extrapolation methods is from Engstler and Lubich's 1997 paper [11]. They generate higher-order solutions for the multirate system from these first-order methods using extrapolation. To this end, successive higher-order solutions are constructed from lower-order solutions at differing time step sizes using the Aitken-Neville formula,

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{(n_j/n_{j-1}) - 1}, \quad j \leq m, \quad k < j,$$

where  $n_i < n_{i+1} \in \mathbb{N}$ ,  $dt_{f,i} = h/n_i$  and  $T_{i,1} = y_{dt_{f,i}}(t+h)$ .

Engstler and Lubich develop a multirate extrapolation method where the base first order method is a particular implementation of extrapolated Euler method from Hairer and Osterman [11, 22, 23]. Engstler and Lubich argue that "Multirating is easy and natural with extrapolation methods" [11]. They use dense output methods in order to make their multirate extrapolation method more efficient [11, 22, 23]. The original  $T_{1,1}$ ,  $T_{1,1}$ , and  $T_{1,1}$  are assumed to be computed directly. In order to compute  $T_{3,1}$  for the next level, Engstler suggests two strategies. Either use polynomial extrapolation using previously computed function values for the unknown  $y_{dt_{f,i}}$ , or approximate Euler steps by using polynomial interpolation information only when the error estimate indicates that information is sufficiently accurate. They compare their results with a single-rate Runge-Kutta code based on an eighth-order Dormand-Prince [10, 11].

More recent methods of this type, including explicit fast integration and explicit slow integration (explicit/explicit), as well as implicit fast integration and explicit slow integration (implicit/explicit), have been explored by Constantinescu and Sandu [8, 9]. Their 2013 paper proposes an extrapolation method based on the basic first-order accurate *slowest-first* and *fastest-first* methods shown in equations (1.3)-(1.8) [9].

These multirate extrapolation method approaches increase the possible order of the resulting method at the cost of extra time step calculations. Constantinescu extorts readers to consider the "quality of solution that one should expect by using more efficient yet lower-order time-integration methods" [9]. As the desired order of accuracy increases, this type of extension increases in computational cost compared to non-extrapolation methods. Hence, these methods are reasonable for extrapolating up by one or two orders of accuracy but becomes cost-prohibitive when generating high-order methods unless complicated procedures are introduced to reduce the work.

### 1.2.2. Kværno-Rentrop Multirate Partitioned Runge-Kutta Methods

These methods improved on the extrapolation methods by reducing the computational cost of integration for the same order of accuracy. Although these methods are described in a

number of works [2, 13, 18, 33], the 1999 preprint which originally defines these methods gives perhaps the most straightforward definition [34]. They assume a partitioned formulation, with variables split into active and latent. Active components occur on a smaller timescale, and therefore are considered fast, while latent components are considered slow. Since this is a partitioned problem, they develop two families of multirate partitioned Runge-Kutta methods, MRKI and MRKII, and propose a specific second order multirate method MRK2(3) with a third order embedding.

Kværno and Rentrop's MRKI family of methods is built as a PRK using two Runge-Kutta methods with tableau

$$\begin{array}{c|c} c^{\{f\}} & A^{\{f\}} \\ \hline & b^{\{f\}}\tau \end{array} \quad \begin{array}{c|c} c^{\{s\}} & A^{\{s\}} \\ \hline & b^{\{s\}}\tau \end{array}$$

that have  $s^{\{f\}}$  and  $s^{\{s\}}$  stages respectively. For ease of notation, they use  $\lambda = 1, \dots, m$  to index the active steps within each latent step. These active steps consist of the fast stages  $\mathbf{k}_i^{\{f,\lambda\}}$  and the fast solutions  $\mathbf{y}_{\lambda+1}^{\{f\}}$ :

$$\begin{aligned} \mathbf{k}_i^{\{f,\lambda\}} &= f^{\{f\}} \left( t_n + \left( \lambda + c_i^{\{f\}} \right) \frac{h}{m}, \mathbf{y}_\lambda^{\{f\}} + \frac{h}{m} \sum_{j=1}^{s^{\{f\}}} a_{ij}^{\{f\}} \mathbf{k}_j^{\{f,\lambda\}}, \mathbf{Y}_i^{\{s,\lambda\}} \right), \\ & \quad i = 1, 2, \dots, s^{\{f\}}, \lambda = 1, \dots, m, \text{ and} \\ \mathbf{y}_{\lambda+1}^{\{f\}} &= \mathbf{y}_\lambda^{\{f\}} + \frac{h}{m} \sum_{i=1}^{s^{\{f\}}} b_i^{\{f\}} \mathbf{k}_i^{\{f,\lambda\}} \quad \lambda = 1, \dots, m. \end{aligned}$$

The latent stages and solution are given by

$$\begin{aligned} \mathbf{k}_i^{\{s\}} &= f^{\{s\}} \left( t_n + c_i^{\{s\}} h, \mathbf{Y}_i^{\{f\}}, \mathbf{y}_0^{\{s\}} + h \sum_{j=1}^{s^{\{s\}}} a_{ij}^{\{s\}} \mathbf{k}_j^{\{s\}} \right), \quad i = 1, 2, \dots, s^{\{s\}}, \\ \mathbf{y}_{n+1}^{\{s\}} &= \mathbf{y}_n^{\{s\}} + \frac{h}{m} \sum_{i=1}^{s^{\{s\}}} b_i^{\{s\}} \mathbf{k}_i^{\{s\}}. \end{aligned}$$

As with the previous methods, extrapolation or interpolation of solution data between the slow and fast variables is required, i.e.  $\mathbf{Y}_i^{\{s,\lambda\}} \approx \mathbf{y}^{\{s\}} \left( t_n + \left( \lambda + c_i^{\{f\}} \right) \frac{h}{m} \right)$  and  $\mathbf{Y}_i^{\{f\}} \approx \mathbf{y}^{\{f\}} \left( t_n + c_i^{\{s\}} h \right)$ . To this end, Kværno and Rentrop suggest the extrapolation

$$\begin{aligned} \mathbf{Y}_i^{\{\lambda,s\}} &= \mathbf{y}_n^{\{s\}} + \frac{h}{m} \sum_{j=1}^{s\{s\}} (\gamma_{ij} + \eta_j(\lambda)) \mathbf{k}_j^{\{s\}} \\ \mathbf{Y}_i^{\{f\}} &= \mathbf{y}_n^{\{f\}} + h \sum_{j=1}^{s\{f\}} (\bar{\gamma}_{ij}) \mathbf{k}_j^{\{f,0\}}. \end{aligned}$$

where  $\eta(\lambda)$  is a function from  $\mathbb{N} \rightarrow \mathbb{R}^{s\{s\}}$ , and where they define the coefficients

$$G = \begin{bmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \cdots & \gamma_{1,s\{s\}} \\ \gamma_{2,1} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \gamma_{i,j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \gamma_{s\{s\}-1,s\{s\}} \\ \gamma_{s\{s\},1} & \cdots & \cdots & \gamma_{s\{s\},s\{s\}-1} & \gamma_{s\{s\},s\{s\}} \end{bmatrix}, \quad \text{and}$$

$$\bar{G} = \begin{bmatrix} \bar{\gamma}_{1,1} & \bar{\gamma}_{1,2} & \cdots & \cdots & \bar{\gamma}_{1,s\{f\}} \\ \bar{\gamma}_{2,1} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \bar{\gamma}_{i,j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \bar{\gamma}_{s\{f\}-1,s\{f\}} \\ \bar{\gamma}_{s\{f\},1} & \cdots & \cdots & \bar{\gamma}_{s\{f\},s\{f\}-1} & \bar{\gamma}_{s\{f\},s\{f\}} \end{bmatrix}.$$

This formulation extrapolates the fast solution for computing the slow stages based on the first fast stage. It is used as the basis for the definition of the MRKI family of Multirate Partitioned Runge-Kutta methods [34]. In this work, Kværno and Rentrop derive order conditions for this family which relate the function  $\eta$  and coefficient matrices  $G$  and  $\bar{G}$  to the coefficients of the base methods. When the simplifying assumptions  $\sum_{j=1}^{\bar{s}} \eta_j(\lambda) =$

$\lambda \sum_{j=1}^{\bar{s}} \gamma_{i,j} = c_i$ ,  $i = 1, 2, \dots, s$  and  $\sum_{j=1}^{\bar{s}} \bar{\gamma}_{i,j} = \bar{c}_i$ ,  $i = 1, 2, \dots, \bar{s}$  are satisfied, and the base method is second order, then the multirate method MRKI will be second order. When these simplifying assumptions are satisfied, and the base method is third order, there are two additional conditions which must be fulfilled:

$$\sum_{i=1}^s \sum_{j=1}^{\bar{s}} b_i \gamma_{i,j} \bar{c}_j = \frac{1}{6m}$$

$$\sum_{i=1}^s \sum_{j=1}^{\bar{s}} b_i \eta_j(\lambda) \bar{c}_j = \frac{\lambda(\lambda+1)}{2m}$$

[34]. Günther, Kväerno and Rentrop's 2001 paper in BIT gives a similar definition, replacing the definition of the MRKI family with the definition of Multirate Partitioned Runge-Kutta 2(3) methods [18]. This newer definition simplifies the interpolation schemes between the fast and the slow base methods, and uses  $G = \bar{G}$ . It also makes the MPRK 2(3) method uniquely determined by the choice of time-scale separation  $m$ . We refer to the resulting scheme as the MPRK2(3) method; it uses the same explicit Runge-Kutta table for both the fast and slow base methods,

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 3/4 & 0 & 3/4 & 0 \\ \hline & 2/9 & 1/3 & 4/9 \end{array},$$

sets the nonzero interpolation coefficients as  $\gamma_{21} = \bar{\gamma}_{21} = \frac{1}{2}$ ,  $\gamma_{31} = \bar{\gamma}_{31} = \frac{3}{4}(1-m)$ , and  $\gamma_{32} = \bar{\gamma}_{32} = \frac{3}{4}m$ , and uses the interpolation function

$$\eta(\lambda) = \begin{bmatrix} \left(-\frac{1}{m} + \frac{3}{2} - \frac{m}{4}\right) \lambda - \frac{1}{2m} \lambda^2 \\ \left(\frac{1}{m} - \frac{3}{2} + \frac{3m}{4}\right) \lambda - \frac{1}{2m} \lambda^2 \\ \left(1 - \frac{m}{2}\right) \lambda + \frac{1}{m} \lambda^2 \end{bmatrix}. \quad (1.9)$$



### 1.2.3. Multirate Infinitesimal Step (MIS)

Multirate Infinitesimal Step methods are a general class of methods which were originally developed by Wensch, Knoth and Galant as a generalization of split-explicit methods from numerical weather prediction [27, 53, 57]. These methods also generalize the exponential integrators of Cellidoni, Marthinsen, and Owren [6]. Early split-explicit methods used a leapfrog scheme to advance from  $t - h$  to  $t + h$  with the slow variables taking two steps and the fast variables taking  $2m$  steps [27]. This is similar to the simplest multirate method mentioned in Section 1.1 [27]. Further methods were developed that took  $m/2$  fast steps between stages of a two-stage Runge-Kutta [53]. This early investigation contributed heavily to a stability theory relevant to problems similar to the compressible Euler equations [53, 58]. Wensch et. al took these well-studied methods and developed a systematic approach to the order conditions using PRK theory [57]. These methods continue to be developed at second and third order for a variety of target problem types in the context of numerical weather prediction and climate modeling [28, 29, 30, 49, 50, 57]. A representative problem for how these methods are applied to a particular problem is some simplification of the one-dimensional advection-diffusion-reaction equation as suggested by Schlegel, Knoth, Arnold and Wolke [49, 52]:

$$\frac{\partial}{\partial t} c = -\frac{\partial}{\partial x} (uc) + \frac{\partial}{\partial x} \left( \rho D \frac{\partial c}{\partial x} \right) + f(x, t, c) \quad (1.10)$$

Martin Schlegel’s 2012 dissertation describes in detail the Recursive Flux-Splitting Multirate (RFSMR) methods he developed based on Multirate Infinitesimal Step methods [49]. These RFSMR methods are named for the characteristic partitioning by fluxes, and account for the support of the different fluxes. Schlegel, Knoth, Arnold and Wolke demonstrated that these RFSMR methods improved parallel performance on the hierarchical grid scheme in COSMO-MUSCAT “developed at the Institute for Tropospheric Research in Leipzig” [51]. Our new methods will extend Multirate Infinitesimal Step theory alongside general multirate theory developed by Michael Günther and Adrian Sandu. To allow for more consistent comparisons

to our newly developed methods, the next chapter gives deeper background on how one step of a MIS or RFSMR method proceeds, and the theory we will leverage for order of accuracy and linear stability of multirate methods.

## Chapter 2

### Detailed Background Theory

In Chapter 1 we introduced context for existing multirate methods, especially focusing on those related to Runge-Kutta theory. In this chapter, we will introduce Generalized-Structure Additively-Partitioned Runge-Kutta (GARK) theory for describing a general family of Runge-Kutta methods, including theory on order of accuracy. Next, we will discuss how GARK theory is applied to MIS methods and how it is extended to develop a family of Multirate Generalized Additive Runge-Kutta schemes. Finally, we will discuss various stability theories as applied to multirate problems.

#### 2.1. Generalized-Structure Additively-Partitioned Runge Kutta Methods

In a 2013 technical report [42] and a subsequent journal article in 2015 [44], Adrian Sandu and Michael Günther proposed a new family of Runge-Kutta methods, named *Generalized-Structure Additively-Partitioned Runge Kutta* (GARK) methods. This GARK family of methods is an extension of some of the ideas put forth in the Kværno-Rentrop preprint from 1999 [34], as well as the PRK and ARK families of methods, previously described in Section 1.1. For this family of GARK methods, Sandu and Günther present order conditions and a linear stability function. In order to simplify the theory, GARK methods are formulated for an autonomous system of ODEs written in additively-split form,

$$\frac{dy}{dt} = f(y) = \sum_{q=1}^N f^{\{q\}}(y).$$

The coefficients that define a GARK method are written in an expanded Butcher tableau:

$$\begin{array}{c|c|c|c|c}
\mathbf{A}^{\{1,1\}} & \mathbf{A}^{\{1,2\}} & \dots & \mathbf{A}^{\{1,N-1\}} & \mathbf{A}^{\{1,N\}} \\
\hline
\mathbf{A}^{\{2,1\}} & \ddots & \ddots & \ddots & \mathbf{A}^{\{2,N\}} \\
\hline
\vdots & \ddots & \mathbf{A}^{\{i,j\}} & \ddots & \vdots \\
\hline
\mathbf{A}^{\{N-1,1\}} & \ddots & \ddots & \ddots & \mathbf{A}^{\{N-1,N\}} \\
\hline
\mathbf{A}^{\{N,1\}} & \mathbf{A}^{\{N,2\}} & \dots & \mathbf{A}^{\{N,N-1\}} & \mathbf{A}^{\{N,N\}} \\
\hline
\mathbf{b}^{\{1\}\top} & \mathbf{b}^{\{2\}\top} & \dots & \mathbf{b}^{\{N-1\}\top} & \mathbf{b}^{\{N\}\top}
\end{array} \tag{2.1}$$

The stages and the solution update are:

$$\begin{aligned}
\mathbf{k}^{\{q,i\}} &= \mathbf{y}_n + h \sum_{m=1}^N \sum_{j=1}^{s^{\{q\}}} a_{ij}^{\{q\}} f^{\{m\}}(\mathbf{k}^{\{m,j\}}), \\
&\quad i = 1, \dots, s^{\{q\}}, \quad q = 1, \dots, N, \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{q=1}^N \sum_{i=1}^{s^{\{q\}}} b_i^{\{q\}} f^{\{q\}}(\mathbf{k}^{\{q,i\}})
\end{aligned}$$

We note that the stage values and the solution update are similar to those in the PRK and ARK formulations. Similar to PRK, we have partitioned stage values. Similar to ARK, we use all the functions  $f^{\{q\}}$  from the additive partition of  $f(y)$  to update each stage as well as the solution.

As we alluded to in Section 1.1, the order conditions depend on the component methods  $\mathbf{A}^{\{i,i\}}$ , as well as coupling methods  $\mathbf{A}^{\{i,j\}}$  where  $i \neq j$ . As usual for Runge-Kutta methods, we make the simplifying assumption that  $c_j^{\{\sigma\}} = \sum_{l=1}^{s^{\{\sigma\}}} a_{j,l}^{\{\sigma,\nu\}} \forall \sigma, \nu = \{1, \dots, N\}$ . Sandu and Günther refer to these as the *internal consistency conditions* in their derivation of GARK order conditions [42, 20]. Their 2015 paper includes a proof using N-tree theory to derive the GARK order conditions for GARK tableau of arbitrary size [44]. Their 2013 technical report explicitly lists order conditions in elementwise form for up to 4th order [42], while the newer paper lists them in matrix-vector notation [44]. The newer paper includes a

Theorem regarding GARK order conditions for arbitrary orders based on Runge-Kutta order conditions [44]. Namely, Theorem 2.3 from Sandu and Günther’s 2015 paper reads, “The order conditions for a GARK method are obtained from the order conditions of ordinary Runge-Kutta methods. The usual labeling of the Runge-Kutta coefficients (subscripts  $i, j, k, \dots$ ) is accompanied by a corresponding labeling of the different partitions for the N-tree (superscripts  $\sigma, \nu, \mu, \dots$ )” [44]. We reproduce the summation form of these order condition below, and then use the internal consistency condition assumptions to reproduce the matrix-vector form of the GARK order conditions as well as a linear system representation. For simplicity of notation when discussing the matrix-vector form of the GARK order conditions, denote a vector of ones of length  $n$  as

$$\mathbb{1}_n = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{n \times 1}$$

and the  $i$ th unit vector as  $\mathbf{e}_i$ . Equations (2.2)-(2.9) are reproduced in the form shown in Sandu and Günther’s 2013 technical report [42]. These order conditions are determined by matching the terms from the Taylor series expansion of the error between the computed and exact solutions to the system of differential equations. They are essentially identical to the usual Runge-Kutta order conditions, however there are superscripts which account for all combinations of the internal GARK tables. In matrix-vector form, these usual Runge-Kutta order conditions up to order 4 are the following,

$$\begin{aligned} \mathbf{b}^\top \mathbb{1} &= 1, & \mathbf{b}^\top (\mathbf{c})^3 &= \frac{1}{4}, \\ \mathbf{b}^\top \mathbf{c} &= \frac{1}{2}, & (\mathbf{bc})^\top \mathbf{Ac} &= \frac{1}{8}, \\ \mathbf{b}^\top (\mathbf{c})^2 &= \frac{1}{3}, & \mathbf{b}^\top \mathbf{A} (\mathbf{c})^2 &= \frac{1}{12}, \\ \mathbf{b}^\top \mathbf{Ac} &= \frac{1}{6}, & \mathbf{b}^\top \mathbf{AAc} &= \frac{1}{24}. \end{aligned}$$

GARK first order conditions  $\forall \sigma = \{1, \dots, N\}$ :

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \mathbf{b}_i^{\{\sigma\}} = 1. \quad (2.2)$$

GARK second order conditions  $\forall \sigma, \nu = \{1, \dots, N\}$ :

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \sum_{j=1}^{\mathbf{s}\{\nu\}} \mathbf{b}_i^{\{\sigma\}} \mathbf{a}_{ij}^{\{\sigma, \nu\}} = \frac{1}{2}. \quad (2.3)$$

GARK third order conditions  $\forall \sigma, \nu, \mu = \{1, \dots, N\}$ :

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \sum_{j=1}^{\mathbf{s}\{\nu\}} \sum_{k=1}^{\mathbf{s}\{\mu\}} \mathbf{b}_i^{\{\sigma\}} \mathbf{a}_{ij}^{\{\sigma, \nu\}} \mathbf{a}_{ik}^{\{\sigma, \mu\}} = \frac{1}{3}, \quad (2.4)$$

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \sum_{j=1}^{\mathbf{s}\{\nu\}} \sum_{k=1}^{\mathbf{s}\{\mu\}} \mathbf{b}_i^{\{\sigma\}} \mathbf{a}_{ij}^{\{\sigma, \nu\}} \mathbf{a}_{jm}^{\{\nu, \mu\}} = \frac{1}{6}. \quad (2.5)$$

GARK fourth order conditions  $\forall \sigma, \nu, \lambda, \mu = \{1, \dots, N\}$ :

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \sum_{j=1}^{\mathbf{s}\{\nu\}} \sum_{l=1}^{\mathbf{s}\{\lambda\}} \sum_{m=1}^{\mathbf{s}\{\mu\}} \mathbf{b}_i^{\{\sigma\}} \mathbf{a}_{ij}^{\{\sigma, \nu\}} \mathbf{a}_{il}^{\{\sigma, \lambda\}} \mathbf{a}_{im}^{\{\sigma, \mu\}} = \frac{1}{4}, \quad (2.6)$$

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \sum_{j=1}^{\mathbf{s}\{\nu\}} \sum_{l=1}^{\mathbf{s}\{\lambda\}} \sum_{m=1}^{\mathbf{s}\{\mu\}} \mathbf{b}_i^{\{\sigma\}} \mathbf{a}_{ij}^{\{\sigma, \nu\}} \mathbf{a}_{jl}^{\{\nu, \lambda\}} \mathbf{a}_{im}^{\{\sigma, \mu\}} = \frac{1}{8}, \quad (2.7)$$

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \sum_{j=1}^{\mathbf{s}\{\nu\}} \sum_{l=1}^{\mathbf{s}\{\lambda\}} \sum_{m=1}^{\mathbf{s}\{\mu\}} \mathbf{b}_i^{\{\sigma\}} \mathbf{a}_{ij}^{\{\sigma, \nu\}} \mathbf{a}_{jl}^{\{\nu, \lambda\}} \mathbf{a}_{jm}^{\{\nu, \mu\}} = \frac{1}{12}, \quad (2.8)$$

$$\sum_{i=1}^{\mathbf{s}\{\sigma\}} \sum_{j=1}^{\mathbf{s}\{\nu\}} \sum_{l=1}^{\mathbf{s}\{\lambda\}} \sum_{m=1}^{\mathbf{s}\{\mu\}} \mathbf{b}_i^{\{\sigma\}} \mathbf{a}_{ij}^{\{\sigma, \nu\}} \mathbf{a}_{jl}^{\{\nu, \lambda\}} \mathbf{a}_{lm}^{\{\lambda, \mu\}} = \frac{1}{24}. \quad (2.9)$$

This matrix-vector form closely matches the form used in Theorem 2.6 of [44], and assumes a GARK method which has the same number of  $\mathbf{A}^{\{\sigma, \nu\}}$  rows and  $\mathbf{A}^{\{\sigma, \nu\}}$  columns. The matrix-vector form is described by Equations (2.10)-(2.17) for all  $\sigma, \nu, \mu = 1, \dots, N$ ,

$$\mathbf{b}^{\{\sigma\}\top} \mathbb{1} = 1, \quad (2.10)$$

$$\mathbf{b}^{\{\sigma\}\top} \mathbf{c}^{\{\sigma\}} = \frac{1}{2}, \quad (2.11)$$

$$\mathbf{b}^{\{\sigma\}\top} \left( \mathbf{c}^{\{\sigma\}} \right)^2 = \frac{1}{3}, \quad (2.12)$$

$$\mathbf{b}^{\{\sigma\}\top} \mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}} = \frac{1}{6}, \quad (2.13)$$

$$\mathbf{b}^{\{\sigma\}\top} \left( \mathbf{c}^{\{\sigma\}} \right)^3 = \frac{1}{4}, \quad (2.14)$$

$$\left( \mathbf{b}^{\{\sigma\}} \mathbf{c}^{\{\sigma\}} \right)^\top \mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}} = \frac{1}{8}, \quad (2.15)$$

$$\mathbf{b}^{\{\sigma\}\top} \mathbf{A}^{\{\sigma,\nu\}} \left( \mathbf{c}^{\{\nu\}} \right)^2 = \frac{1}{12}, \quad (2.16)$$

$$\mathbf{b}^{\{\sigma\}\top} \mathbf{A}^{\{\sigma,\mu\}} \mathbf{A}^{\{\mu,\nu\}} \mathbf{c}^{\{\nu\}} = \frac{1}{24}. \quad (2.17)$$

Later in this dissertation, we will use a linear system form of the order conditions which will be modeled after Equation (2.18). Equation (2.18) describes how the order conditions depend linearly on  $\mathbf{b}^{\{\sigma\}}$ , where each row is reproduced for all appropriate  $\sigma, \nu, \mu, \lambda = 1, \dots, N$ ,

$$\begin{bmatrix} 1 & \dots & 1 \\ c_1^{\{\sigma\}} & \dots & c_s^{\{\sigma\}} \\ (c_1^{\{\sigma\}})^2 & \dots & (c_s^{\{\sigma\}})^2 \\ [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_1 & \dots & [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_{s\{\sigma\}} \\ (c_1^{\{\sigma\}})^3 & \dots & (c_s^{\{\sigma\}})^3 \\ c_1^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_1 & \dots & c_s^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_{s\{\sigma\}} \\ \sum_{j=1}^{s\{\nu\}} \mathbf{A}_{1j}^{\{\sigma,\nu\}} (c_j^{\{\nu\}})^2 & \dots & \sum_{j=1}^{s\{\nu\}} \mathbf{A}_{s\{\sigma\}j}^{\{\sigma,\nu\}} (c_j^{\{\nu\}})^2 \\ [\mathbf{A}^{\{\sigma,\mu\}} \mathbf{A}^{\{\mu,\nu\}} \mathbf{c}^{\{\nu\}}]_1 & \dots & [\mathbf{A}^{\{\sigma,\mu\}} \mathbf{A}^{\{\mu,\nu\}} \mathbf{c}^{\{\nu\}}]_{s\{\sigma\}} \\ (c_1^{\{\sigma\}})^4 & \dots & (c_s^{\{\sigma\}})^4 \\ (c_1^{\{\sigma\}})^2 [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_1 & \dots & (c_s^{\{\sigma\}})^2 [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_{s\{\sigma\}} \\ \sum_{j=1}^{s\{\nu\}} c_1^{\{\sigma\}} \mathbf{A}_{1j}^{\{\sigma,\nu\}} (c_j^{\{\nu\}})^2 & \dots & \sum_{j=1}^{s\{\nu\}} c_s^{\{\sigma\}} \mathbf{A}_{s\{\sigma\}j}^{\{\sigma,\nu\}} (c_j^{\{\nu\}})^2 \\ c_1^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_1 & \dots & c_s^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_{s\{\sigma\}} \\ ([\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_1)^2 & \dots & ([\mathbf{A}^{\{\sigma,\nu\}} \mathbf{c}^{\{\nu\}}]_{s\{\sigma\}})^2 \\ \sum_{j=1}^{s\{\sigma\}} \mathbf{A}_{1j}^{\{\sigma,\nu\}} (c_j^{\{\nu\}})^3 & \dots & \sum_{j=1}^{s\{\sigma\}} \mathbf{A}_{s\{\sigma\}j}^{\{\sigma,\nu\}} (c_j^{\{\nu\}})^3 \\ \sum_{i=1}^{s\{\nu\}} \sum_{j=1}^{s\{\mu\}} \mathbf{A}_{1i}^{\{\sigma,\nu\}} c_i^{\{\nu\}} \mathbf{A}_{ij}^{\{\nu,\mu\}} (c_j^{\{\mu\}}) & \dots & \sum_{i=1}^{s\{\nu\}} \sum_{j=1}^{s\{\mu\}} \mathbf{A}_{s\{\sigma\}i}^{\{\sigma,\nu\}} c_i^{\{\nu\}} \mathbf{A}_{ij}^{\{\nu,\mu\}} (c_j^{\{\mu\}}) \\ \sum_{i=1}^{s\{\nu\}} \sum_{j=1}^{s\{\mu\}} \mathbf{A}_{1i}^{\{\sigma,\nu\}} \mathbf{A}_{ij}^{\{\nu,\mu\}} (c_j^{\{\mu\}})^2 & \dots & \sum_{i=1}^{s\{\nu\}} \sum_{j=1}^{s\{\mu\}} \mathbf{A}_{s\{\sigma\}i}^{\{\sigma,\nu\}} \mathbf{A}_{ij}^{\{\nu,\mu\}} (c_j^{\{\mu\}})^2 \\ [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{A}^{\{\nu,\mu\}} \mathbf{A}^{\{\mu,\lambda\}} \mathbf{c}^{\{\lambda\}}]_1 & \dots & [\mathbf{A}^{\{\sigma,\nu\}} \mathbf{A}^{\{\nu,\mu\}} \mathbf{A}^{\{\mu,\lambda\}} \mathbf{c}^{\{\lambda\}}]_{s\{\sigma\}} \end{bmatrix} \begin{bmatrix} b_1^{\{\sigma\}} \\ \vdots \\ \vdots \\ \vdots \\ b_{s\{\sigma\}}^{\{\sigma\}} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{6} \\ \frac{1}{4} \\ \frac{1}{8} \\ \frac{1}{12} \\ \frac{1}{24} \\ \frac{1}{5} \\ \frac{1}{10} \\ \frac{1}{15} \\ \frac{1}{30} \\ \frac{1}{20} \\ \frac{1}{20} \\ \frac{1}{40} \\ \frac{1}{60} \\ \frac{1}{120} \end{bmatrix}. \quad (2.18)$$

We will leverage GARK theory, including these order conditions, in this dissertation.

## 2.2. Multirate Generalized Additive Runge Kutta

In an extension to their GARK family of methods, Sandu and Günther proposed a *multirate* GARK (mGARK) family of methods [21], that provides a general basis for constructing multirate GARK methods. Their impetus behind constructing a general framework is that it allows multirate methods derived with potentially disparate contexts to be understood and analyzed using a single unified approach. The same paper includes a discussion on how to represent multirate methods which are based on Runge-Kutta methods that do not fit the mGARK framework as a  $2 \times 2$  GARK. The mGARK schemes require two base methods:

$$\begin{array}{c|c} c^{\{f\}} & A^{\{f\}} \\ \hline & b^{\{f\}}\tau \end{array} \quad \begin{array}{c|c} c^{\{s\}} & A^{\{s\}} \\ \hline & b^{\{s\}}\tau \end{array}$$

that have  $s^{\{f\}}$  and  $s^{\{s\}}$  stages respectively, and assumes  $f(\mathbf{y}) = f^{\{f\}}(\mathbf{y}) + f^{\{s\}}(\mathbf{y})$ .

The definition of *Multirate GARK method* defines "One macro-step of a generalized additive multirate Runge-Kutta method with  $m$  equal micro-steps"

$$\begin{aligned} \mathbf{k}_i^{\{s\}} &= \mathbf{y}_n + h \sum_{j=1}^{s^{\{s\}}} a_{i,j}^{\{s,s\}} f^{\{s\}}(\mathbf{k}_j^{\{s\}}) \\ &\quad \frac{h}{m} \sum_{\lambda=1}^m \sum_{j=1}^{s^{\{f\}}} a_{i,j}^{\{s,f,\lambda\}} f^{\{f\}}(\mathbf{k}_j^{\{f,\lambda\}}), \quad i = 1, \dots, s^{\{s\}} \\ \mathbf{k}_i^{\{f,\lambda\}} &= \mathbf{y}_n + \frac{h}{m} \sum_{l=1}^{\lambda-1} \sum_{j=1}^{s^{\{f\}}} b_j^{\{f\}} f^{\{f\}}(\mathbf{k}_j^{\{f,l\}}) + h \sum_{j=1}^{s^{\{s\}}} a_{i,j}^{\{f,s,\lambda\}} f^{\{s\}}(\mathbf{k}_j^{\{s\}}) \\ &\quad \frac{h}{m} \sum_{j=1}^{s^{\{f\}}} a_{i,j}^{\{f,f\}} f^{\{f\}}(\mathbf{k}_j^{\{f,\lambda\}}), \quad \lambda = 1, \dots, m, \quad i = 1, \dots, s^{\{f\}} \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{m} \sum_{\lambda=1}^m \sum_{i=1}^{s^{\{f\}}} b_i^{\{f\}} f^{\{f\}}(\mathbf{k}_i^{\{f,\lambda\}}) + h \sum_{i=1}^{s^{\{s\}}} b_i^{\{s\}} f^{\{s\}}(\mathbf{k}_i^{\{s\}})'' \end{aligned}$$



An mGARK method may be represented by the following choice of tableau:

$$\begin{array}{c|c}
\mathbf{A}^{\{f,f\}} & \mathbf{A}^{\{f,s\}} \\
\hline
\mathbf{A}^{\{s,f\}} & \mathbf{A}^{\{s,s\}} \\
\hline
\mathbf{b}^{\{f\}\top} & \mathbf{b}^{\{s\}\top}
\end{array}
=
\begin{array}{cccccc|c}
\frac{1}{m}A^{\{f,f\}} & 0 & \dots & \dots & 0 & & A^{\{f,s,1\}} \\
\mathbb{1}_{s\{f\}}b^{\{f\}\top} & \frac{1}{m}A^{\{f,f\}} & \ddots & 0 & \vdots & & A^{\{f,s,2\}} \\
\vdots & \ddots & \frac{1}{m}A^{\{f,f\}} & \ddots & \vdots & & \vdots \\
\vdots & \ddots & \ddots & \ddots & 0 & & \vdots \\
\vdots & \mathbb{1}_{s\{f\}}b^{\{f\}\top} & \dots & \dots & \mathbb{1}_{s\{f\}}b^{\{f\}\top} & \frac{1}{m}A^{\{f,f\}} & A^{\{f,s,m\}} \\
\hline
\frac{1}{m}A^{\{s,f,1\}} & \frac{1}{m}A^{\{s,f,2\}} & \frac{1}{m}A^{\{s,f,3\}} & \dots & \frac{1}{m}A^{\{s,f,m\}} & & A^{\{s,s\}} \\
\hline
\frac{1}{m}b^{\{f\}\top} & \frac{1}{m}b^{\{f\}\top} & \frac{1}{m}b^{\{f\}\top} & \dots & \frac{1}{m}b^{\{f\}\top} & & b^{\{s\}\top}
\end{array}$$

This tableau takes advantage in notation of the vector product,

$$\mathbb{1}_{s\{f\}}b^{\{f\}\top} = \begin{bmatrix} b^{\{f\}\top} \\ b^{\{f\}\top} \\ \vdots \\ b^{\{f\}\top} \end{bmatrix} \in \mathbb{R}^{s\{f\} \times s\{f\}}.$$

Each block row of  $\mathbf{A}^{\{f,f\}}$  can be considered one fast substep. The block lower triangular portion of  $\mathbf{A}^{\{f,f\}}$  which consists of blocks of  $\mathbb{1}_{s\{f\}}b^{\{f\}\top}$  for the mGARK can be considered as selecting the initial condition for each fast stage.  $\mathbf{A}^{\{f,s\}}$  contributes from the slow stage solutions  $\mathbf{k}^{\{s\}}$  to the fast stage solutions  $\mathbf{k}^{\{f\}}$ .  $\mathbf{A}^{\{s,f\}}$  contributes from the fast stage solutions  $\mathbf{k}^{\{f\}}$  to the slow stage solutions  $\mathbf{k}^{\{s\}}$ . Recall Sandu and Günther require that GARK methods satisfy the internal consistency conditions for the stage times, namely  $\mathbf{c}^{\{f\}} = \mathbf{A}^{\{f,f\}}\mathbb{1}_{s\{f\}} = \mathbf{A}^{\{f,s\}}\mathbb{1}_{s\{s\}}$  and  $\mathbf{c}^{\{s\}} = \mathbf{A}^{\{s,f\}}\mathbb{1}_{s\{f\}} = \mathbf{A}^{\{s,s\}}\mathbb{1}_{s\{s\}}$ . Equations (2.19) and (2.20) describe how these stage times are determined using the mGARK structure.

$$\frac{1}{m}A^{\{f,f\}}\mathbb{1}_{s\{f\}} + \frac{\lambda-1}{m}\mathbb{1}_{s\{f\}} = A^{\{f,s,\lambda\}}\mathbb{1}_{s\{s\}} = \mathbf{c}^{\{f,\lambda\}}, \quad \lambda = 1, \dots, m \quad (2.19)$$

$$\frac{1}{m}\sum_{\lambda=1}^m A^{\{s,f,\lambda\}}\mathbb{1}_{s\{f\}} = A^{\{s,s\}}\mathbb{1}_{s\{s\}} = \mathbf{c}^{\{s\}} \quad (2.20)$$

The mGARK’s autonomous formulation is assumed for theoretical purposes, however, the assumed use case is one where the mGARK method accounts for variation in time  $t$ , as stated by the concern that the stage times are “internally consistent” by enforcing the previously stated row-sum simplifying assumption [44]. For a mGARK to be second order overall, the base methods must each be second order and the second order coupling conditions must be satisfied. Similarly, for a mGARK to be third order overall, the base methods must each be third order and the second and third order coupling conditions must be satisfied. These order conditions follow directly from the GARK order conditions stated previously.

As shown in [21], this multirate GARK framework is very useful for analysing a wide variety of existing multirate methods, including those discussed in Chapter 1. However, the mGARK paper [21] returns to the GARK framework to describe Multirate Infinitesimal Step methods, rather than using the previously introduced mGARK framework. We will also use the GARK framework directly to describe our newly developed methods. Since the mGARK framework is built around base methods, it assumes that at most two different base methods are used. The theory we develop will allow for an arbitrary number of base methods, however in practice only one or two base methods will be used. This property of allowing for multiple base methods is one way to address dynamically changing time-scale separation within a GARK step. Another way to address this changing time-scale separation was investigated by Bremicker-Trubelhorn and Ortleb in their recent paper which develops theory on using adaptive micro-step sizes with mGARK methods [4]. For the mGARK, the weights  $b_j$  are used to construct the fast stage solutions and the final step solution the same way. In our new methods, we will consider lifting this restriction.

### 2.3. Creating Multirate Methods using GARK theory

Although Sandu and Günther only defined one possible structure for a multirate GARK in their paper [21], their paper [44] derived more general theoretical properties for GARKs, including order conditions. In their discussion of Multirate Infinitesimal Step methods they define the coefficient tables and the order conditions in terms of a  $2 \times 2$  GARK. We will discuss these methods in detail in the next section. In order to discuss MIS methods and the methods which we have developed based on the MIS methods consistently, we will introduce some notation alongside the relevant GARK theory. For clarity between our notation and our implementation, we will write

out the non-autonomous form of the stage and solution updates. Using the GARK framework and the associated theory, we consider arranging the following set of tableau in Butcher table form:

$$\mathbf{A}^{\{f,f\}} \in \mathbb{R}^{s^{\{f\}} \times s^{\{f\}}}, \mathbf{A}^{\{f,s\}} \in \mathbb{R}^{s^{\{f\}} \times s^{\{s\}}}, \mathbf{A}^{\{s,f\}} \in \mathbb{R}^{s^{\{s\}} \times s^{\{f\}}}, \text{ and } \mathbf{A}^{\{s,s\}} \in \mathbb{R}^{s^{\{s\}} \times s^{\{s\}}}.$$

$\mathbf{A}^{\{f,f\}}$	$\mathbf{A}^{\{f,s\}}$
$\mathbf{A}^{\{s,f\}}$	$\mathbf{A}^{\{s,s\}}$
$\mathbf{b}^{\{f\}\top}$	$\mathbf{b}^{\{s\}\top}$

The stage and solution updates based on these coefficients are defined as follows:

$$\begin{aligned} \mathbf{k}_j^{\{f\}} &= \mathbf{y}_n + h \sum_{l=1}^{s^{\{f\}}} a_{jl}^{\{f,f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + h \sum_{l=1}^{s^{\{s\}}} a_{jl}^{\{f,s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \\ \mathbf{k}_i^{\{s\}} &= \mathbf{y}_n + h \sum_{l=1}^{s^{\{f\}}} a_{il}^{\{s,f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + h \sum_{l=1}^{s^{\{s\}}} a_{il}^{\{s,s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{l=1}^{s^{\{f\}}} b_l^{\{f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + h \sum_{l=1}^{s^{\{s\}}} b_l^{\{s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \end{aligned}$$

where  $j = 1, \dots, s^{\{f\}}$  and  $i = 1, \dots, s^{\{s\}}$ .

The stages and solution updates when all stages are explicit, which generally follows from the base methods being explicit, are defined as follows:

$$\mathbf{k}_j^{\{f\}} = \mathbf{y}_n + h \sum_{l=1}^{j-1} a_{jl}^{\{f,f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + h \sum_{l=1}^{s^{\{s\}}} a_{jl}^{\{f,s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \quad (2.21)$$

$$\mathbf{k}_i^{\{s\}} = \mathbf{y}_n + h \sum_{l=1}^{s^{\{f\}}} a_{il}^{\{s,f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + h \sum_{l=1}^{i-1} a_{il}^{\{s,s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \quad (2.22)$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{l=1}^{s^{\{f\}}} b_l^{\{f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + h \sum_{l=1}^{s^{\{s\}}} b_l^{\{s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \quad (2.23)$$

where  $j = 1, \dots, s^{\{f\}}$  and  $i = 1, \dots, s^{\{s\}}$ . Multirate methods which use base methods commonly have intermediate solutions which accumulate portions of the fast stage solutions. In some cases, the first fast stage solution for each fast substep can be expressed as a linear combination of these accumulated solutions and the slow function values for previous stages. Specifically, when the

GARK coefficients  $a_{jl}^{\{f,f\}} = a_{pl}^{\{f,f\}}$ , we can take advantage of this structure to reduce the number of vectors the size of  $y$  which must be stored during each stage computation. We will discuss notation for these intermediate and partial solutions for specific examples and implementations later, in Section 3.3.

Given this specific notation, we can use GARK theory to enumerate the order conditions for a GARK of this size. As we alluded to in Section 2.1, the order conditions for full or overall order depend on the classical order conditions, as well as coupling conditions, and internal consistency conditions. For the methods we construct, we will rely on the GARK order conditions in matrix-vector form as in Equations (2.10)-(2.17), as well as in linear system form as in Equation (2.18). The classical conditions concern how the method works when either function from the splitting is removed, i.e. the classical conditions correspond to the standard order conditions for the RK method  $\mathbf{A}^{\{f,f\}}$ ,  $\mathbf{b}^{\{f\}\tau}$ , and  $\mathbf{c}^{\{f\}}$ , and similarly for the slow set of tableau. The coupling conditions involve all the other possible combinations of fast and slow, such as  $\mathbf{b}^{\{f\}\tau} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}} = \frac{1}{6}$ . Sandu and Günther make a distinction between fast order conditions and slow order conditions based on whether the order condition uses  $\mathbf{b}^{\{f\}}$  or  $\mathbf{b}^{\{s\}}$ .

The fast order conditions up to fourth order are as follows:

$$\mathbf{b}^{\{f\}\top} \mathbb{1} = 1 \quad (2.24)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{c}^{\{f\}} = \frac{1}{2} \quad (2.25)$$

$$\mathbf{b}^{\{f\}\top} \left( \mathbf{c}^{\{f\}} \right)^2 = \frac{1}{3} \quad (2.26)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}} = \frac{1}{6} \quad (2.27)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}} = \frac{1}{6} \quad (2.28)$$

$$\mathbf{b}^{\{f\}\top} \left( \mathbf{c}^{\{f\}} \right)^3 = \frac{1}{4} \quad (2.29)$$

$$\left( \mathbf{b}^{\{f\}\top} \mathbf{c}^{\{f\}} \right) \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}} = \frac{1}{8} \quad (2.30)$$

$$\left( \mathbf{b}^{\{f\}\top} \mathbf{c}^{\{f\}} \right) \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}} = \frac{1}{8} \quad (2.31)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,f\}} \left( \mathbf{c}^{\{f\}} \right)^2 = \frac{1}{12} \quad (2.32)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,s\}} \left( \mathbf{c}^{\{s\}} \right)^2 = \frac{1}{12} \quad (2.33)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}} = \frac{1}{24} \quad (2.34)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}} = \frac{1}{24} \quad (2.35)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}} = \frac{1}{24} \quad (2.36)$$

$$\mathbf{b}^{\{f\}\top} \mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}} = \frac{1}{24} \quad (2.37)$$

The slow order conditions up to order four are as follows:

$$\mathbf{b}^{\{s\}\top} \mathbf{1} = 1 \quad (2.38)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{c}^{\{s\}} = \frac{1}{2} \quad (2.39)$$

$$\mathbf{b}^{\{s\}\top} \left( \mathbf{c}^{\{s\}} \right)^2 = \frac{1}{3} \quad (2.40)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}} = \frac{1}{6} \quad (2.41)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}} = \frac{1}{6} \quad (2.42)$$

$$\mathbf{b}^{\{s\}\top} \left( \mathbf{c}^{\{s\}} \right)^3 = \frac{1}{4} \quad (2.43)$$

$$\left( \mathbf{b}^{\{s\}\top} \mathbf{c}^{\{s\}} \right) \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}} = \frac{1}{8} \quad (2.44)$$

$$\left( \mathbf{b}^{\{s\}\top} \mathbf{c}^{\{s\}} \right) \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}} = \frac{1}{8} \quad (2.45)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,s\}} \left( \mathbf{c}^{\{s\}} \right)^2 = \frac{1}{12} \quad (2.46)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,f\}} \left( \mathbf{c}^{\{f\}} \right)^2 = \frac{1}{12} \quad (2.47)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}} = \frac{1}{24} \quad (2.48)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}} = \frac{1}{24} \quad (2.49)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}} = \frac{1}{24} \quad (2.50)$$

$$\mathbf{b}^{\{s\}\top} \mathbf{A}^{\{s,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}} = \frac{1}{24} \quad (2.51)$$

Equation 2.52 expresses these in linear system form, where  $\sigma = s, \nu = f$  or  $\sigma = f, \nu = s$  for the slow or fast order conditions respectively as a specification of Equation 2.18.

$$\begin{bmatrix}
 1 & \cdots & 1 \\
 c_1^{\{\sigma\}} & \cdots & c_{\nu\{\sigma\}}^{\{\sigma\}} \\
 \left(c_1^{\{\sigma\}}\right)^2 & \cdots & \left(c_{\nu\{\sigma\}}^{\{\sigma\}}\right)^2 \\
 [\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{c}^{\{\sigma\}}]_1 & \cdots & [\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{c}^{\{\sigma\}}]_{\nu\{\sigma\}} \\
 [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{c}^{\{\nu\}}]_1 & \cdots & [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{c}^{\{\nu\}}]_{\nu\{\sigma\}} \\
 \left(c_1^{\{\sigma\}}\right)^3 & \cdots & \left(c_{\nu\{\sigma\}}^{\{\sigma\}}\right)^3 \\
 c_1^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{c}^{\{\sigma\}}]_1 & \cdots & c_{\nu\{\sigma\}}^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{c}^{\{\sigma\}}]_{\nu\{\sigma\}} \\
 c_1^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{c}^{\{\nu\}}]_1 & \cdots & c_{\nu\{\sigma\}}^{\{\sigma\}} [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{c}^{\{\nu\}}]_{\nu\{\sigma\}} \\
 \sum_{j=1}^{\nu\{\sigma\}} \mathbf{A}_{1j}^{\{\sigma,\sigma\}} \left(c_j^{\{\sigma\}}\right)^2 & \cdots & \sum_{j=1}^{\nu\{\sigma\}} \mathbf{A}_{\nu\{\sigma\}j}^{\{\sigma,\sigma\}} \left(c_j^{\{\sigma\}}\right)^2 \\
 \sum_{j=1}^{\nu\{\nu\}} \mathbf{A}_{1j}^{\{\sigma,\nu\}} \left(c_j^{\{\nu\}}\right)^2 & \cdots & \sum_{j=1}^{\nu\{\nu\}} \mathbf{A}_{\nu\{\sigma\}j}^{\{\sigma,\nu\}} \left(c_j^{\{\nu\}}\right)^2 \\
 [\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{c}^{\{\sigma\}}]_1 & \cdots & [\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{c}^{\{\sigma\}}]_{\nu\{\sigma\}} \\
 [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{A}^{\{\nu,\sigma\}}\mathbf{c}^{\{\sigma\}}]_1 & \cdots & [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{A}^{\{\nu,\sigma\}}\mathbf{c}^{\{\sigma\}}]_{\nu\{\sigma\}} \\
 [\mathbf{A}^{\{\sigma,\sigma\}}\mathbf{A}^{\{\sigma,\nu\}}\mathbf{c}^{\{\nu\}}]_1 & \cdots & [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{A}^{\{\nu,\sigma\}}\mathbf{c}^{\{\sigma\}}]_{\nu\{\sigma\}} \\
 [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{A}^{\{\nu,\nu\}}\mathbf{c}^{\{\nu\}}]_1 & \cdots & [\mathbf{A}^{\{\sigma,\nu\}}\mathbf{A}^{\{\nu,\nu\}}\mathbf{c}^{\{\nu\}}]_{\nu\{\sigma\}}
 \end{bmatrix}
 \begin{bmatrix}
 b_1^{\{\sigma\}} \\
 \vdots \\
 \vdots \\
 \vdots \\
 b_{\nu\{\sigma\}}^{\{\sigma\}}
 \end{bmatrix}
 =
 \begin{bmatrix}
 1 \\
 \frac{1}{2} \\
 \frac{1}{3} \\
 \frac{1}{6} \\
 \frac{1}{6} \\
 \frac{1}{4} \\
 \frac{1}{8} \\
 \frac{1}{8} \\
 \frac{1}{12} \\
 \frac{1}{12} \\
 \frac{1}{24} \\
 \frac{1}{24} \\
 \frac{1}{24} \\
 \frac{1}{24}
 \end{bmatrix}
 \quad (2.52)$$

Specifically, for the fast order conditions as a linear system we have the following,

$$\begin{bmatrix}
1 & \dots & 1 \\
c_1^{\{f\}} & \dots & c_{s\{f\}}^{\{f\}} \\
\left(c_1^{\{f\}}\right)^2 & \dots & \left(c_{s\{f\}}^{\{f\}}\right)^2 \\
[\mathbf{A}^{\{f,f\}}\mathbf{c}^{\{f\}}]_1 & \dots & [\mathbf{A}^{\{f,f\}}\mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,s\}}\mathbf{c}^{\{s\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}}\mathbf{c}^{\{s\}}]_{s\{f\}} \\
\left(c_1^{\{f\}}\right)^3 & \dots & \left(c_{s\{f\}}^{\{f\}}\right)^3 \\
c_1^{\{f\}} [\mathbf{A}^{\{f,f\}}\mathbf{c}^{\{f\}}]_1 & \dots & c_{s\{f\}}^{\{f\}} [\mathbf{A}^{\{f,f\}}\mathbf{c}^{\{f\}}]_{s\{f\}} \\
c_1^{\{f\}} [\mathbf{A}^{\{f,s\}}\mathbf{c}^{\{s\}}]_1 & \dots & c_{s\{f\}}^{\{f\}} [\mathbf{A}^{\{f,s\}}\mathbf{c}^{\{s\}}]_{s\{f\}} \\
\sum_{j=1}^{s\{f\}} \mathbf{A}_{1j}^{\{f,f\}} \left(c_j^{\{f\}}\right)^2 & \dots & \sum_{j=1}^{s\{f\}} \mathbf{A}_{s\{f\}j}^{\{f,f\}} \left(c_j^{\{f\}}\right)^2 \\
\sum_{j=1}^{s\{s\}} \mathbf{A}_{1j}^{\{f,s\}} \left(c_j^{\{s\}}\right)^2 & \dots & \sum_{j=1}^{s\{s\}} \mathbf{A}_{s\{f\}j}^{\{f,s\}} \left(c_j^{\{s\}}\right)^2 \\
[\mathbf{A}^{\{f,f\}}\mathbf{A}^{\{f,f\}}\mathbf{c}^{\{f\}}]_1 & \dots & [\mathbf{A}^{\{f,f\}}\mathbf{A}^{\{f,f\}}\mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,s\}}\mathbf{A}^{\{s,f\}}\mathbf{c}^{\{f\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}}\mathbf{A}^{\{s,f\}}\mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,f\}}\mathbf{A}^{\{f,s\}}\mathbf{c}^{\{s\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}}\mathbf{A}^{\{s,f\}}\mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,s\}}\mathbf{A}^{\{s,s\}}\mathbf{c}^{\{s\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}}\mathbf{A}^{\{s,s\}}\mathbf{c}^{\{s\}}]_{s\{f\}}
\end{bmatrix}
\begin{bmatrix}
\hat{b}_1^{\{f\}} \\
\vdots \\
\vdots \\
\vdots \\
\hat{b}_{s\{f\}}^{\{f\}}
\end{bmatrix}
=
\begin{bmatrix}
1 \\
\frac{1}{2} \\
\frac{1}{3} \\
\frac{1}{6} \\
\frac{1}{6} \\
\frac{1}{4} \\
\frac{1}{8} \\
\frac{1}{8} \\
\frac{1}{12} \\
\frac{1}{12} \\
\frac{1}{24} \\
\frac{1}{24} \\
\frac{1}{24} \\
\frac{1}{24}
\end{bmatrix}
\quad (2.53)$$



Note that adding the fifth order conditions, which follow from Theorem 2.3 in Sandu and Günther's 2015 paper, increases the total number of fast order conditions from 14 to 53 [44],

$$\left[ \begin{array}{ccc}
(c_1^{\{f\}})^4 & \dots & (c_{s\{f\}}^{\{f\}})^4 \\
(c_1^{\{f\}})^2 [\mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_1 & \dots & (c_{s\{f\}}^{\{f\}})^2 [\mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}} \\
(c_1^{\{f\}})^2 [\mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_1 & \dots & (c_{s\{f\}}^{\{f\}})^2 [\mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}} \\
\sum_{j=1}^{s\{f\}} c_1^{\{f\}} \mathbf{A}_{1j}^{\{f,f\}} (c_j^{\{f\}})^2 & \dots & \sum_{j=1}^{s\{f\}} c_{s\{f\}}^{\{f\}} \mathbf{A}_{s\{f\}j}^{\{f,f\}} (c_j^{\{f\}})^2 \\
\sum_{j=1}^{s\{s\}} c_1^{\{s\}} \mathbf{A}_{1j}^{\{f,s\}} (c_j^{\{s\}})^2 & \dots & \sum_{j=1}^{s\{s\}} c_{s\{f\}}^{\{s\}} \mathbf{A}_{s\{f\}j}^{\{f,s\}} (c_j^{\{s\}})^2 \\
c_1^{\{f\}} [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_1 & \dots & c_{s\{f\}}^{\{f\}} [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}} \\
c_1^{\{f\}} [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}}]_1 & \dots & c_{s\{f\}}^{\{f\}} [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}} \\
c_1^{\{f\}} [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_1 & \dots & c_{s\{f\}}^{\{f\}} [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}} \\
c_1^{\{f\}} [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}}]_1 & \dots & c_{s\{f\}}^{\{f\}} [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}} \\
([\mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_1)^2 & \dots & ([\mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}})^2 \\
([\mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_1)^2 & \dots & ([\mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}})^2 \\
\sum_{j=1}^{s\{f\}} \mathbf{A}_{1j}^{\{f,f\}} (c_j^{\{f\}})^3 & \dots & \sum_{j=1}^{s\{f\}} \mathbf{A}_{s\{f\}j}^{\{f,f\}} (c_j^{\{f\}})^3 \\
\sum_{j=1}^{s\{f\}} \mathbf{A}_{1j}^{\{f,s\}} (c_j^{\{s\}})^3 & \dots & \sum_{j=1}^{s\{f\}} \mathbf{A}_{s\{f\}j}^{\{f,s\}} (c_j^{\{s\}})^3 \\
\sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{1i}^{\{f,f\}} c_i^{\{f\}} \mathbf{A}_{ij}^{\{f,f\}} (c_j^{\{f\}}) & \dots & \sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{s\{f\}i}^{\{f,f\}} c_i^{\{f\}} \mathbf{A}_{ij}^{\{f,f\}} (c_j^{\{f\}}) \\
\sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{1i}^{\{f,f\}} c_i^{\{f\}} \mathbf{A}_{ij}^{\{f,s\}} (c_j^{\{s\}}) & \dots & \sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{s\{f\}i}^{\{f,f\}} c_i^{\{f\}} \mathbf{A}_{ij}^{\{f,s\}} (c_j^{\{s\}}) \\
\sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{1i}^{\{f,s\}} c_i^{\{s\}} \mathbf{A}_{ij}^{\{s,f\}} (c_j^{\{f\}}) & \dots & \sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{s\{f\}i}^{\{f,s\}} c_i^{\{s\}} \mathbf{A}_{ij}^{\{s,f\}} (c_j^{\{f\}}) \\
\sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{1i}^{\{f,s\}} c_i^{\{s\}} \mathbf{A}_{ij}^{\{s,s\}} (c_j^{\{s\}}) & \dots & \sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{s\{f\}i}^{\{f,s\}} c_i^{\{s\}} \mathbf{A}_{ij}^{\{s,s\}} (c_j^{\{s\}}) \\
\sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{1i}^{\{f,f\}} \mathbf{A}_{ij}^{\{f,f\}} (c_j^{\{f\}})^2 & \dots & \sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{s\{f\}i}^{\{f,f\}} \mathbf{A}_{ij}^{\{f,f\}} (c_j^{\{f\}})^2 \\
\sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{1i}^{\{f,f\}} \mathbf{A}_{ij}^{\{f,s\}} (c_j^{\{s\}})^2 & \dots & \sum_{i=1}^{s\{f\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{s\{f\}i}^{\{f,f\}} \mathbf{A}_{ij}^{\{f,s\}} (c_j^{\{s\}})^2 \\
\sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{1i}^{\{f,s\}} \mathbf{A}_{ij}^{\{s,f\}} (c_j^{\{f\}})^2 & \dots & \sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{f\}} \mathbf{A}_{s\{f\}i}^{\{f,s\}} \mathbf{A}_{ij}^{\{s,f\}} (c_j^{\{f\}})^2 \\
\sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{1i}^{\{f,s\}} \mathbf{A}_{ij}^{\{s,s\}} (c_j^{\{s\}})^2 & \dots & \sum_{i=1}^{s\{s\}} \sum_{j=1}^{s\{s\}} \mathbf{A}_{s\{f\}i}^{\{f,s\}} \mathbf{A}_{ij}^{\{s,s\}} (c_j^{\{s\}})^2 \\
[\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_1 & \dots & [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}}]_1 & \dots & [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_1 & \dots & [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}}]_1 & \dots & [\mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{A}^{\{f,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}} \\
[\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}}]_1 & \dots & [\mathbf{A}^{\{f,s\}} \mathbf{A}^{\{s,s\}} \mathbf{A}^{\{s,s\}} \mathbf{c}^{\{s\}}]_{s\{f\}}
\end{array} \right] = \begin{array}{c} \frac{1}{5} \\ \frac{1}{10} \\ \frac{1}{10} \\ \frac{1}{15} \\ \frac{1}{15} \\ \frac{1}{30} \\ \frac{1}{30} \\ \frac{1}{30} \\ \frac{1}{30} \\ \frac{1}{20} \\ \frac{1}{20} \\ \frac{1}{20} \\ \frac{1}{20} \\ \frac{1}{40} \\ \frac{1}{40} \\ \frac{1}{40} \\ \frac{1}{40} \\ \frac{1}{60} \\ \frac{1}{60} \\ \frac{1}{60} \\ \frac{1}{60} \\ \frac{1}{120} \\ \frac{1}{120} \\ \frac{1}{120} \\ \frac{1}{120} \\ \frac{1}{120} \\ \frac{1}{120} \\ \frac{1}{120} \\ \frac{1}{120} \\ \frac{1}{120} \end{array} \quad (2.54)$$

### 2.3.1. More Flexible Structure for New Methods

To give us a consistent notation to compare multirate methods based on a 2 by 2 GARK table, we consider using a block structure where each block-row represents one fast sub-step. We denote block structure with superscripts, i.e.  $[\mathbf{A}^{\{f,f\}}]_{i,j} = \mathbf{A}^{\{f,f,i,j\}}$ ,  $[\mathbf{b}^{\{f\}}]_j = \mathbf{b}^{\{f,j\}}$ , and specific elements with subscripts, i.e.  $[\mathbf{A}^{\{f,f,i,j\}}]_{k,l} = a_{k,l}^{\{f,f,i,j\}}$  and  $[\mathbf{b}^{\{f,j\}}]_l = b_l^{\{f,j\}}$ . This yields the following table:

$$\begin{array}{c|c}
 \mathbf{A}^{\{f,f\}} & \mathbf{A}^{\{f,s\}} \\
 \hline
 \mathbf{A}^{\{s,f\}} & \mathbf{A}^{\{s,s\}} \\
 \hline
 \mathbf{b}^{\{f\}\tau} & \mathbf{b}^{\{s\}\tau}
 \end{array} = \begin{array}{cccccc|c}
 A^{\{f,f,1,1\}} & 0 & \dots & \dots & 0 & A^{\{f,s,1\}} \\
 A^{\{f,f,2,1\}} & A^{\{f,f,2,2\}} & \ddots & 0 & \vdots & A^{\{f,s,2\}} \\
 A^{\{f,f,3,1\}} & \ddots & A^{\{f,f,3,3\}} & \ddots & \vdots & \vdots \\
 \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\
 \vdots & & & & & \\
 A^{\{f,f,m,1\}} & \dots & \dots & A^{\{f,f,m,m-2\}} & A^{\{f,f,m,m-1\}} & A^{\{f,f,m,m\}} & A^{\{f,s,m\}} \\
 \hline
 A^{\{s,f,1\}} & A^{\{s,f,2\}} & A^{\{s,f,3\}} & \dots & A^{\{s,f,m\}} & & A^{\{s,s\}} \\
 \hline
 b^{\{f,1\}\tau} & b^{\{f,2\}\tau} & b^{\{f,3\}\tau} & \dots & b^{\{f,m\}\tau} & & b^{\{s\}\tau}
 \end{array} \tag{2.55}$$

The blue portions of this GARK table indicates changes from the mGARK formulation. Each block row from  $\mathbf{A}^{\{f,f\}}$  and  $\mathbf{A}^{\{f,s\}}$  may be considered as the fast stage updates for a fast micro-step. The strictly lower triangular portion of  $\mathbf{A}^{\{f,f\}}$  may be considered as the initial conditions for each block row, based upon previous fast stage solution values. This more flexible notation addresses the situation where the fast base method changes between fast sub-steps. It also gives the option of using a different initial condition for the fast sub-steps. The added flexibility with the  $b^{\{f,j\}}$  assists in developing optimized methods and addressing new embedding and solution techniques in a general fashion.

## 2.4. MIS methods as an extension of GARK theory

Although MIS methods are discussed in Sandu and Günther’s paper which develops mGARK methods, Sandu and Günther use GARK theory, rather than the newly defined mGARK theory, to analyze these methods [21]. The methods developed in this dissertation will follow closely from the pairing of GARK theory with MIS methods. For the purpose of detailed examination, we will consider MIS methods as described by Sandu and Günther in the GARK context [21]. Sandu and Günther specifically present MIS methods in a GARK context in Theorem 4 of their 2013 paper in *Numerische Mathematik* on multirate GARK methods, summarized as “the MIS scheme is a particular instance of a GARK method” [20]. In this formulation, the additive two-rate problem formulation is appropriate. The problem is stated as being autonomous, i.e. the ODE right-hand side function depends only on the solution,  $f(\mathbf{y})$ , rather than  $f(t, \mathbf{y})$ , in order to simplify the theory and the algorithm. This assumption is relaxed in a recent paper by Knoth wherein a way to apply a Multirate Infinitesimal Step method to a non-autonomous problem is described [29]. Similar to the mGARK formulation, MIS methods in the GARK context are constructed using Runge-Kutta methods for the base methods:

$$\begin{array}{c|c}
 \text{Fast or Inner Base Method} & \text{Slow or Outer Base Method} \\
 \hline
 \frac{c^{\{f\}}}{b^{\{f\}\tau} \Big| \frac{A^{\{f\}}}{b^{I\tau}} = \frac{c^I}{b^{I\tau}} \Big| A^I & \frac{c^{\{s\}}}{b^{\{s\}\tau} \Big| \frac{A^{\{s\}}}{b^{O\tau}} = \frac{c^O}{b^{O\tau}} \Big| A^O
 \end{array}$$

Unless otherwise specified, MIS methods typically assume the same inner and outer base methods, i.e.  $c^I = c^O$ ,  $A^I = A^O$  and  $b^{I\tau} = b^{O\tau}$ .

The coefficients that define a MIS method are written in an expanded Butcher tableau:

$$\begin{aligned}
\mathbf{A}^{\{f,f\}} &= \begin{bmatrix} c_2^O A^I & 0 & \cdots & 0 \\ c_2^O \mathbb{1} \mathbf{b}^{I\top} & (c_3^O - c_2^O) A^I & \ddots & \vdots \\ \vdots & (c_3^O - c_2^O) \mathbb{1} \mathbf{b}^{I\top} & \ddots & 0 \\ & \vdots & & \\ c_2^O \mathbb{1} \mathbf{b}^{I\top} & (c_3^O - c_2^O) \mathbb{1} \mathbf{b}^{I\top} & \cdots & (1 - c_{s^O}^O) A^I \end{bmatrix} \\
\mathbf{A}^{\{s,f\}} &= \begin{bmatrix} \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{0} \\ c_2^O \mathbf{b}^{I\top} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ c_2^O \mathbf{b}^{I\top} & (c_3^O - c_2^O) \mathbf{b}^{I\top} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \mathbf{0} & \mathbf{0} \\ c_2^O \mathbf{b}^{I\top} & (c_3^O - c_2^O) \mathbf{b}^{I\top} & \cdots & (c_{s^O}^O - c_{s^O-1}^O) \mathbf{b}^{I\top} & \mathbf{0} \end{bmatrix} \\
\mathbf{A}^{\{f,s\}} &= \begin{bmatrix} \mathbf{c}^I (\mathbf{e}_2^\top) A^O \\ \mathbb{1} \mathbf{e}_2^\top A^O + \mathbf{c}^I (\mathbf{e}_3^\top - \mathbf{e}_2^\top) A^O \\ \mathbb{1} \mathbf{e}_3^\top A^O + \mathbf{c}^I (\mathbf{e}_4^\top - \mathbf{e}_3^\top) A^O \\ \mathbb{1} \mathbf{e}_4^\top A^O + \mathbf{c}^I (b^{O\top} - \mathbf{e}_4^\top) A^O \end{bmatrix} \\
\mathbf{A}^{\{s,s\}} &= A^O \\
\mathbf{c}^{\{f\}} &= \begin{bmatrix} c_2^O \mathbf{c}^I \\ c_2^O \mathbb{1} + (c_3^O - c_2^O) \mathbf{c}^I \\ c_3^O \mathbb{1} + (c_4^O - c_3^O) \mathbf{c}^I \\ c_4^O \mathbb{1} + (1 - c_4^O) \mathbf{c}^I \end{bmatrix} \\
\mathbf{c}^{\{s\}} &= c^O \\
\mathbf{b}^{\{f\}} &= \begin{bmatrix} c_2^O \mathbf{b}^I & (c_3^O - c_2^O) \mathbf{b}^I & \cdots & (c_{s^O}^O - c_{s^O-1}^O) \mathbf{b}^I & (1 - c_{s^O}^O) \mathbf{b}^I \end{bmatrix} \\
\mathbf{b}^{\{s\}} &= \mathbf{b}^O
\end{aligned}$$

These coefficients can be represented elementwise as follows:

$$\begin{aligned}
a_{i,j}^{\{s,s\}} &= a_{i,j}^O, \quad j = 1, \dots, i-1, \\
a_{i,p}^{\{s,f,l\}} &= b_p^I (c_{l+1}^O - c_l^O), \quad l = 1, \dots, s^O - 1, \quad i = l+1, \dots, s^O, \quad p = 1, \dots, s^I, \\
a_{k,j}^{\{f,s,i\}} &= a_{i,j}^O + c_k^I (a_{i+1,j}^O - a_{i,j}^O), \quad i = 1, \dots, s^O - 1, \quad j = 1, \dots, i, \quad k = 1, \dots, s^I, \\
a_{k,p}^{\{f,f,i,l\}} &= \begin{cases} b_p^I (c_{l+1}^O - c_l^O) & l = 1, \dots, i-1 \\ a_{k,p}^I (c_{i+1}^O - c_i^O) & l = i, \end{cases} \quad i = 1, \dots, s^O - 1, \quad p, k = 1, \dots, s^I, \\
b_p^{\{f,i\}} &= b_p^I (c_{i+1}^O - c_i^O), \quad i = 1, \dots, s^O - 1, \\
b_p^{\{s\}} &= b_p^O, \quad p = 1, \dots, s^I.
\end{aligned}$$

The final fast sub-step defines the last block-row of the  $\mathbf{A}^{\{f,f\}}$  and  $\mathbf{A}^{\{f,s\}}$  tables, as well as the final step solution update.

$$\begin{aligned}
a_{k,j}^{\{f,s,s^O\}} &= a_{s^O,j}^O + c_k^I (b_j^O - a_{s^O,j}^O), \quad j = 1, \dots, s^O - 1, \quad k = 1, \dots, s^I, \\
a_{k,m}^{\{f,f,s^O,l\}} &= \begin{cases} b_p^I (c_{l+1}^O - c_l^O) & l = 1, \dots, s^O - 1 \\ a_{k,p}^I (1 - c_{s^O}^O) & l = s^O, \end{cases} \quad p, k = 1, \dots, s^I, \\
b_p^{\{f,s^O\}} &= b_p^I (1 - c_{s^O}^O), \quad p = 1, \dots, s^I, \\
b_p^{\{s\}} &= b_p^O, \quad p = 1, \dots, s^I.
\end{aligned}$$

Augmenting the Butcher tableau of the base methods,

$$\begin{array}{c|c} \mathbf{c}^O & A^O \\ \hline & \mathbf{b}^{O\top} \end{array} \rightarrow \begin{array}{c|c} \mathbf{c}^O & A^O \quad \mathbf{0} \\ \hline 1 & \mathbf{b}^{O\top} \quad 0 \\ \hline & \mathbf{b}^{O\top} \quad 0 \end{array}$$

we are able to write the MIS method in the GARK formulation more concisely, by considering  $i = 1, \dots, s^O + 1$  for  $A$ 's and  $c$ 's,

$$\begin{aligned}
a_{i,j}^{\{s,s\}} &= a_{i,j}^O, \quad j = 1, \dots, i - 1, \\
a_{i,p}^{\{s,f,l\}} &= b_p^I (c_{l+1}^O - c_l^O), \quad l = 1, \dots, s^O + 1, \quad i = l + 1, \dots, s^O, \quad p = 1, \dots, s^I, \\
a_{k,j}^{\{f,s,i\}} &= a_{i,j}^O + c_k^I (a_{i+1,j}^O - a_{i,j}^O), \quad i = 1, \dots, s^O, \quad j = 1, \dots, i, \quad k = 1, \dots, s^I, \\
a_{k,p}^{\{f,f,i,l\}} &= \begin{cases} b_p^I (c_{l+1}^O - c_l^O) & l = 1, \dots, i - 1 \\ a_{k,p}^I (c_{i+1}^O - c_i^O) & l = i, \end{cases} \quad i = 1, \dots, s^O, \quad p, k = 1, \dots, s^I, \\
b_p^{\{f,i\}} &= b_p^I (c_{i+1}^O - c_i^O) \quad i = 1, \dots, s^O, \quad p = 1, \dots, s^I, \\
b_p^{\{s\}} &= b_p^O, \quad p = 1, \dots, s^I.
\end{aligned}$$

For Multirate Infinitesimal Step methods, Sandu and Günther proved that given 3rd order base methods, Equation (2.56) is the additional necessary and sufficient condition for making the multirate method 3rd order [43]

$$\frac{1}{2} \sum_{i=2}^{s^O} (c_i^O - c_{i-1}^O) (\mathbf{e}_i^\top + \mathbf{e}_{i-1}^\top) A^O c^O + (1 - c_{s^O}^O) \left( \frac{1}{4} + \frac{1}{2} \mathbf{e}_{s^O}^\top A^O c^O \right) = \frac{1}{6}. \quad (2.56)$$

Equation (2.56) is equivalent to the condition implicitly defined by Knoth and Wensch in their 2009 paper [57]. Their aim is to improve the stability region of these methods, by using genetic optimization on two free parameters and then using visual inspection to compare the resulting stability plots. They use this strategy to pick 3 specific sets of MIS coefficients to test. Knoth and Wensch's 2014 paper improves on their earlier results by strictly defining the optimization objectives by aiming to minimize error coefficients and maximize the stable step size [29]. In both cases, the stability is optimized by considering a simplification of the compressible Euler equations as a stability test problem [29].

If the inner and outer base method are the same Runge-Kutta method, then the time-scale separation based on the ratio of fast function calls to slow function calls is  $m = s^O$ . This can be increased through recursion, as suggested by Schlegel, Knoth and Arnold [50]. Increasing the time-scale separation of these methods through recursion is both an asset and a

detriment. They are very simple to write down and very simple to implement. However, since the method's time-scale separation must be  $m = (s^O)^j$ ,  $j \in \mathbb{N}$ , the time-scale separation of the problem may not be captured precisely, which leads to more computational work.

We also note that these authors have presented another formulation where instead of increasing the time-scale separation through recursion, the inner base method is composed of a sub-cycled inner method. This allows for more general time-scale separations  $m = (s^O)^* j$ ,  $j \in \mathbb{N}$ . This approach, which uses recursion to address operator splitting and subcycling to address time-scale separations, is fully described in Schlegel's PhD dissertation [49]. We will discuss our application of the GARK framework to this formulation in Chapter 3.

## 2.5. Linear Stability Theory for Multirate Systems

The usual linear stability approach poses a linear test problem, and investigates the stability for varied eigenvalues and time-step sizes. Given an eigenvalue  $\lambda$  having negative real part, the following linear stability test problem is considered

$$y' = \lambda y, y(0) = 1, \tag{2.57}$$

having exact solution  $y = e^{\lambda t}$ . A recurrence relation is developed to relate successive solution vectors  $y_{n+1}$  and  $y_n$  by an amplification function  $R$ :

$$y_{n+1} = R(h\lambda) y_n = \cdots = [R(h\lambda)]^{n+1} y_0 = [R(h\lambda)]^{n+1}.$$

Since the true solution  $y \rightarrow 0$  as  $t \rightarrow \infty$ , this leads to the definition of the stability region associated with the linear stability test problem:  $S = \{h\lambda = z \in \mathbb{C} : |R(z)| \leq 1\}$ . While this stability concept is sufficient for most IVP methods, it does not capture the complexity of multiple eigenvalues for the different parts of a multirate method. An alternate multirate stability approach considered by Sandu and Günther for their Multirate Generalized Additive Runge Kutta is an additive linear stability approach which focuses on how the stability of the base methods affect the stability of the multirate method directly. This approach was previously used by Wensch and Knoth in 2009 for MIS methods to examine stability properties under the assumption of an infinite number of fast substeps [57]. Equation (2.58)

below describes the initial value problem used by Sandu and Günther to test linear stability given the dominant eigenvalues of  $f^{\{f\}}$  and  $f^{\{s\}}$ ,  $\lambda_f$  and  $\lambda_s$  respectively, where  $\lambda_s$  is assumed to have negative real part [21, 44]:

$$y' = \lambda_f y + \lambda_s y, \quad y(t_0) = 1. \quad (2.58)$$

Similar to the previous linear stability recurrence relation, a recurrence relation,

$$y_{n+1} = R(h\lambda_s, h\lambda_f) y_n \implies y_n = [R(h\lambda_s, h\lambda_f)]^n y_0 = [R(h\lambda_s, h\lambda_f)]^n,$$

can be derived [21]. In Sandu and Günther’s analysis of this linear stability function for the MGARK, they define conditions on the coupling matrices  $\mathbf{A}^{\{f,s,\lambda\}}$  and  $\mathbf{A}^{\{s,f,\lambda\}}$  such that if time steps are chosen that guarantee the individual algebraic stability of the fast base method and the slow base method, then this is “a sufficient condition for the stability of the overall multirate method” [21]. Sandu and Günther call this Theorem 1 (Stability of multirate GARK schemes), which is restated below. [21]

**Theorem 2.1** *Consider a multirate GARK scheme with positive fast weights,  $b_i^{\{f\}} > 0$  for  $i = i, \dots, s^{\{f\}}$ . The scheme is stability-decoupled iff  $\mathbf{A}^{\{f,s\}}$  is given by*

$$A^{\{f,s,\mu\}} = [B^{\{f\}}]^{-1} \left( b^{\{f\}} b^{\{s\}\top} - A^{\{s,f,\mu\}} B^{\{s\}} \right), \quad \mu = 1, \dots, m.$$

$$\text{with } B_{i,j}^{\{f\}} = \begin{cases} b_i^{\{f\}} & i = j \\ 0 & i \neq j \end{cases} \quad \text{and } B_{i,j}^{\{s\}} = \begin{cases} b_i^{\{s\}} & i = j \\ 0 & i \neq j \end{cases}$$

This theorem on stability–decoupling focuses on defining coupling matrix  $\mathbf{A}^{\{f,s\}}$  based upon existing base methods and choice of coupling matrix  $\mathbf{A}^{\{s,f\}}$ , and directly applies only to multirate GARK schemes. Another potential stability concept is to use the more general approach to linear stability proposed by Sandu and Günther’s in 2016 paper for a GARK method (2.1) which proposes the following test problem:  $y' = \sum_{i=1}^N \lambda^{\{i\}} y$ ,  $y(0) = 1$  [44]. By restricting the GARK table to the 2x2 GARK table used in multirate contexts and relabeling



according to fast and slow tables, we can represent the GARK stability concept in a multirate context, resulting in the following stability function,

$$R(h\lambda_f, h\lambda_s) = 1 + \begin{bmatrix} h\lambda_f \mathbf{b}^{\{f\}\top} & h\lambda_s \mathbf{b}^{\{s\}\top} \end{bmatrix} \begin{bmatrix} I_{s^{\{f\}}} - h\lambda_f \mathbf{A}^{\{f,f\}} & h\lambda_s \mathbf{A}^{\{f,s\}} \\ h\lambda_f \mathbf{A}^{\{s,f\}} & I_{s^{\{s\}}} - h\lambda_s \mathbf{A}^{\{s,s\}} \end{bmatrix}^{-1} \begin{bmatrix} \mathbb{1}_{s^{\{f\}}} \\ \mathbb{1}_{s^{\{s\}}} \end{bmatrix}.$$

A more thorough concept of linear stability for multirate methods was first proposed by Kværno [33] in 2000. This stability concept was not published again until 2007, when it began to be commonly used [24, 31, 47, 45]. The previously mentioned test problems are specific instances of Kværno's stability test problem with various coefficients set to 0. The benefit of Kværno's stability concept is that unlike the others it includes in its analysis the mechanisms for fast-slow coupling within the fast and slow solution updates. More specifically, we consider the test problem (2.59) below, having fast variable  $y_f$  and slow variable  $y_s$ ,

$$\begin{aligned} \mathbf{y}'(t) &= G\mathbf{y}(t), \quad \mathbf{y}(t) = \begin{pmatrix} y_f(t) \\ y_s(t) \end{pmatrix}, \quad \mathbf{y}(t_0) = \begin{pmatrix} y_f(t_0) \\ y_s(t_0) \end{pmatrix} \\ G &= \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix}, \quad Z = hG = \begin{pmatrix} hg_{11} & hg_{12} \\ hg_{21} & hg_{22} \end{pmatrix}. \end{aligned} \quad (2.59)$$

As usual, the stability function is defined by constructing a recursion based on the numerical method,

$$\begin{pmatrix} y_{f,n+1} \\ y_{s,n+1} \end{pmatrix} = S(Z) \begin{pmatrix} y_{f,n} \\ y_{s,n} \end{pmatrix} \quad \text{where} \quad S(Z) = \begin{pmatrix} s_{11}(Z) & s_{12}(Z) \\ s_{21}(Z) & s_{22}(Z) \end{pmatrix}. \quad (2.60)$$

The multirate method is stable if the dominant eigenvalue of  $S$  has magnitude less than 1 for a given step size  $h$  and coupling matrix  $G$ . While more complex than typical linear stability regions (seemingly depends on five parameters,  $h$ ,  $g_{11}$ ,  $g_{12}$ ,  $g_{21}$  and  $g_{22}$  instead of only two,  $h$  and  $\lambda$ ), Kværno's test problem gives plottable stability information since we

can find the eigenvalues of  $S$  as a function of 3 real parameters. The parametrization used in papers from 2007, 2008 and 2014 by Kuhn and Savcenko depends on the three derived parameters  $\xi$ ,  $\eta$ , and  $\kappa$  [31, 47, 45]. A similar set of parameters was used by Constantinescu in 2013 [9]. The parameter  $\beta = \frac{g_{12}g_{21}}{g_{11}g_{22}}$  measures the off-diagonal coupling strength of the problem. Given the assumption that  $g_{11} < 0$  and  $g_{22} < 0$ , the eigenvalues of  $G$  have negative real part if  $\beta < 1$ . We also assume that  $z_{11} \leq 1$ . The parameter  $\eta = \frac{\beta}{2-\beta}$  measures has the same sign as  $g_{12}g_{21}$ , and changes sign when  $g_{12}g_{21} = g_{11}g_{22}$ . For rescaled versions of this problem where  $g_{11} = -1$  and  $g_{22} = m = \kappa$ , this change of sign occurs when  $g_{12}g_{21} = m$ . Constantinescu's rescaling of this problem assumes that  $g_{22} = -m$ ,  $g_{11} = -1$ ,  $|g_{21}| \leq m$ , and  $|g_{12}| \leq 1$ . If those assumptions are satisfied, then  $g_{12}g_{21} \leq m$ , so  $\eta$  has the same sign as  $g_{12}g_{21}$ . In this case, if  $\eta < 0$ , the off-diagonal coupling terms have opposite signs. Using our previous assumptions on  $\beta$  and on  $z_{11}$ ,  $-1 \leq z_{11} \leq 1$ . The problem stiffness is related to the parameter  $\xi = \frac{hg_{11}}{1-hg_{11}} \in (-1, 0)$ , so as  $\xi \rightarrow -1$ ,  $hg_{11} \rightarrow -\infty$ . The problem stiffness increases as  $\xi$  moves from 0 to -1. The parameter  $\kappa = g_{22}/g_{11}$  measures the time-scale separation of the problem. In these references, a method is presented for plotting the corresponding stability region for a fixed coupling strength. Figure 2.1 gives an example of how we will plot these stability regions, and how to read those plots. Note that  $\eta = -1$  is never stable, since this value of  $\eta$  corresponds to off-diagonal coupling of opposite sign which is infinitely large.

Since Kværno's stability concept is the most general for multirate methods and takes into account stability resulting from how the fast and slow components are coupled together, we will base our stability analyses in the remainder of this thesis on this approach.

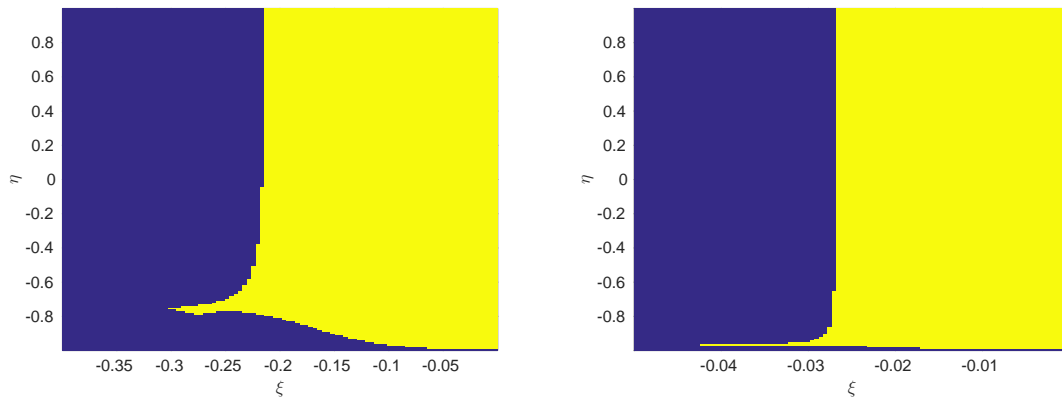


Figure 2.1.  $\kappa = 10$  and  $\kappa = 100$  for similarly constructed methods. Note that the  $\xi$  axis changes for the  $\kappa = 100$  plot on the right, to show that the stability area is diminished for larger  $\kappa$ . The yellow area is stable by our stability concept.  $\eta < 0$  shows stability parameters where the off-diagonal coupling terms have opposite signs. One explanation for the extended stability in the  $\eta = -0.6$  and  $\xi = -0.25$  on the  $\kappa = 10$  plot is that the opposite signs on the off-diagonal coupling terms have a net effect of decreasing the stability requirements.

## Chapter 3

### Framework for Relaxed Multirate Infinitesimal Step Methods

#### 3.1. Proposed Flexible Multirate GARK (fmGARK) structure

We tested a wide range of existing multirate methods in order to evaluate strengths and weaknesses of existing approaches, and to consider potential areas for improvement. One potential limitation that we identified in the multirate GARK structure is the assumption that the same fast method is used for all fast micro-steps. To ameliorate this, we consider a more general multirate GARK structure than the one proposed by Sandu and Günther [43, 21]. Our description of the fmGARK structure as a GARK is modeled after Sandu and Günther’s Theorem 4, “the MIS scheme is a particular instance of a GARK method” [20]. We propose a new multirate GARK framework, which we name Flexible Multirate Generalized–Structure Additively–Partitioned Runge–Kutta (fmGARK) methods. This is a more flexible approach than the multirate GARK described in Section 2.2 [21]. In the following section, we specify the fmGARK structure, and show how it describes a particular instance of a GARK method. Notation is consistent with our earlier definition of an MIS in a GARK context in Section 2.3.

#### 3.2. Structure

We first considered constructing this structure by relaxing assumptions about the initial conditions and base methods used in internal fast sub-steps. This structure is based on the MIS definition, and cannot be used to describe the entire set of multirate methods the

mGARK does. Definitions: Let  $T^{I_n} = \begin{array}{c|c} \mathbf{c}^{I_n} & \mathbf{A}^{I_n} \\ \hline & (\mathbf{b}^{I_n})^\top \end{array}$  be an “inner” Runge–Kutta table of at

least second order, and  $T_O = \begin{array}{c|c} \mathbf{c}^O & \mathbf{A}^O \\ \hline & (\mathbf{b}^O)^\top \end{array}$  be an “outer” Runge-Kutta table of at least third order. We assume that these Runge-Kutta methods have  $s^{I_n}$  and  $s^O$  stages, respectively.

In general, elementwise form, we define the fmGARK matrix entries as follows. We index the slow stages as  $i = 1, \dots, s^O$  similar to how they are indexed for the MIS methods. The superscript indices correspond to the general structure from (2.55), and indicate the GARK tableau block, as stated in Section 2.3. We include below some examples of how the coefficients relate to the block structure. The algorithm we developed and tested assumes both the inner and outer base methods are explicit. Our later investigation of the order condition requirements regarding our newly developed methods assumes that the first stage of the inner base method is explicit. The indexes given for coefficients here assume explicit inner and outer base methods; these indexes may be extended to include the full Runge-Kutta table from each base method if needed. The  $\mathbf{A}^{\{s,s\}}$  matrix shows how the slow base method is only used once in the slow component method, unlike the fast base method, which is used  $m$  times in the fast component method.

$$\mathbf{A}^{\{s,s\}} = \begin{array}{c|cccc} & 0 & 0 & 0 & 0 \\ \hline & a_{2,1}^{\{s,s\}} & 0 & 0 & 0 \\ & a_{3,1}^{\{s,s\}} & a_{3,2}^{\{s,s\}} & 0 & 0 \\ & a_{4,1}^{\{s,s\}} & a_{4,2}^{\{s,s\}} & a_{4,3}^{\{s,s\}} & 0 \end{array}$$

The block matrices  $\mathbf{A}^{\{s,f,i\}}$  describe how the fast stages associated with the  $i$ th fast sub-step affect the slow stages.

$$\mathbf{A}^{\{s,f,1\}} = \begin{array}{c|cccc} & 0 & 0 & 0 & 0 \\ \hline & a_{2,1}^{\{s,f,1\}} & a_{2,2}^{\{s,f,1\}} & a_{2,3}^{\{s,f,1\}} & a_{2,4}^{\{s,f,1\}} \\ & a_{3,1}^{\{s,f,1\}} & a_{3,2}^{\{s,f,1\}} & a_{3,3}^{\{s,f,1\}} & a_{3,4}^{\{s,f,1\}} \\ & a_{4,1}^{\{s,f,1\}} & a_{4,2}^{\{s,f,1\}} & a_{4,3}^{\{s,f,1\}} & a_{4,4}^{\{s,f,1\}} \end{array} \quad \mathbf{A}^{\{s,f,2\}} = \begin{array}{c|cccc} & 0 & 0 & 0 & 0 \\ \hline & 0 & 0 & 0 & 0 \\ & a_{3,1}^{\{s,f,2\}} & a_{3,2}^{\{s,f,2\}} & a_{3,3}^{\{s,f,2\}} & a_{3,4}^{\{s,f,2\}} \\ & a_{4,1}^{\{s,f,2\}} & a_{4,2}^{\{s,f,2\}} & a_{4,3}^{\{s,f,2\}} & a_{4,4}^{\{s,f,2\}} \end{array}$$

The first row of  $\mathbf{A}^{\{f,s,l\}}$  has one fewer non-zero entry than the other rows, since we have assumed the first stage of the base methods is explicit. These matrices describe how the slow stages affect the fast stages.

$$\mathbf{A}^{\{f,s,1\}} = \begin{vmatrix} 0 & 0 & 0 & 0 \\ a_{2,1}^{\{f,s,1\}} & 0 & 0 & 0 \\ a_{3,1}^{\{f,s,1\}} & 0 & 0 & 0 \\ a_{4,1}^{\{f,s,1\}} & 0 & 0 & 0 \end{vmatrix} \quad \mathbf{A}^{\{f,s,2\}} = \begin{vmatrix} a_{1,1}^{\{f,s,2\}} & 0 & 0 & 0 \\ a_{2,1}^{\{f,s,2\}} & a_{2,2}^{\{f,s,2\}} & 0 & 0 \\ a_{3,1}^{\{f,s,2\}} & a_{3,2}^{\{f,s,2\}} & 0 & 0 \\ a_{4,1}^{\{f,s,2\}} & a_{4,2}^{\{f,s,2\}} & 0 & 0 \end{vmatrix}$$

In addition to  $\mathbf{A}^{\{f,f\}}$  being block lower triangular,  $\mathbf{A}^{\{f,f,i,i\}}$  has the same nonzero pattern as  $A^{I_n}$ ,

$$\mathbf{A}^{\{f,f,1,1\}} = \begin{vmatrix} 0 & 0 & 0 & 0 \\ a_{2,1}^{\{f,f,1,1\}} & 0 & 0 & 0 \\ a_{3,1}^{\{f,f,1,1\}} & a_{3,2}^{\{f,f,1,1\}} & 0 & 0 \\ a_{4,1}^{\{f,f,1,1\}} & a_{4,2}^{\{f,f,1,1\}} & a_{4,3}^{\{f,f,1,1\}} & 0 \end{vmatrix}$$

$$\mathbf{A}^{\{f,f,2,1\}} = \begin{vmatrix} a_{1,1}^{\{f,f,2,1\}} & a_{1,2}^{\{f,f,2,1\}} & a_{1,3}^{\{f,f,2,1\}} & a_{1,4}^{\{f,f,2,1\}} \\ a_{2,1}^{\{f,f,2,1\}} & a_{2,2}^{\{f,f,2,1\}} & a_{2,3}^{\{f,f,2,1\}} & a_{2,4}^{\{f,f,2,1\}} \\ a_{3,1}^{\{f,f,2,1\}} & a_{3,2}^{\{f,f,2,1\}} & a_{3,3}^{\{f,f,2,1\}} & a_{3,4}^{\{f,f,2,1\}} \\ a_{4,1}^{\{f,f,2,1\}} & a_{4,2}^{\{f,f,2,1\}} & a_{4,3}^{\{f,f,2,1\}} & a_{4,4}^{\{f,f,2,1\}} \end{vmatrix} \quad \mathbf{A}^{\{f,f,2,2\}} = \begin{vmatrix} 0 & 0 & 0 & 0 \\ a_{2,1}^{\{f,f,2,2\}} & 0 & 0 & 0 \\ a_{3,1}^{\{f,f,2,2\}} & a_{3,2}^{\{f,f,2,2\}} & 0 & 0 \\ a_{4,1}^{\{f,f,2,2\}} & a_{4,2}^{\{f,f,2,2\}} & a_{4,3}^{\{f,f,2,2\}} & 0 \end{vmatrix}.$$

Now that we have shown the non-zero structure of these coupling matrices, let us consider how specifically to determine those non-zero coefficients.

The base method tableau  $T_{I_n}$  and  $T_O$  determine the resulting multirate method. We define component-wise the intermediate fast sub-step solutions using the same MIS problem formulation as in Section 2.4 as follows,

$$\begin{aligned}
a_{i,j}^{\{s,s\}} &= a_{i,j}^O, \quad j = 1, \dots, i-1, \\
a_{i,p}^{\{s,f,l\}} &= b_p^{I_l} (c_{l+1}^O - c_l^O), \quad l = 1, \dots, s^O - 1, \quad i = l+1, \dots, s^O, \quad p = 1, \dots, s^{I_l}, \\
a_{k,j}^{\{f,s,i\}} &= a_{i,j}^O + c_k^{I_i} (a_{i+1,j}^O - a_{i,j}^O), \quad i = 1, \dots, s^O - 1, \quad j = 1, \dots, i, \quad k = 1, \dots, s^{I_i}, \\
a_{k,p}^{\{f,f,i,l\}} &= \begin{cases} b_p^{I_l} (c_{l+1}^O - c_l^O) & l = 1, \dots, i-1 \\ a_{k,p}^{I_i} (c_{i+1}^O - c_i^O) & l = i, \end{cases} \quad i = 1, \dots, s^O - 1, \quad p, k = 1, \dots, s^{I_i},
\end{aligned}$$

where all values not defined are 0. The final fast sub-step defines the last block-row of the  $\mathbf{A}^{\{f,f\}}$  and  $\mathbf{A}^{\{f,s\}}$  tables, as well as the final step solution update in a similar fashion,

$$\begin{aligned}
a_{k,j}^{\{f,s,s^O\}} &= a_{s^O,j}^O + c_k^{I_{s^O}} (b_j^O - a_{s^O,j}^O), \quad j = 1, \dots, s^O - 1, \quad k = 1, \dots, s^{I_{s^O}}, \\
a_{k,m}^{\{f,f,s^O,l\}} &= \begin{cases} b_p^{I_l} (c_{l+1}^O - c_l^O) & l = 1, \dots, s^O - 1 \\ a_{k,p}^{I_{s^O}} (1 - c_{s^O}^O) & l = s^O, \end{cases} \quad p, k = 1, \dots, s^{I_l}.
\end{aligned}$$

Again as with the MIS methods from Section 2.4, we augment the Butcher tableau of the base methods,

$$\begin{array}{c|c} \mathbf{c}^O & A^O \\ \hline \mathbf{b}^{O\top} & \end{array} \rightarrow \begin{array}{c|c} \mathbf{c}^O & A^O \quad \mathbf{0} \\ \hline 1 & \mathbf{b}^{O\top} \quad 0 \\ \hline & \mathbf{b}^{O\top} \quad 0 \end{array}$$

which allows us to write the MIS method in the fmGARK formulation more concisely, by considering  $i = 1, \dots, s^O + 1$  for  $A$ 's and  $c$ 's. This is relevant to how we consider implementing these methods in a memory-efficient manner, since the extra stage of the  $A$ 's shows the complete fast substep, including the final step at  $c_i = 1$ .

This more concise formulation written componentwise is as follows:

$$\begin{aligned}
a_{i,j}^{\{s,s\}} &= a_{i,j}^O, \quad j = 1, \dots, i-1, \\
b_i^{\{s\}} &= b_i^O, \\
a_{i,p}^{\{s,f,l\}} &= b_p^{I_l} (c_{l+1}^O - c_l^O), \quad l = 1, \dots, s^O, \quad i = l+1, \dots, s^O, \quad p = 1, \dots, s^{I_l}, \\
a_{k,j}^{\{f,s,i\}} &= a_{i,j}^O + c_k^{I_i} (a_{i+1,j}^O - a_{i,j}^O), \quad i = 1, \dots, s^O, \quad j = 1, \dots, i, \quad k = 1, \dots, s^{I_i}, \\
a_{k,p}^{\{f,f,i,l\}} &= \begin{cases} b_p^{I_l} (c_{l+1}^O - c_l^O) & l = 1, \dots, i-1 \\ a_{k,p}^{I_l} (c_{i+1}^O - c_i^O) & l = i, \end{cases} \quad i = 1, \dots, s^O, \quad p, k = 1, \dots, s^{I_i}.
\end{aligned}$$

The structure of the fmGARK leaves  $\mathbf{b}^{\{f\}}$  undetermined. Specific choices for  $\mathbf{b}^{\{f\}}$  will be discussed in Chapter 4. We note that with this approach, we may consider the traditional MIS fast coefficients as an embedding option  $\tilde{\mathbf{b}}_p^{\{f,i\}} = b_p^{I_i} (c_{i+1}^O - c_i^O)$ , alongside  $\mathbf{b}_i^{\{s\}} = b_i^O$ , leaving  $\mathbf{b}^{\{f\}}$  free to be determined based on the user's target objective.

We note that with the above definition, the internal ‘‘row-sum’’ consistency conditions are satisfied, since

$$\begin{aligned}
c_i^{\{s,s\}} &= c_i^O, \\
c_i^{\{s,f\}} &= \sum_{l=2}^i (c_l^O - c_{l-1}^O) \sum_{p=1}^{s^{I_{l-1}}} b_p^{I_{l-1}} = \sum_{l=2}^i (c_l^O - c_{l-1}^O) = c_i^O, \\
c_k^{\{f,s,i\}} &= \sum_{j=1}^{i-1} a_{i-1,j}^O + c_k^{I_{i-1}} (a_{i,j}^O - a_{i-1,j}^O) = c_{i-1}^O + c_k^{I_{i-1}} (c_i^O - c_{i-1}^O), \\
c_k^{\{f,f,i\}} &= \sum_{p=1}^{s^{I_{i-1}}} b_p^{I_{i-1}} \sum_{l=2}^{i-1} (c_l^O - c_{l-1}^O) + \sum_{p=1}^{s^{I_{i-1}}} a_{k,p}^{I_{i-1}} (c_i^O - c_{i-1}^O) = c_{i-1}^O + c_k^{I_{i-1}} (c_i^O - c_{i-1}^O).
\end{aligned}$$

We can represent these stage times in vector form as follows:

$$\mathbf{c}^{\{f\}} = \begin{bmatrix} \mathbf{c}^{\{f,1\}} \\ \vdots \\ \mathbf{c}^{\{f,s^O\}} \end{bmatrix} \quad \text{where } \mathbf{c}^{\{f,i\}} = \begin{bmatrix} c_1^{\{f,f,i\}} \\ \vdots \\ c_{s^I}^{\{f,f,i\}} \end{bmatrix} = \begin{bmatrix} c_1^{\{f,s,i\}} \\ \vdots \\ c_{s^I}^{\{f,s,i\}} \end{bmatrix} = \begin{bmatrix} c_{i-1}^O + c_1^{I_{i-1}} (c_i^O - c_{i-1}^O) \\ \vdots \\ c_{i-1}^O + c_{s^O}^{I_{i-1}} (c_i^O - c_{i-1}^O) \end{bmatrix}.$$



This fully defines an fmGARK scheme as an instance of a GARK method. Representing the stage solutions in a similar block-vector form to  $\mathbf{c}^{\{f\}}$  gives us the following notation,

$$\mathbf{k}^{\{f\}} = \begin{bmatrix} \mathbf{k}^{\{f,1\}} \\ \vdots \\ \mathbf{k}^{\{f,s^O\}} \end{bmatrix} \text{ stage vectors: } \mathbf{k}^{\{f,i\}} = \begin{bmatrix} \mathbf{k}_1^{\{f,i\}} \\ \vdots \\ \mathbf{k}_{s^I}^{\{f,i\}} \end{bmatrix}, \text{ at stage times: } \begin{bmatrix} c_{i-1}^O + c_1^{I_{i-1}} (c_i^O - c_{i-1}^O) \\ \vdots \\ c_{i-1}^O + c_{s^O}^{I_{i-1}} (c_i^O - c_{i-1}^O) \end{bmatrix}.$$

Consider Equations (2.21)-(2.23) for the stage and solution updates for the fully explicit  $2 \times 2$  GARK method. These update formulae allow us to simplify the fmGARK stage and solution update formulae by substituting for the specific block-structure. We assume that the first stage of the inner base method is explicit. The MIS structure is formulated so that if there is no coupling between the fast and slow processes in the differential equation, the original base methods are used as they would be if there was no interpolation or coupling matrices in the multirate method. For explicit methods, this means that the coefficients in  $\mathbf{A}_{1,q}^{\{f,f,i,j\}}$  are identical to the coefficients in  $\mathbf{A}_{i,q}^{\{s,f,j\}}$ . In order to take advantage of this structure, we define  $\hat{\mathbf{k}}_1^{\{f,i\}}$  to be  $\mathbf{k}_1^{\{f,i\}}$  minus the slow coupling portion, that is,  $\hat{\mathbf{k}}_1^{\{f,i\}} = \mathbf{k}_1^{\{f,i\}} - h \sum_{l=1}^{s^{\{s\}}} \mathbf{a}_{jl}^{\{f,s\}} f^{\{s\}} \left( t + \mathbf{c}_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right)$ . We can use this portion of the stage solution to accumulate an initial condition for  $\mathbf{k}_p^{\{f,i\}}$ , where  $p = 2, \dots, s^{I_i}$  and for calculating the next initial condition  $\hat{\mathbf{k}}_1^{\{f,i+1\}}$ . Finally, this accumulated initial condition  $\mathbf{k}_1^{\{f,s^O\}}$  can be used with the remaining  $\mathbf{k}_p^{\{f,s^O\}}$  stages to form the fast part of the embedded MIS solution method.

$$\hat{\mathbf{k}}_1^{\{f,1\}} = \boxed{\mathbf{y}_n} \quad (3.1)$$

$$\mathbf{k}_1^{\{f,i\}} = \mathbf{y}_n + \sum_{l=1}^{j-1} h * \mathbf{a}_{jl}^{\{f,f\}} * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + \sum_{l=1}^{s^{\{s\}}} h * \mathbf{a}_{jl}^{\{f,s\}} * f^{\{s\}} \left( t + \mathbf{c}_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \quad (3.2)$$

$$\mathbf{k}_1^{\{f,i\}} = \boxed{\hat{\mathbf{k}}_1^{\{f,i\}} + \sum_{l=1}^{s^{\{s\}}} h * \mathbf{a}_{jl}^{\{f,s,i\}} * f^{\{s\}} \left( t + \mathbf{c}_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right)} \quad (3.3)$$

$$\hat{\mathbf{k}}_1^{\{f,i\}} = \mathbf{y}_n + \sum_{l=1}^{j-1} h * \mathbf{a}_{jl}^{\{f,f\}} * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) \quad (3.4)$$

$$= \mathbf{y}_n + \sum_{p=1}^{i-1} \sum_{l=1}^{s^{I_i}} h * \mathbf{a}_{jl}^{\{f,f,i,p\}} * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f,p\}} h, \mathbf{k}_l^{\{f,p\}} \right) \quad (3.5)$$

$$= \mathbf{y}_n + \sum_{p=1}^{i-1} \sum_{l=1}^{s^{I_i}} h * b_l^{I_p} (c_{p+1}^O - c_p^O) * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f,p\}} h, \mathbf{k}_l^{\{f,p\}} \right) \quad (3.6)$$

$$= \sum_{p=1}^{i-1} \mathbf{k}_1^{\{f,i\}} + \sum_{l=1}^{s^{I_i}} h * b_l^{I_{i-1}} (c_i^O - c_{i-1}^O) * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f,i-1\}} h, \mathbf{k}_l^{\{f,i-1\}} \right) \quad (3.7)$$

$$\mathbf{k}_j^{\{f,i\}} = \boxed{\hat{\mathbf{k}}_1^{\{f,i\}} + \sum_{l=1}^{j-1} h * \mathbf{a}_{jl}^{\{f,f,i,i\}} * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + \sum_{l=1}^{s^{\{s\}}} h * \mathbf{a}_{jl}^{\{f,s,i\}} * f^{\{s\}} \left( t + \mathbf{c}_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right)} \quad (3.8)$$

$$\mathbf{k}_i^{\{s\}} = \boxed{\hat{\mathbf{k}}_1^{\{f,i\}} + \sum_{l=1}^{i-1} h * \mathbf{a}_{il}^{\{s,s\}} * f^{\{s\}} \left( t + \mathbf{c}_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right)} \quad (3.9)$$

$$\tilde{\mathbf{y}}_{n+1} = \mathbf{k}_1^{\{f,s^O\}} + \sum_{l=1}^{s^{I_sO}} h * \mathbf{b}_l^{I_sO} (1 - c_{s^O}) * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f,s^O\}} h, \mathbf{k}_l^{\{f,s^O\}} \right) \quad (3.10)$$

$$+ \sum_{l=1}^{s^{\{s\}}} h * \mathbf{b}_l^{\{s\}} * f^{\{s\}} \left( t + \mathbf{c}_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \quad (3.11)$$

$$\mathbf{y}_{n+1} = \boxed{\mathbf{y}_n + \sum_{l=1}^{s^{\{f\}}} h * \mathbf{b}_l^{\{f\}} * f^{\{f\}} \left( t + \mathbf{c}_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + \sum_{l=1}^{s^{\{s\}}} h * \mathbf{b}_l^{\{s\}} * f^{\{s\}} \left( t + \mathbf{c}_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right)} \quad (3.12)$$

This notation will assist in describing our memory-efficient implementation for fmGARK methods.

### 3.3. Implementation and Memory Considerations

To ensure no duplicate function calls, we must store vectors the size of our solution vector  $y$ , commensurate with the number of stages of our base methods. In particular, while computing the stage solution updates for  $\mathbf{k}^{\{f,i\}}$  and  $\mathbf{k}_i^{\{s\}}$ , we must store  $\mathbf{k}_1^{\{f,i\}}$ , the  $s^{\{I\}}$  fast function vectors  $\mathbf{f}_l^{\{f,i\}} = f^{\{f\}} \left( t + \mathbf{c}_l^{\{f,i\}} h, \mathbf{k}_l^{\{f,i\}} \right)$ , and the  $s^{\{O\}}$  slow function vectors  $\mathbf{f}_i^{\{s\}} = f^{\{s\}} \left( t + \mathbf{c}_i^{\{s\}} h, \mathbf{k}_i^{\{s\}} \right)$ . This is sufficient storage to compute the embedded MIS solution  $\tilde{\mathbf{y}}_{n+1}$  without requiring recomputed function calls. In order to compute the fmGARK solution  $\mathbf{y}_{n+1}$ , we must also store  $\mathbf{y}_{n+1}$  and the vector of coefficients  $\mathbf{b}^{\{f\}}$ , if these are unique or unstructured. In order to only store  $\max_{i=1}^{s^{\{O\}}} s^{\{I_i\}} + s^{\{O\}} + 2$  vectors the size of our solution vector  $y$ , we overwrite  $\mathbf{f}^{\{f,i\}}$  with  $\mathbf{f}^{\{f,i+1\}}$  after the  $i$ th fast substep has completed, and  $\mathbf{f}^{\{f,i\}}$  has been used to generate the initial condition  $\hat{\mathbf{k}}_1^{\{f,i+1\}}$  for the  $(i+1)$ st fast substep. In the subcycled version of the code, we overwrite  $\mathbf{f}^{\{f,i\}}$  after each subcycled fast substep has completed, and  $\mathbf{f}^{\{f,i\}}$  has been used to update the temporary initial condition  $\hat{\mathbf{k}}_1^{\{f,i\}}$  for the next subcycled fast substep. To emphasize those vectors which are overwritten while remaining consistent with previous notation, we mark out the extra index in the pseudocode, i.e.  $\hat{\mathbf{k}}^{\{f,i\}} \rightarrow \hat{\mathbf{k}}^{\{f,f\}}$  and  $\mathbf{f}^{\{f,i\}} \rightarrow \mathbf{f}^{\{f,f\}}$ . Algorithm 1 gives the pseudocode for a generic fast sub-step when there is subcycling given an fmGARK structure, but does not define specifically how to determine  $\mathbf{A}^{\{f,f\}}$ .

---

**Algorithm 1:** Fast Substep for general  $\mathbf{A}^{\{f,f\}}$  (Corresponds to  $i$ th outer stage,  $k$ th subcycle)

---

**Data:**  $f^{\{f\}}, h, t = t_n, \mathbf{k}_1^{\{f,i\}}, \mathbf{f}_{1,\dots,s^{\{I_k\}}}^{\{f,i\}}, \mathbf{f}_{1,\dots,s^{\{O\}}}^{\{s\}}, i, k, T^I, T^O$

;

// i.e.  $T^O \implies \mathbf{A}^{\{O\}}, \mathbf{b}^{\{O\}}, \mathbf{c}^{\{O\}}, s^{\{O\}}$

**Result:**  $\mathbf{f}_{1,\dots,s^{\{I_k\}}}^{\{f,i\}}$

- 1  $\mathbf{c}_j^{\{f,l\}} = c_i^O + c_j^{I_i} (c_{i+1}^O - c_i^O)$ ; // Calculate stage times for  $i$ th fast substep
- 2  $l=1$ ;
- 3  $\mathbf{k}_l^{\{f,l\}} = \mathbf{k}_1^{\{f,l\}} + \sum_{j=1}^{l-1} ha_{lj}^{\{f,f,i,i\}} \mathbf{f}_j^{\{f,l\}} + \sum_{j=1}^i ha_{lj}^{\{f,s,i\}} \mathbf{f}_j^{\{s\}}$ ; // Linear combination
- 4  $\mathbf{f}_l^{\{f,l\}} = f^{\{f\}} \left( t + \mathbf{c}_l^{\{f,i\}} h, \mathbf{k}_l^{\{f,l\}} \right)$ ; // Calculate func vecs:1st fast stage
- 5 **for**  $l \leftarrow 1$  **to**  $s^{\{I_k\}}$  **do**
- 6  $\left| \mathbf{k}_l^{\{f,l\}} = \mathbf{k}_1^{\{f,l\}} + \sum_{j=1}^{l-1} ha_{lj}^{\{f,f,i,i\}} \mathbf{f}_j^{\{f,l\}} + \sum_{j=1}^i ha_{lj}^{\{f,s,i\}} \mathbf{f}_j^{\{s\}} \right$ ; // Linear comb
- 7  $\left| \mathbf{f}_l^{\{f,l\}} = f^{\{f\}} \left( t + \mathbf{c}_l^{\{f,i\}} h, \mathbf{k}_l^{\{f,l\}} \right) \right$ ; // Calculate func vecs: $l$ th fast stage
- 8 **end**

---

Consider the case where  $T^{\{I_i\}} = \frac{\mathbf{c}^{I_i} \mid \mathbf{A}^{I_i}}{\mid (\mathbf{b}^{I_i})^\top}$  is chosen to be a composition of internal steps, so that the fast stages are sub-cycled. This commonly happens in practice, and removes the computational overhead of recursion. When we represent this as a butcher table, a composition of steps of an inner method is represented as shown in Equation (3.13).

$$T^{\{I_i\}} = \begin{array}{c|cccc}
\left(\frac{1}{n_i}\mathbf{c}^I\right) & \frac{1}{n_i}A^I & & & \\
\left(\frac{1}{n_i}\mathbf{c}^I + \frac{1}{n_i}\mathbf{1}\right) & \frac{1}{n_i}\mathbf{b}^I & \cdots & & \\
\vdots & \vdots & & & \\
\left(\frac{1}{n_i}\mathbf{c}^I + \frac{2}{n_i}\mathbf{1}\right) & \vdots & & \frac{1}{n_i}A^I & \\
\vdots & \vdots & & \frac{1}{n_i}\mathbf{b}^I & \cdots \\
\vdots & \vdots & & \vdots & \\
\vdots & \vdots & & \frac{1}{n_i}\mathbf{b}^I & \\
\left(\frac{1}{n_i}\mathbf{c}^I + \frac{n_i-1}{n_i}\mathbf{1}\right) & \frac{1}{n_i}\mathbf{b}^I & \cdots & \cdots & \vdots & \frac{1}{n_i}A^I \\
\hline
& & & \frac{1}{n_i}\mathbf{b}^I & \frac{1}{n_i}\mathbf{b}^I & \\
\hline
& \frac{1}{n_i}\mathbf{b}^I & \cdots & \cdots & \frac{1}{n_i}\mathbf{b}^I & \frac{1}{n_i}\mathbf{b}^I
\end{array} \quad (3.13)$$

Since any updates to slow function values are calculated at the same time as the first fast stage in  $T^{\{I_i\}}$ , each block of fast stage updates in  $T^{\{I_i\}}$  requires the same amount of mem-

ory or re-computation as using a single method of the size of  $\frac{\mathbf{c}^I \mid A^I}{\mid (\mathbf{b}^I)^\top}$ . Algorithm 2

describes how  $\mathbf{f}_{1,\dots,s}^{\{f,i\}}$  is updated. This update is similar to an ARK step where the fast function  $f^{\{f\}}$  corresponds to  $A^{\{1\}}$ , and the slow function  $f^{\{s\}}$  corresponds to  $A^{\{2\}}$ . The non-zero structure of these matrices of coefficients is demonstrated in Equations (3.14) and (3.15). When  $T^{\{I_i\}}$  is a diagonally-implicit Runge-Kutta (where  $a_{i,i}^{\{I\}}$  is not necessarily equal to zero), line 8 changes to  $\mathbf{k}_i^{\{f,i\}} = \hat{\mathbf{k}}_1^{\{f,i\}} + \sum_{j=1}^l h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (a_{i,j}^{\{I\}}) \right) \mathbf{f}_j^{\{f,i\}} +$

$\sum_{j=1}^i h \left( a_{i,j}^{\{O\}} + \left( \frac{c_i^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,j}^{\{O\}} - a_{i,j}^{\{O\}}) \right) \mathbf{f}_j^{\{s\}}$ . To make solving our order conditions more straightforward, we generally assume the first stage of the inner base method to be explicit. This is reflected in the pseudocode shown below.

$$A^{\{1\}} = \begin{pmatrix} 0 & \dots & \dots & \dots & \dots & 0 \\ \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (a_{2,1}^{\{I\}}) \right) & \ddots & & & & \vdots \\ \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (a_{3,1}^{\{I\}}) \right) & \ddots & \ddots & & & \vdots \\ \vdots & & & & & \vdots \\ \vdots & & \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (a_{l,j}^{\{I\}}) \right) & \ddots & & \vdots \\ \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (a_{s\{I\},1}^{\{I\}}) \right) & \dots & \dots & \dots & \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (a_{s\{I\},s\{I\}-1}^{\{I\}}) \right) & 0 \end{pmatrix} \quad (3.14)$$

$$A^{\{2\}} = \begin{pmatrix} \left( a_{i,1}^{\{O\}} + \left( \frac{c_1^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,1}^{\{O\}} - a_{i,1}^{\{O\}}) \right) & \dots & \left( a_{i,i}^{\{O\}} + \left( \frac{c_1^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,i}^{\{O\}} - a_{i,i}^{\{O\}}) \right) & 0 & \dots & 0 \\ \left( a_{i,1}^{\{O\}} + \left( \frac{c_2^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,1}^{\{O\}} - a_{i,1}^{\{O\}}) \right) & \dots & \left( a_{i,i}^{\{O\}} + \left( \frac{c_2^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,i}^{\{O\}} - a_{i,i}^{\{O\}}) \right) & \vdots & & \vdots \\ \vdots & \dots & \vdots & \vdots & & \vdots \\ \left( a_{i,j}^{\{O\}} + \left( \frac{c_j^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,j}^{\{O\}} - a_{i,j}^{\{O\}}) \right) & \dots & \left( a_{i,i}^{\{O\}} + \left( \frac{c_j^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,i}^{\{O\}} - a_{i,i}^{\{O\}}) \right) & \vdots & & \vdots \\ \vdots & \dots & \vdots & \vdots & & \vdots \\ \left( a_{i,1}^{\{O\}} + \left( \frac{c_{s\{I\}}^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,1}^{\{O\}} - a_{i,1}^{\{O\}}) \right) & \dots & \left( a_{i,i}^{\{O\}} + \left( \frac{c_{s\{I\}}^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,i}^{\{O\}} - a_{i,i}^{\{O\}}) \right) & 0 & \dots & 0 \end{pmatrix}. \quad (3.15)$$

Algorithm 3 describes how to incrementally update  $y$ ,  $\tilde{y}$  and  $\hat{\mathbf{k}}_1^{\{f;f\}}$  within a macrostep based on the RFSMR step strategy. Our new methods replace this algorithm in order to create a higher order method overall. We will discuss these new methods, as well as the method-specific algorithm updates Algorithm 5 and Algorithm 6 in Chapter 4. Algorithm 3 is included here solely as an example for potential embedding purposes. Algorithm 4 describes the overall structure of a fmGARK step.

Aside from the particular input looping parameters, and `flagCase` which denotes the final solution algorithm, all variables can be considered to be in a global namespace for the purpose of the algorithms in this work. If a local namespace is required, this is clarified by specifying input variables or flags such as `flagCase = 1, i_in := i, k_in := k`. In summary, Algorithm 2 describes how a fast step update uses data from slow function values with the coefficients from a particular  $\mathbf{A}^{\{f,s,i\}}$ . This fast sub-step function takes as meaningful inputs  $i$ , which tells the function which slow stage loop it's being called from,  $k$  which tells the

---

**Algorithm 2:** Fast Substep Showing base methods for MIS  $\mathbf{A}^{\{f,f\}}$  (Corresponds to  $i$ th outer stage,  $k$ th subcycle out of  $n_i$  subcycles)

---

**Data:**  $\mathbf{f}_{1,\dots,s^{\{I\}}}^{\{f,i\}}, \mathbf{f}_{1,\dots,s^{\{O\}}}^{\{s\}}, h, t = t_n, \hat{\mathbf{k}}_1^{\{f,i\}}, \boxed{i}, \boxed{k}, T^{\{I\}}, T^{\{O\}}$

; // i.e.  $T^{\{O\}} \implies \mathbf{A}^{\{O\}}, \mathbf{b}^{\{O\}}, \mathbf{c}^{\{O\}}, s^{\{O\}}$  and  $T^{\{I\}} \implies \mathbf{A}^{\{I\}}, \mathbf{b}^{\{I\}}, \mathbf{c}^{\{I\}}, s^{\{I\}}$

**Result:**  $\mathbf{f}_{1,\dots,s^{\{I\}}}^{\{f,i\}}$

1 for  $l \leftarrow 1$  to  $s^{\{I\}}$  do

2      $\text{tch}_l = t + \left( c_i^{\{O\}} + (c_{i+1}^{\{O\}} - c_i^{\{O\}}) \left( \frac{c_i^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) \right) h$

3 end

4  $l=1$ ;

5  $\mathbf{k}_l^{\{f,f\}} = \hat{\mathbf{k}}_1^{\{f,f\}} + \sum_{j=1}^i h \left( a_{i,j}^{\{O\}} + \left( \frac{c_i^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,j}^{\{O\}} - a_{i,j}^{\{O\}}) \right) \mathbf{f}_j^{\{s\}}$ ; // Linear

combination

6  $\mathbf{f}_l^{\{f,f\}} = f^{\{f\}} \left( \text{tch}_l, \mathbf{k}_l^{\{f,f\}} \right)$ ; // Calculate func vecs:1st fast stage

7 for  $l \leftarrow 1$  to  $s^{\{I\}}$  do

8      $\mathbf{k}_l^{\{f,f\}} = \hat{\mathbf{k}}_1^{\{f,f\}} + \sum_{j=1}^{l-1} h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (a_{l,j}^{\{I\}}) \right) \mathbf{f}_j^{\{f,f\}} +$   
 $\sum_{j=1}^i h \left( a_{i,j}^{\{O\}} + \left( \frac{c_i^{\{I\}}}{n_i} + \frac{(k-1)}{n_i} \right) (a_{i+1,j}^{\{O\}} - a_{i,j}^{\{O\}}) \right) \mathbf{f}_j^{\{s\}}$ ; // Linear comb

9      $\mathbf{f}_l^{\{f,f\}} = f^{\{f\}} \left( \text{tch}_l, \hat{\mathbf{k}}_l^{\{f,f\}} \right)$ ; // Calculate func vecs: $l$ th fast stage

10 end

---

---

**Algorithm 3:** Solution update for RFSMR
 

---

**Data:**  $\mathbf{f}_{1,\dots,s\{I\}}^{\{f,i\}}, \mathbf{f}_{1,\dots,s\{O\}}^{\{s\}}, h, \boxed{i}, \boxed{k}, \boxed{\text{flagCase}}, T^{\{I\}}, T^{\{O\}}$   
 ; // i.e.  $T^{\{O\}} \implies \mathbf{A}^{\{O\}}, \mathbf{b}^{\{O\}}, \mathbf{c}^{\{O\}}, s^{\{O\}}$  and  $T^{\{I\}} \implies \mathbf{A}^{\{I\}}, \mathbf{b}^{\{I\}}, \mathbf{c}^{\{I\}}, s^{\{I\}}$   
**Result:**  $\boxed{y}, \boxed{\hat{\mathbf{k}}_1^{\{f,i\}}}$

```

1 if flagCase == 0 ; // Incremental solution updates within macro-step
2 then
3    $\hat{\mathbf{k}}_1^{\{f,i\}} = \hat{\mathbf{k}}_1^{\{f,f\}} + \sum_{j=1}^{s^I} h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (b_j^{\{I_k\}}) \right) \mathbf{f}_j^{\{f,f\}};$ 
4 else if flagCase == 1 ; // Final solution updates
5 then
6    $y = \hat{\mathbf{k}}_1^{\{f,f\}} + \sum_{j=1}^{s^I} h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (b_j^{\{I_k\}}) \right) \mathbf{f}_j^{\{f,f\}};$ 
7    $y = y + \sum_{j=1}^{s^O} h \left( (b_j^{\{O\}}) \right) \mathbf{f}_j^{\{s\}};$ 

```

---

function which subcycle loop it's being called from, and the  $\boxed{\hat{\mathbf{k}}_1^{\{f,i\}}}$  which has accumulated the fast initial condition from the previous fast sub-steps. Algorithm 3 describes how the intermediate solution updates are accumulated within a macro-step.

### 3.4. Order Conditions for Relaxed Framework

Order considerations are dependent both on the original outer method and on the choice of how to weight the fast stage function value's contribution to the final step solution. The existing RFSMR methods are at least second order accurate, given a second order accurate inner method and a second order accurate outer method. Equations (2.2)-(2.9) which describe the summation form of the order conditions can be applied to analyzing the order of the fmGARK methods. Conceptually, we will consider separately the row-sum conditions, the conditions with  $\mathbf{b}^{\{f\}}$ , and the conditions for  $\mathbf{b}^{\{s\}}$ . The row-sum conditions, or internal consistency conditions were addressed earlier when considering the fmGARK structure. The linear system form of the order conditions is described generally in Equation (2.18) and specifically for these types of structures in Equation (2.52). When discussing

---

**Algorithm 4:** fmGARK macro-step
 

---

**Data:**  $f^{\{f\}}, f^{\{s\}}, h, t = t_n, \mathbf{y} = \mathbf{y}_n, T^{I_n}, T^O$  ; // i.e.  $T^O \implies \mathbf{A}^{\{O\}}, \mathbf{b}^{\{O\}}, \mathbf{c}^{\{O\}}, s^{\{O\}}$   
**Result:**  $\mathbf{y} = \mathbf{y}_{n+1}$

- 1 **for**  $l \leftarrow 1$  **to**  $s^{\{O\}}$  **do**
- 2 |  $\text{tch}_l^s = t + c_l^O h$
- 3 **end**
- 4  $i=1; k=1; l=1;$
- 5 initialize  $\mathbf{f}_{1, \dots, s^{\{I\}}}^{\{f\}}, \mathbf{f}_{1, \dots, s^{\{O\}}}^{\{s\}}$  to zero vectors;
- 6 initialize  $\hat{\mathbf{k}}_1^{\{f, f\}} = y, \mathbf{k}_1^{\{s\}} = y;$
- 7 Update  $\mathbf{k}_i^{\{s\}} = \mathbf{k}_1^{\{f, i\}} + \sum_{l=1}^{i-1} ha_{il}^{\{s, s\}} \mathbf{f}_l^{\{s\}};$
- 8 Calculate  $\mathbf{f}_i^{\{s\}} = f^{\{s\}}(\text{tch}_i^s, \mathbf{k}_i^{\{s\}});$
- 9 Algorithm 3-6 flagCase = -1,  $i := i, k := k$  ; // Update  $y$
- 10 Algorithm 2 ; // Update  $\mathbf{f}_{1, \dots, s^{\{I\}}}^{\{f, f\}}$  for  $k$ th substep
- 11 **for**  $k \leftarrow 2$  **to**  $n_i$  **do**
- 12 | Algorithm 3-6 flagCase = 0,  $i_{in} := i, k_{in} := k - 1$  ; // Update  $y, \hat{\mathbf{k}}_1^{\{f, f\}}$
- 13 | Algorithm 2 ; // Update  $\mathbf{f}_{1, \dots, s^{\{I\}}}^{\{f, f\}}$  for  $k$ th substep
- 14 **end**
- 15 **for**  $i \leftarrow 2$  **to**  $s^{\{O\}}$  **do**
- 16 | Algorithm 3-6 flagCase = 0,  $i_{in} := i - 1, k_{in} := n_i$  ; // Update  $y, \hat{\mathbf{k}}_1^{\{f, f\}}$
- 17 | Update  $\mathbf{k}_i^{\{s\}} = \hat{\mathbf{k}}_1^{\{f, i\}} + \sum_{l=1}^{i-1} ha_{il}^{\{s, s\}} \mathbf{f}_l^{\{s\}};$
- 18 | Calculate  $\mathbf{f}_i^{\{s\}} = f^{\{s\}}(\text{tch}_i^s, \mathbf{k}_i^{\{s\}});$
- 19 |  $k=1; l=1;$
- 20 | Algorithm 2 ; // Update  $\mathbf{f}_{1, \dots, s^{\{I\}}}^{\{f, f\}}$  for  $k$ th substep
- 21 | Algorithm 3-6 flagCase = -1,  $i_{in} := i, k_{in} := k$  ; // Update  $y$
- 22 | **for**  $k \leftarrow 2$  **to**  $n_i$  **do**
- 23 | Algorithm 3-6 flagCase = 0,  $i_{in} := i, k_{in} := k - 1$  ; // Update  $y, \hat{\mathbf{k}}_1^{\{f, f\}}$
- 24 | Algorithm 2 ; // Update  $\mathbf{f}_{1, \dots, s^{\{I\}}}^{\{f, f\}}$  for  $k$ th substep
- 25 | **end**
- 26 **end**
- 27 Algorithm 3-6 flagCase = 1,  $i_{in} := i, k_{in} := k$  ; // Update  $y, \tilde{y}$

---



fmGARK order conditions, it is helpful to draw on analysis found in Sandu and Günther's paper on Multirate GARK [21].

The fmGARK structure draws from the theory behind MIS methods. We will show here that this means that the slow order conditions Equations (2.38)-(2.42) are automatically satisfied if the base inner method is at least second order and the base outer method is at least third order. This is accomplished by leveraging certain assumed properties of the inner base method, such as  $\sum_j b_j^{\{I\}} = 1$ ,  $\sum_j b_j^{\{I\}} c_j^{\{I\}} = \frac{1}{2}$  and  $\sum_j b_j^{\{O\}} [c_j^{\{O\}}]^2 = \frac{1}{3}$ . As mentioned in Section 3.2, the row-sum conditions, also called the internal consistency conditions are automatically satisfied by the fmGARK structure, which simplifies the order conditions. Equations (2.38)-(2.41) follow directly from the outer base method being third order. For the overall method to be third-order, we require two coupling conditions be satisfied:  $b^{\{s\}\top} A^{\{s,f\}} \mathbf{c}^{\{f\}} = \frac{1}{6}$  and  $b^{\{f\}\top} A^{\{f,s\}} \mathbf{c}^{\{s\}} = \frac{1}{6}$ . By matrix-vector multiplication, we can show that given our assumptions the first condition is satisfied automatically. The second condition may be addressed based upon a specific choice of  $b^{\{f\}\top}$ . Equation (3.25) demonstrates by substitution how the fmGARK structured table automatically satisfies the third order slow coupling condition [20].

$$b^{\{s\}} \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}} = \sum_{i=1}^{s^{\{O\}}} b_i^{\{O\}} \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) (\mathbf{b}_j^I)^\top (c_j^O \mathbf{1} + (c_{j+1}^O - c_j^O) \mathbf{c}_j^I) \quad (3.16)$$

$$= \sum_{i=1}^{s^{\{O\}}} b_i^{\{O\}} \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( c_j^O (\mathbf{b}_j^I)^\top \mathbf{1} + (c_{j+1}^O - c_j^O) (\mathbf{b}_j^I)^\top \mathbf{c}_j^I \right) \quad (3.17)$$

$$= \sum_{i=1}^{s^{\{O\}}} b_i^{\{O\}} \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( c_j^O * 1 + (c_{j+1}^O - c_j^O) \frac{1}{2} \right) \quad (3.18)$$

$$= \sum_{i=1}^{s^{\{O\}}} b_i^{\{O\}} \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( \frac{1}{2} (c_{j+1}^O + c_j^O) \right) \quad (3.19)$$

$$= \frac{1}{2} \sum_{i=1}^{s^{\{O\}}} b_i^{\{O\}} \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) (c_{j+1}^O + c_j^O) \quad (3.20)$$

$$= \frac{1}{2} \sum_{i=1}^{s^{\{O\}}} b_i^{\{O\}} \sum_{j=1}^{i-1} [c_{j+1}^O]^2 - [c_j^O]^2 = \frac{1}{2} \sum_{i=1}^{s^{\{O\}}} b_i^{\{O\}} \left[ \sum_{j=1}^{i-1} [c_{j+1}^O]^2 - [c_j^O]^2 \right] \quad (3.21)$$

$$= \frac{1}{2} \sum_{i=1}^{s^{O}} b_i^{\{O\}} \left[ \sum_{j=2}^i [c_j^O]^2 - \sum_{j=1}^{i-1} [c_j^O]^2 \right] \quad (3.22)$$

$$= \frac{1}{2} \sum_{i=1}^{s^{O}} b_i^{\{O\}} \left[ [c_i^O]^2 - [c_1^O]^2 \right] \quad (3.23)$$

$$= \frac{1}{2} \sum_{i=1}^{s^{O}} b_i^{\{O\}} [c_i^O]^2 \quad (3.24)$$

$$= \frac{1}{2} \left( \frac{1}{3} \right) = \frac{1}{6} \quad (3.25)$$

Note that Equation (3.24) implies that  $\sum_k \mathbf{A}_{ik}^{\{s,f\}} \mathbf{c}_k^{\{f\}} = \frac{1}{2} [c_i^O]^2 = \frac{1}{2} [c_i^{\{s\}}]^2$ . We can show that the fourth-order slow conditions represented by Equations (2.43)-(2.50) can be similarly satisfied given a third-order inner base method and a fourth-order outer base method. The fourth-order slow condition represented by Equation (2.51) will require an additional condition on the outer base method to be satisfied, and therefore will be discussed at the end. Equations (2.43), (2.44), (2.46), and (2.48) follow directly from assuming the outer base method is fourth-order. Given our assumed method orders, there are 4 fourth-order coupling conditions which include  $(\mathbf{b}^{\{s\}})^\top$ , that are automatically satisfied, which were stated previously as Equations (2.45), (2.47), (2.49), and (2.50). These are as follows:

$$\begin{aligned} (\mathbf{b}^{\{s\}} \mathbf{c}^{\{s\}})^\top \mathbf{A}^{\{s,f\}} \mathbf{c}^{\{f\}} &= \sum_{i=1}^s \sum_k \left( \mathbf{b}_i^{\{s\}} \mathbf{c}_i^{\{s\}} \right) \mathbf{A}_{ik}^{\{s,f\}} \mathbf{c}_k^{\{f\}} \\ &= \sum_{i=1}^s \left( \mathbf{b}_i^{\{s\}} \mathbf{c}_i^{\{s\}} \right) \frac{1}{2} \left( \mathbf{c}_i^{\{s\}} \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^s \mathbf{b}_i^{\{s\}} \left( \mathbf{c}_i^{\{s\}} \right)^3 \\ &= \frac{1}{2} \left( \frac{1}{4} \right) \text{ Since the outer method is fourth-order} \end{aligned}$$

$$\begin{aligned}
\text{Eq. (2.42)} &\implies (\mathbf{b}^{\{s\}})^\top \mathbf{A}^{\{s,f\}} (\mathbf{c}^{\{f\}})^2 \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \sum_{k=1}^{s_{I_j}} b_k^{I_j} (c_j^O + (c_{j+1}^O - c_j^O) c_k^{I_j})^2 \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \sum_{k=1}^{s_{I_j}} b_k^{I_j} \left( (1 - c_k^{I_j}) c_j^O + c_{j+1}^O c_k^{I_j} \right)^2 \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \sum_{k=1}^{s_{I_j}} b_k^{I_j} \left( (1 - c_k^{I_j})^2 (c_j^O)^2 + 2(1 - c_k^{I_j}) c_j^O c_{j+1}^O c_k^{I_j} + (c_{j+1}^O c_k^{I_j})^2 \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \sum_{k=1}^{s_{I_j}} b_k^{I_j} \left( (1 - c_k^{I_j})^2 (c_j^O)^2 + 2c_j^O c_{j+1}^O c_k^{I_j} - 2c_j^O c_{j+1}^O (c_k^{I_j})^2 + (c_{j+1}^O c_k^{I_j})^2 \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \sum_{k=1}^{s_{I_j}} \left( b_k^{I_j} (c_j^O)^2 - 2(c_j^O)^2 b_k^{I_j} c_k^{I_j} + (c_j^O)^2 b_k^{I_j} (c_k^{I_j})^2 \right) \\
&\quad + \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \sum_{k=1}^{s_{I_j}} \left( 2c_j^O c_{j+1}^O b_k^{I_j} c_k^{I_j} - 2c_j^O c_{j+1}^O b_k^{I_j} (c_k^{I_j})^2 + (c_{j+1}^O)^2 b_k^{I_j} (c_k^{I_j})^2 \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( (c_j^O)^2 \sum_{k=1}^{s_{I_j}} b_k^{I_j} - 2(c_j^O)^2 \sum_{k=1}^{s_{I_j}} b_k^{I_j} c_k^{I_j} + (c_j^O)^2 \sum_{k=1}^{s_{I_j}} b_k^{I_j} (c_k^{I_j})^2 \right) \\
&\quad + \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( 2c_j^O c_{j+1}^O \sum_{k=1}^{s_{I_j}} b_k^{I_j} c_k^{I_j} - 2c_j^O c_{j+1}^O \sum_{k=1}^{s_{I_j}} b_k^{I_j} (c_k^{I_j})^2 + (c_{j+1}^O)^2 \sum_{k=1}^{s_{I_j}} b_k^{I_j} (c_k^{I_j})^2 \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( (c_j^O)^2 - 2(c_j^O)^2 \frac{1}{2} + (c_j^O)^2 \frac{1}{3} + 2c_j^O c_{j+1}^O \frac{1}{2} - 2c_j^O c_{j+1}^O \frac{1}{3} + (c_{j+1}^O)^2 \frac{1}{3} \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( \cancel{(c_j^O)^2} - 2 \cancel{(c_j^O)^2} \frac{1}{2} + (c_j^O)^2 \frac{1}{3} + c_j^O c_{j+1}^O - \frac{2}{3} c_j^O c_{j+1}^O + \frac{1}{3} (c_{j+1}^O)^2 \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} (c_{j+1}^O - c_j^O) \left( \frac{1}{3} (c_j^O)^2 + \frac{1}{3} c_j^O c_{j+1}^O + \frac{1}{3} (c_{j+1}^O)^3 \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} \left( \frac{1}{3} (c_j^O)^2 c_{j+1}^O + \frac{1}{3} c_j^O c_{j+1}^O c_{j+1}^O + \frac{1}{3} (c_{j+1}^O)^3 \right) - \left( \frac{1}{3} (c_j^O)^2 c_j^O + \frac{1}{3} c_j^O c_{j+1}^O c_j^O + \frac{1}{3} (c_{j+1}^O)^2 c_j^O \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \sum_{j=1}^{i-1} \left( \cancel{\frac{1}{3} (c_j^O)^2 c_{j+1}^O} + \cancel{\frac{1}{3} c_j^O c_{j+1}^O c_{j+1}^O} + \frac{1}{3} (c_{j+1}^O)^3 \right) - \left( \frac{1}{3} (c_j^O)^2 c_j^O + \cancel{\frac{1}{3} c_j^O c_{j+1}^O c_j^O} + \cancel{\frac{1}{3} (c_{j+1}^O)^2 c_j^O} \right) \\
&= \sum_{i=1}^s b_i^{\{s\}} \tau \left[ \sum_{j=1}^{i-1} \frac{1}{3} (c_{j+1}^O)^2 - \frac{1}{3} (c_j^O)^2 c_j^O \right] \\
&= \sum_{i=1}^s b_i^O \left[ \sum_{j=1}^{i-1} \frac{1}{3} (c_{j+1}^O)^3 - \frac{1}{3} (c_j^O)^2 c_j^O \right] \\
&= \sum_{i=1}^s b_i^O \left[ \sum_{j=2}^i \frac{1}{3} (c_j^O)^3 - \sum_{j=1}^{i-1} \frac{1}{3} (c_j^O)^2 c_j^O \right] \\
&= \sum_{i=1}^s b_i^O \left[ \frac{1}{3} (c_i^O)^3 - \cancel{\frac{1}{3} (c_1^O)^3} \right] \\
&= \frac{1}{3} \sum_{i=1}^s b_i^O (c_i^O)^3 = \frac{1}{3} \left( \frac{1}{4} \right) = \frac{1}{12}
\end{aligned}$$

Some order conditions follow from the proof for Equation (2.42) directly.

$$\begin{aligned}
b^{\{s\}} \tau A^{\{s,s\}} A^{\{s,f\}} \mathbf{c}^{\{f\}} &= \sum_{i=1}^s b_i^{\{s\}} \tau A_{ij}^{\{s,s\}} \frac{1}{2} (\mathbf{c}_j^{\{s\}})^2 \\
&= \frac{1}{2} b^{\{s\}} \tau A^{\{s,s\}} (\mathbf{c}_j^{\{s\}})^2 \\
&= \frac{1}{2} \frac{1}{12} = \frac{1}{24}
\end{aligned}$$

Final algebraic steps for showing Eq. (2.50) is automatically satisfied for the fmGARK structure are omitted.

$$\text{Eq. (2.50)} \implies b^{\{s\}\tau} A^{\{s,f\}} A^{\{f,f\}} \mathbf{c}^{\{f\}}$$

$$\begin{aligned}
&= b^{\{s\}\tau} A^{\{s,f\}} \begin{bmatrix} c_2^O A_1^I \\ c_2^O \mathbf{b}_1^I & (c_3^O - c_2^O) A_2^I \\ \vdots & \\ (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top & \\ \vdots & \vdots & (c_4^O - c_3^O) A_3^I \\ \vdots & \vdots & \\ (c_4^O - c_3^O) (\mathbf{b}_3^I)^\top & \vdots & (1 - c_4^O) A_4^I \\ c_2^O (\mathbf{b}_1^I)^\top & (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top & (c_4^O - c_3^O) (\mathbf{b}_3^I)^\top \end{bmatrix} * \mathbf{c}^{\{f\}} \\
&= b^{\{s\}\tau} \begin{bmatrix} \mathbf{0} & \dots & \dots & \mathbf{0} \\ c_2^O (\mathbf{b}_1^I)^\top & \mathbf{0} & \dots & \mathbf{0} \\ c_2^O (\mathbf{b}_1^I)^\top & (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top & \mathbf{0} & \mathbf{0} \\ c_2^O (\mathbf{b}_1^I)^\top & (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top & (c_4^O - c_3^O) (\mathbf{b}_3^I)^\top & \mathbf{0} \end{bmatrix} \\
&\quad \times \begin{bmatrix} c_2^O A_1^I c_2^O \mathbf{c}_1^I \\ c_2^O (\mathbf{b}_1^I)^\top c_2^O \mathbf{c}_1^I + (c_3^O - c_2^O) A_2^I (c_2^O \mathbf{1} + (c_3^O - c_2^O) \mathbf{c}_2^I) \\ c_2^O (\mathbf{b}_1^I)^\top c_2^O \mathbf{c}_1^I + (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top (c_2^O \mathbf{1} + (c_3^O - c_2^O) \mathbf{c}_2^I) + (c_4^O - c_3^O) A_3^I (c_3^O \mathbf{1} + (c_4^O - c_3^O) \mathbf{c}_3^I) \\ c_2^O (\mathbf{b}_1^I)^\top c_2^O \mathbf{c}_1^I + (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top (c_2^O \mathbf{1} + (c_3^O - c_2^O) \mathbf{c}_2^I) + (c_4^O - c_3^O) (\mathbf{b}_3^I)^\top (c_3^O \mathbf{1} + (c_4^O - c_3^O) \mathbf{c}_3^I) + (1 - c_4^O) A_4^I (c_4^O \mathbf{1} + (1 - c_4^O) \mathbf{c}_4^I) \end{bmatrix} \\
&= b^{\{s\}\tau} \begin{bmatrix} \mathbf{0} & \dots & \dots & \mathbf{0} \\ c_2^O (\mathbf{b}_1^I)^\top & \mathbf{0} & \dots & \mathbf{0} \\ c_2^O (\mathbf{b}_1^I)^\top & (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top & \mathbf{0} & \mathbf{0} \\ c_2^O (\mathbf{b}_1^I)^\top & (c_3^O - c_2^O) (\mathbf{b}_2^I)^\top & (c_4^O - c_3^O) (\mathbf{b}_3^I)^\top & \mathbf{0} \end{bmatrix} \\
&\quad \times \begin{bmatrix} (c_2^O)^2 A_1^I \mathbf{c}_1^I \\ \frac{1}{2} (c_2^O)^2 \mathbf{1} + (c_3^O - c_2^O) (c_2^O \mathbf{c}_2^I + (c_3^O - c_2^O) A_2^I \mathbf{c}_2^I) \\ \frac{1}{2} (c_2^O)^2 \mathbf{1} + \frac{1}{2} (c_3^O - c_2^O) (c_2^O + c_3^O) \mathbf{1} + (c_4^O - c_3^O) (c_3^O \mathbf{c}_3^I + (c_4^O - c_3^O) A_3^I \mathbf{c}_3^I) \\ \frac{1}{2} (c_2^O)^2 \mathbf{1} + \frac{1}{2} (c_3^O - c_2^O) (c_2^O + c_3^O) \mathbf{1} + \frac{1}{2} (c_4^O - c_3^O) (c_3^O + c_4^O) \mathbf{1} + (1 - c_4^O) (c_4^O \mathbf{c}_4^I + (1 - c_4^O) A_4^I \mathbf{c}_4^I) \end{bmatrix} \\
&= b^{\{s\}\tau} \begin{bmatrix} \mathbf{0} \\ (c_2^O)^3 \frac{1}{6} \\ \frac{1}{6} (c_3^O)^3 \\ \frac{1}{6} (c_4^O)^3 \end{bmatrix} = \frac{1}{6} b^{\{s\}\tau} (c^{\{s\}})^3 = \frac{1}{6} \frac{1}{4} = \frac{1}{24} \text{ because outer method is fourth order and inner method is at least third order}
\end{aligned}$$

Finally, we consider equation for satisfying the coupling condition Equation (2.51) can be simplified for a 4-stage outer method as follows.

$$b^{s\top} A^{s,f} A^{f,s} c^s = b_2^O \left( \frac{c_2^O}{2} (\mathbf{e}_2^\top) A^O c^O \right) + b_3^O \left( \left( \frac{c_2^O}{2} (\mathbf{e}_2^\top) A^O + (c_3^O - c_2^O) (\mathbf{e}_2^\top A^O + \frac{1}{2} (\mathbf{e}_3^\top - \mathbf{e}_2^\top) A^O) \right) c^O \right) \quad (3.26)$$

$$+ b_4^O \left( \left( \frac{c_2^O}{2} (\mathbf{e}_2^\top) A^O + (c_3^O - c_2^O) (\mathbf{e}_2^\top A^O + \frac{1}{2} (\mathbf{e}_3^\top - \mathbf{e}_2^\top) A^O) + (c_4^O - c_3^O) (\mathbf{e}_3^\top A^O + \frac{1}{2} (\mathbf{e}_4^\top - \mathbf{e}_3^\top) A^O) \right) c^O \right) \quad (3.27)$$

$$= \left( (b_2^O + b_3^O + b_4^O) \left( \frac{c_2^O}{2} \right) + \left( 1 - \frac{1}{2} \right) b_3^O (c_3^O - c_2^O) + \left( 1 - \frac{1}{2} \right) b_4^O (c_3^O - c_2^O) \right) (\mathbf{e}_2^\top) A^O c^O \quad (3.28)$$

$$+ \left( b_3^O (c_3^O - c_2^O) \frac{1}{2} + b_4^O (c_3^O - c_2^O) \frac{1}{2} + b_4^O (c_4^O - c_3^O) \left( 1 - \frac{1}{2} \right) \right) ((\mathbf{e}_3^\top) A^O) c^O \quad (3.29)$$

$$+ b_4^O (c_4^O - c_3^O) \left( \frac{1}{2} (\mathbf{e}_4^\top) A^O \right) c^O \quad (3.30)$$

$$= \left( (b_2^O + b_3^O + b_4^O) \left( \frac{c_2^O}{2} \right) + \left( \frac{1}{2} \right) b_3^O (c_3^O - c_2^O) + \left( \frac{1}{2} \right) b_4^O (c_3^O - c_2^O) \right) (\mathbf{e}_2^\top) A^O c^O \quad (3.31)$$

$$+ \left( b_3^O (c_3^O - c_2^O) \frac{1}{2} + b_4^O (c_3^O - c_2^O) \frac{1}{2} + b_4^O (c_4^O - c_3^O) \left( \frac{1}{2} \right) \right) ((\mathbf{e}_3^\top) A^O) c^O \quad (3.32)$$

$$+ b_4^O (c_4^O - c_3^O) \left( \frac{1}{2} (\mathbf{e}_4^\top) A^O \right) c^O \quad (3.33)$$

$$= \left( (b_2^O) \left( \frac{c_2^O}{2} \right) + \left( \frac{1}{2} \right) b_3^O (c_3^O) + \left( \frac{1}{2} \right) b_4^O (c_3^O) \right) (\mathbf{e}_2^\top) A^O c^O \quad (3.34)$$

$$+ \left( b_3^O (c_3^O - c_2^O) \frac{1}{2} + b_4^O (-c_2^O) \frac{1}{2} + b_4^O (c_4^O) \left( \frac{1}{2} \right) \right) ((\mathbf{e}_3^\top) A^O) c^O \quad (3.35)$$

$$+ b_4^O (c_4^O - c_3^O) \left( \frac{1}{2} (\mathbf{e}_4^\top) A^O \right) c^O \quad (3.36)$$

$$= \begin{bmatrix} \left( (b_2^O) \left( \frac{c_2^O}{2} \right) + \left( \frac{1}{2} \right) b_3^O (c_3^O) + \left( \frac{1}{2} \right) b_4^O (c_3^O) \right) \\ \left( b_3^O (c_3^O - c_2^O) \frac{1}{2} + b_4^O (-c_2^O) \frac{1}{2} + b_4^O (c_4^O) \left( \frac{1}{2} \right) \right) \\ b_4^O (c_4^O - c_3^O) \left( \frac{1}{2} (\mathbf{e}_4^\top) A^O \right) c^O \end{bmatrix}^{\top} A^O c^O \quad (3.37)$$

$$= \begin{bmatrix} \left( (b_2^O) \left( \frac{c_2^O}{2} \right) + \left( \frac{1}{2} \right) b_3^O (c_3^O) + \left( \frac{1}{2} \right) b_4^O (c_3^O) \right) \\ \left( b_3^O (c_3^O - c_2^O) \frac{1}{2} + b_4^O (-c_2^O) \frac{1}{2} + b_4^O (c_4^O) \left( \frac{1}{2} \right) \right) \\ b_4^O (c_4^O - c_3^O) \left( \frac{1}{2} \right) \end{bmatrix}^{\top} A^O c^O \quad (3.38)$$

$$= \frac{1}{2} \begin{bmatrix} 0 \\ \left( (b_2^O) (c_2^O) + b_3^O (c_3^O) + b_4^O (c_3^O) \right) \\ \left( b_3^O (c_3^O - c_2^O) + b_4^O (-c_2^O) + b_4^O (c_4^O) \right) \\ b_4^O (c_4^O - c_3^O) \end{bmatrix}^{\top} A^O c^O \quad (3.39)$$

This simplification of the slow fmGARK order condition which is not automatically satisfied gives us a specific condition on the outer method which forces the not-automatically satisfied condition regarding  $\mathbf{b}^{\{s\}}$  to be satisfied. Now that we have introduced the fmGARK structure and given a basic idea of how the RFSMR can be implemented in the fmGARK context, we will discuss in the following chapter the development and properties of our new specific fmGARK methods.

## Chapter 4

### Proposed Relaxed MIS methods and comparative fmGARK methods

We will distinguish these different types of fmGARK methods by choosing  $\hat{\mathbf{b}}^{\{f\}}$  to target a specific objective. This choice does not change the intermediate stage solutions within the fmGARK step, but does alter how those stage solutions are glued together to generate the final step solution.

In order to uniquely determine a fmGARK method, we must choose the base inner method and the base outer method, as well as make a choice about the method's targeted time-scale separation. These choices determine the structure detailed in Section 3.2. For our implementations, we relied on Butcher's derivation of families of explicit 3rd order and 4th order base Runge-Kutta methods [5]. The form of these 3-stage and 4-stage Runge-Kutta methods were derived in another form by Runge and König in 1924 [40]. According to Hairer, the original form may be included in a publication by Kutta from 1901 [22]. These tables are reproduced below for convenience.

Explicit 2-stage RK method:

0	
$c_2$	$c_2$
	$1 - \frac{1}{2c_2} \quad \frac{1}{2c_2}$

Explicit 3-stage RK method for  $c_2 \neq 0 \neq c_3 \neq c_2 \neq \frac{2}{3}$

0		
$c_2$	$c_2$	
$c_3$	$\frac{c_3(3c_2 - 3c_2^2 - c_3)}{c_2(2 - 3c_2)}$	$\frac{c_3(c_3 - c_2)}{c_2(2 - 3c_2)}$
	$\frac{-3c_3 + 6c_2c_3 + 2 - 3c_2}{6c_2c_3}$	$\frac{3c_3 - 2}{6c_2(c_3 - c_2)} \quad \frac{2 - 3c_2}{6c_3(c_3 - c_2)}$

Explicit 3-stage RK method for  $c_2 = c_3 = \frac{2}{3}, b_3 \neq 0$ :

0		
$\frac{2}{3}$	$\frac{2}{3}$	
$\frac{2}{3}$	$\frac{2}{3} - \frac{1}{4b_3}$	$\frac{1}{4b_3}$
	$\frac{1}{4}$	$\frac{3}{4} - b_3 \quad b_3$

Explicit 4-stage RK method satisfying condI:

0				
1 - c <sub>3</sub>		1 - c <sub>3</sub>		
c <sub>3</sub>		$\frac{c_3(1-2c_3)}{2(1-c_3)}$	$\frac{c_3}{2(1-c_3)}$	
1		$\frac{12c_3^3 - 24c_3^2 + 17c_3 - 4}{2(1-c_3)(6c_3 - 1 - 6c_3^2)}$	$\frac{c_3(1-2c_3)}{2(1-c_3)(6c_3 - 1 - 6c_3^2)}$	$\frac{1-c_3}{(6c_3 - 1 - 6c_3^2)}$
		$\frac{6c_3 - 1 - 6c_3^2}{12c_3(1-c_3)}$	$\frac{1}{12c_3(1-c_3)}$	$\frac{1}{12c_3(1-c_3)}$

Explicit 4-stage RK method satisfying condII:

0				
c <sub>2</sub>		c <sub>2</sub>		
$\frac{1}{2}$		$\frac{1}{2} - \frac{1}{8c_2}$	$\frac{1}{8c_2}$	
1		$\frac{1}{2c_2} - 1$	$-\frac{1}{2c_2}$	2
		$\frac{1}{6}$	0	$\frac{2}{3}$

Explicit 4-stage RK method satisfying condV:

0				
$\frac{1}{2}$		$\frac{1}{2}$		
$\frac{1}{2}$		$\frac{1}{2} - \frac{1}{6b_3}$	$\frac{1}{6b_3}$	
1		0	1 - 3b <sub>3</sub>	3b <sub>3</sub>
		$\frac{1}{6}$	$\frac{2}{3} - b_3$	$b_3$

Butcher also derives a general solution for a 3-stage third order explicit Runge-Kutta method which depends only on two free variables, c<sub>2</sub> and c<sub>3</sub>.

0			
c <sub>2</sub>		c <sub>2</sub>	
c <sub>3</sub>		$\frac{c_3(3c_2 - 3c_2^2 - c_3)}{c_2(2 - 3c_2)}$	$\frac{c_3(c_3 - c_2)}{c_2(2 - 3c_2)}$
		$\frac{-3c_3 + 6c_2c_3 + 2 - 3c_2}{6c_2c_3}$	$\frac{3c_3 - 2}{6c_2(c_3 - c_2)}$

(4.1)

Butcher also derives a general solution for a 4-stage fourth order explicit Runge-Kutta method which depends only on two free variables,  $c_2$  and  $c_3$ .

$$a_{21} = c_2, \quad (4.2)$$

$$a_{31} = \frac{c_3 (c_3 + 4c_2^2 - 3c_2)}{2c_2 (2c_2 - 1)}, \quad (4.3)$$

$$a_{32} = -\frac{c_3 (c_3 - c_2)}{2c_2 (2c_2 - 1)}, \quad (4.4)$$

$$a_{41} = \frac{-12c_3c_2^2 + 12c_3^2c_2^2 + 4c_2^2 - 6c_2 + 15c_2c_3 - 12c_3^2c_2 + 2 + 4c_3^2 - 5c_3}{2c_2c_3 (-4c_3 + 6c_3c_2 + 3 - 4c_2)}, \quad (4.5)$$

$$a_{42} = \frac{(c_2 - 1)(4c_3^2 - 5c_3 + 2 - c_2)}{2c_2 (c_3 - c_2) (-4c_3 + 6c_3c_2 + 3 - 4c_2)}, \quad (4.6)$$

$$a_{43} = -\frac{(2c_2 - 1)(c_2 - 1)(c_3 - 1)}{c_3 (c_3 - c_2) (-4c_3 + 6c_3c_2 + 3 - 4c_2)}, \quad (4.7)$$

$$b_1 = \frac{6c_3c_2 - 2c_3 - 2c_2 + 1}{12c_3c_2}, \quad (4.8)$$

$$b_2 = -\frac{(2c_3 - 1)}{12c_2 (c_2 - 1) (c_3 - c_2)}, \quad (4.9)$$

$$b_3 = \frac{(2c_2 - 1)}{12c_3 (c_2 - c_3c_2 + c_3^2 - c_3)}, \quad (4.10)$$

$$b_4 = \frac{-4c_3 + 6c_3c_2 + 3 - 4c_2}{12 (c_3 - 1) (c_2 - 1)}. \quad (4.11)$$

When specifying a method to implement, choosing the  $\hat{\mathbf{b}}^{\{f\}}$  is related to how we choose these base methods.

#### 4.1. Same strategy: RFSMR or MIS

As defined in Section 2.4 and briefly mentioned in Section 3.2, the RFSMR approach and MIS methods can be represented using the fmGARK structure we have defined. This essentially chooses  $b_p^{\{f,i\}} = b_p^I (c_{i+1}^O - c_i^O) i = 1, \dots, s^O, ; p = 1, \dots, s^I$ . This strategy for gluing together the final step solution uses the same strategy as for determining the fast stage solutions. Conceptually, it sets up fast integration problem between  $\tau_{i,1} = c_{s^O}^{\{O\}}$ , the final stage time and  $\tau_{i+1,1} = 1$ , where the final step solution is calculated. It uses a source term made out of the slow function data, such as  $r_i = \sum_{j=1}^{i-1} (a_{ij}^O - a_{i-1,j}^O) f^{\{s\}}(\mathbf{k}_j^{\{s\}})$ . The corresponding ODE setup is  $\frac{\partial \mathbf{k}^{\{f,i\}}}{\partial \tau} = \frac{1}{c_i^O - c_{i-1}^O} r_i + f^{\{f\}}(\mathbf{k}^{\{f,i\}})$ ,  $\tau \in [\tau_{i,1}, \tau_{i+1,1}]$ ,  $i =$



$2, \dots, s^O + 1$ . This is shown in pseudocode in Algorithm 3 when flagCase is equal to 1.

When considering the method order, we can draw from the fmGARK slow order conditions and from previous literature as referenced in Section 2.4 which yields Equation (2.56). Methods of this type are at worst second order methods. For an inner method which has four stages or less, methods of this type are at worst second order and at best third order.

Given a 3-stage RK method defined by Equation (4.1), Equation (4.12) gives the two choices of  $c_2$  which will force the RK method to automatically satisfy (2.28).

$$c_2 = \begin{cases} (2(c_3^4 - 5c_3^3 + \frac{21}{4}c_3^2 - \frac{11}{6}c_3 + \frac{1}{4})^{(1/2)} - c_3 + 2c_3^2 + 1)/(4c_3) \\ \text{for } c_3 \neq 0 \text{ and } c_3 \neq 2/3 \text{ and } 0 \leq 12c_3^4 - 60c_3^3 + 63c_3^2 - 22c_3 + 3 \\ \\ -(c_3 + 2(c_3^4 - 5c_3^3 + \frac{21}{4}c_3^2 - \frac{11}{6}c_3 + \frac{1}{4})^{(1/2)} - 2c_3^2 - 1)/(4c_3) \\ \text{for } c_3 \neq 0 \text{ and } 0 \leq 12c_3^4 - 60c_3^3 + 63c_3^2 - 22c_3 + 3 \end{cases} \quad (4.12)$$

A specific example of a 3-stage RK method from the literature which we will reference in future tests is the three-stage third order method where  $c = [0, 1/3, 3/4]$  originally proposed by Knoth and Wolke. This method, reproduced below as Equation (4.13) automatically satisfies (2.28). We will reference Equation (4.13) later when we test our methods.

$$\text{KW3 : } \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{3}{4} & \frac{-3}{16} & \frac{15}{16} & 0 \\ \hline & \frac{1}{6} & \frac{3}{10} & \frac{8}{15} \end{array} \quad (4.13)$$

Given a 4-stage RK method defined by Equations (4.2) -(4.11), Equation (4.14) must be satisfied in order for the RK method to automatically satisfy (2.28).

$$\begin{aligned} & -3(c_2 - 1)(6c_2^2c_3^2 - 4c_2^2c_3 - 6c_2c_3^3 + 8c_2c_3^2 - 11c_2c_3 + 6c_2 + 4c_3^3 - 7c_3^2 + 7c_3 - 3) \\ & + 2(2c_2 - 1)(4c_2 + 4c_3 - 6c_2c_3 - 3) = 0 \end{aligned} \quad (4.14)$$

The solutions to Equation (4.14) are plotted in Figure 4.1. Note that for the RFSMR and MIS methods the higher order error terms may differ based on the base methods chosen and the time-scale separation. Although our implementation pseudocode only considers the case where the inner method is explicit, it is not a necessary assumption in order for a RFSMR method which satisfies Equation (2.28) to be third-order.

## 4.2. Relaxed Multirate Infinitesimal Step Methods: preserves linear invariants

The target objective for the Relaxed Multirate Infinitesimal Step (RMIS) method is to create a fully fourth-order accurate method which is defined generically in terms of the time-scale separation  $m$ . We accomplish this by considering a variation on the so-called mass-preserving extension Schlegel’s dissertation proposes which preserves linear invariants [49]. Schlegel proposes expanding  $\mathbf{A}^{\{s,s\}}$  and  $\mathbf{b}^{\{s\}}$  by inserting redundant zero columns to force  $\mathbf{b}^{\{s\}}$  and  $\mathbf{b}^{\{f\}}$  to have the same dimension. He then proposes a general form with  $\bar{\mathbf{b}}_j^{\{s\}} = \bar{\mathbf{b}}_j^{\{f\}} = \omega \mathbf{b}_j^{\{f\}} + (1 - \omega) \mathbf{b}_j^{\{s\}}$ . Schlegel’s dissertation claims that if  $\bar{\mathbf{b}}_j^{\{s\}} = \bar{\mathbf{b}}_j^{\{f\}} = \mathbf{b}_j^{\{s\}}$  is chosen that stability is reduced, but does not provide any supporting results. Schlegel also demonstrates some examples where choosing  $\omega \neq 0$  can introduce extra computational work by requiring extra function calls of the slow function. A disadvantage of Schlegel’s approach for these methods is that his tripartite splitting which increased the efficiency of his methods is not easily extendable with the “extended partitioned methods.” Schlegel’s dissertation includes a preliminary proof that this mass preserving extension is third order if and only if the original RFSMR it is based on is third order.

We develop the Relaxed Multirate Infinitesimal Step Method after we attempt to examine the properties of  $\hat{\mathbf{b}}^{\{f\}} \mathbf{A}^{\{f,f\}} \mathbf{A}^{\{f,s\}} \mathbf{c}^{\{s\}} = \frac{1}{24}$  and whether there are any consistent features of  $\hat{\mathbf{b}}^{\{f\}}$  regardless of time-scale separation and base method selection. In order to do this, we reduce the number of non-zero elements we allow  $\hat{\mathbf{b}}^{\{f\}}$  to have while solving the order conditions as a linear system which is linear in  $\hat{\mathbf{b}}^{\{f\}}$  as in Equation (2.53) and Equation (2.54). This leads to the observation that when the stage times match for the fast and the slow method, and we use the same  $b$  weights for the final step, we can automatically solve

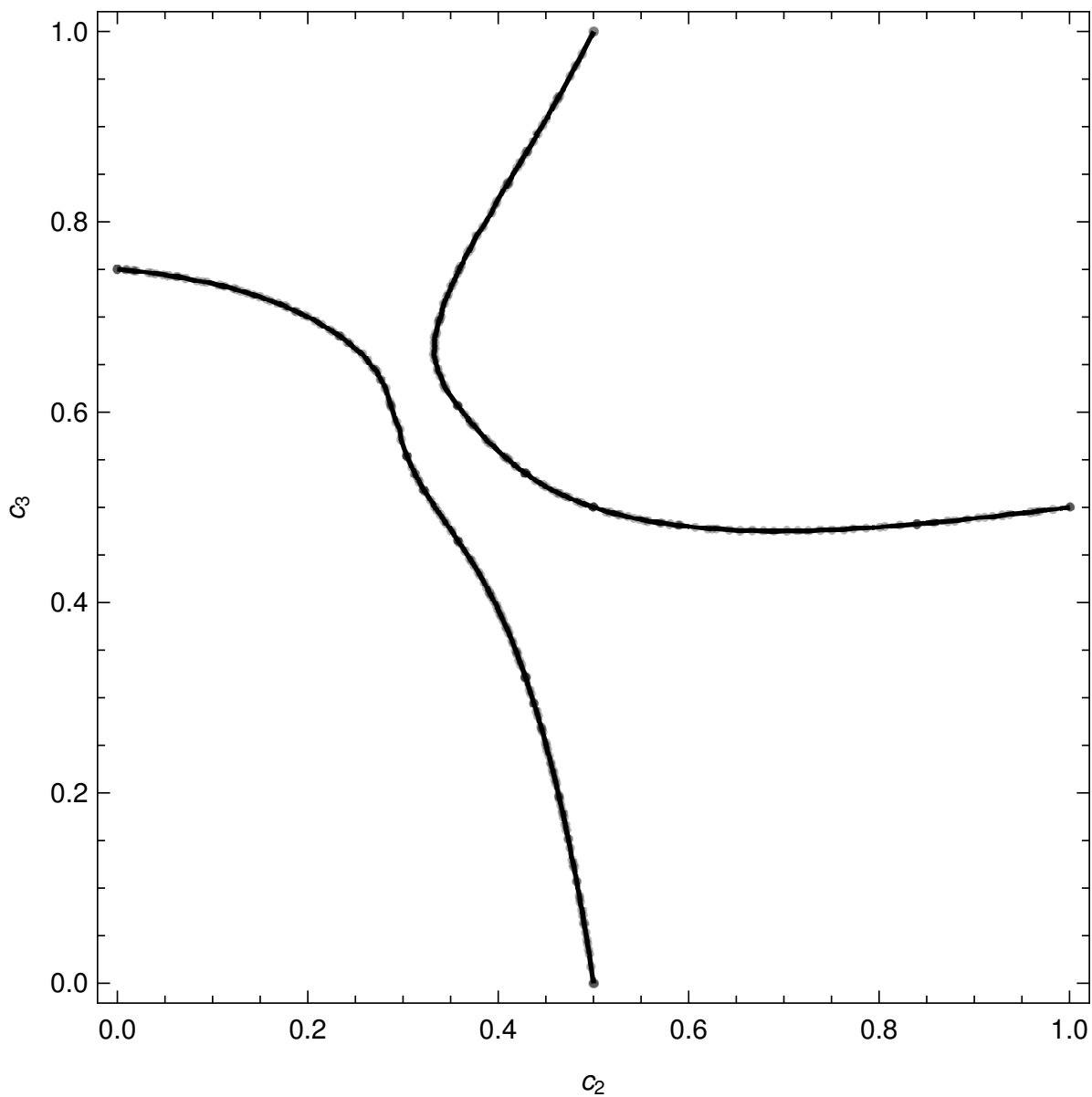


Figure 4.1. Choices of  $c_2$  and  $c_3$  on these lines result in the 3rd order RFMSR method, otherwise the method is 2nd order

the linear system of order conditions.

The RMIS method yields  $\mathbf{b}^{\{f\}}$  with the embedding  $\tilde{\mathbf{b}}^{\{f\}}$  :

$$b_p^{\{f,l\}} = \begin{cases} b_{l-1}^O & p = 1 \\ 0 & p = 2, \dots, s^{I-1} \end{cases}, \quad l = 2, \dots, s^O, s^O + 1$$

$$\tilde{b}_p^{\{f,l\}} = \begin{cases} b_p^{I s^O} (c_l^O - c_{l-1}^O) & l = 2, \dots, s^O, \\ b_p^{I s^O} (1 - c_{s^O}^O) & l = s^O + 1, \end{cases}, \quad p = 1, \dots, s^{I-1}.$$

An example of how we implement this is with the pseudocode from Section 3.3, by choosing to use Algorithm 5 instead of Algorithm 3.

---

**Algorithm 5:** Solution update for RMIS

---

**Data:**  $\mathbf{f}_{1, \dots, s^{\{I\}}}^{\{f, i\}}, \mathbf{f}_{1, \dots, s^{\{O\}}}^{\{s\}}, h, \boxed{i}, \boxed{k}, \boxed{\text{flagCase}}, T^{\{I\}}, T^O$   
; // i.e.  $T^O \implies \mathbf{A}^{\{O\}}, \mathbf{b}^{\{O\}}, \mathbf{c}^{\{O\}}, s^{\{O\}}$  and  $T^I \implies \mathbf{A}^{\{I\}}, \mathbf{b}^{\{I\}}, \mathbf{c}^{\{I\}}, s^{\{I\}}$

**Result:**  $\boxed{y}, \boxed{\tilde{y}}, \boxed{\hat{\mathbf{k}}_1^{\{f, \tilde{t}\}}}$

```

1 if flagCase == 0 ; // Incremental solution updates within macro-step
2 then
3    $\hat{\mathbf{k}}_1^{\{f, \tilde{t}\}} = \hat{\mathbf{k}}_1^{\{f, \tilde{t}\}} + \sum_{j=1}^{s^I} h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (b_j^{\{I_k\}}) \right) \mathbf{f}_j^{\{f, \tilde{t}\}};$ 
4 else if flagCase == 1 ; // Final solution updates
5 then
6    $\tilde{y} = \hat{\mathbf{k}}_1^{\{f, \tilde{t}\}} + \sum_{j=1}^{s^I} h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (b_j^{\{I_k\}}) \right) \mathbf{f}_j^{\{f, \tilde{t}\}};$ 
7    $\tilde{y} = \tilde{y} + \sum_{j=1}^{s^O} h \left( (b_j^{\{O\}}) \right) \mathbf{f}_j^{\{s\}};$ 
8    $y = y + \sum_{j=1}^{s^O} h \left( (b_j^{\{O\}}) \right) \mathbf{f}_j^{\{s\}};$ 
9 else
10  ; // Incremental solution updates occurring once per slow stage
     $y = y + h(b_i^{\{I_k\}}) \mathbf{f}_1^{\{f, \tilde{t}\}};$ 

```

---

As we mentioned previously, the fast order condition from Equation (2.37) partially motivates the design decisions in formulating the RMIS type of multirate method. For thoroughness, we now consider all the fast order conditions for this particular type of  $\mathbf{b}^{\{f\}}$ . In Section 3.4 we show the conditions required for all slow order conditions to be satisfied, including Equation (3.39). For the Relaxed Multirate Infinitesimal Step methods, the fast order conditions Equations (2.24)-(2.37) either reduce to classical conditions on the inner method which can be satisfied by original assumptions of a certain order for the inner method, or reduce to the slow order conditions on  $\mathbf{b}^{\{s\}}$ , Equations (2.38)-(2.51). We get this result for the RMIS automatically when the first stage of the inner base method is explicit since this causes  $\mathbf{b}^{\{f\}}\mathbf{A}^{\{f,f\}} = \mathbf{b}^{\{s\}}\mathbf{A}^{\{s,f\}}$  and  $\mathbf{b}^{\{f\}}\mathbf{A}^{\{f,s\}} = \mathbf{b}^{\{s\}}\mathbf{A}^{\{s,s\}}$ . For an RMIS method where the first row of  $A^{\{I\}}$  which is equal to zero,  $\mathbf{b}^{\{s\}}A^{\{s,f\}} = \mathbf{b}^{\{s\}}A^{\{s,s\}}$  and  $\mathbf{b}^{\{f\}}A^{\{f,f\}} = \mathbf{b}^{\{f\}}A^{\{f,s\}}$ . This means that if we satisfy the order conditions regarding  $\mathbf{b}^{\{s\}}$ , we force the order conditions regarding  $\mathbf{b}^{\{f\}}$  to be satisfied by construction of the RMIS method. We have not examined the fast order conditions for an RMIS method without this assumption. Therefore, without the assumption that the first stage of the inner method is explicit, we can only guarantee that an RMIS method is second order. This means that in order for an RMIS method to be fourth order, the first stage of the inner method must be explicit, the inner method must be at least third order, the outer method must be at least fourth order, and the condition derived in Equation (3.39) must be satisfied.

Given a 4-stage RK method defined by Equations (4.2) -(4.11), Equations (4.15)-(4.16) give the two choices of  $c_2$  which will force the RK method to automatically satisfy (2.51). The solutions to Equations (4.15)-(4.16) are plotted in Figure 4.2.

$$c_2 = -\frac{-6c_3^2 + \sqrt{36c_3^4 - 120c_3^3 + 80c_3^2 - 12c_3 + 1} + 2c_3 - 3}{4(3c_3 + 1)}, c_3 \in (0, 3/4] \quad (4.15)$$

$$c_2 = \frac{6c_3^2 + \sqrt{36c_3^4 - 120c_3^3 + 80c_3^2 - 12c_3 + 1} - 2c_3 + 3}{4(3c_3 + 1)}, c_3 \in (0, 3/4] \quad (4.16)$$

We generally only consider base methods with non-decreasing stage times, that is  $c_{i+1} \geq c_i$ . Listed below are the three methods based on Butcher's specific cases which satisfy the condition on the outer base method in order to generate a 4-stage fourth order RMIS method [5].

Case I					Case II					Case V				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	$\frac{1}{5}$	$\frac{1}{5}$	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{2}{3}$	$-\frac{1}{3}$	1	0	0	$\frac{1}{2}$	$-\frac{1}{8}$	$\frac{5}{8}$	0	0	$\frac{1}{2}$	$-\frac{1}{2}$	1	0	0
1	1	-1	1	0	1	$\frac{3}{2}$	$-\frac{5}{2}$	2	0	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0
	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$		$\frac{1}{6}$	0	$\frac{2}{3}$	$\frac{1}{6}$		$\frac{1}{6}$	$\frac{1}{2}$	$\frac{1}{6}$	$\frac{1}{6}$

We will reference Equation (4.17) later when we test our methods.

$$\begin{array}{c|cccc}
 & 0 & 0 & 0 & 0 \\
 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\
 3/8 - \text{Rule} : & \frac{2}{3} & -\frac{1}{3} & 1 & 0 \\
 & 1 & 1 & -1 & 1 \\
 \hline
 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8}
 \end{array} \tag{4.17}$$

Note that for the RMIS the higher order error terms will only differ due to the base methods chosen, and will not be affected by the time-scale separation.

### 4.3. Optimized methods

Rather than approaching the choice of  $\mathbf{b}^{\{f\}}$  in a general way, we can also consider optimizing for specific base methods, time scale separations, and objectives. The fmGARK order conditions are related to the local truncation error terms in a similar fashion that a

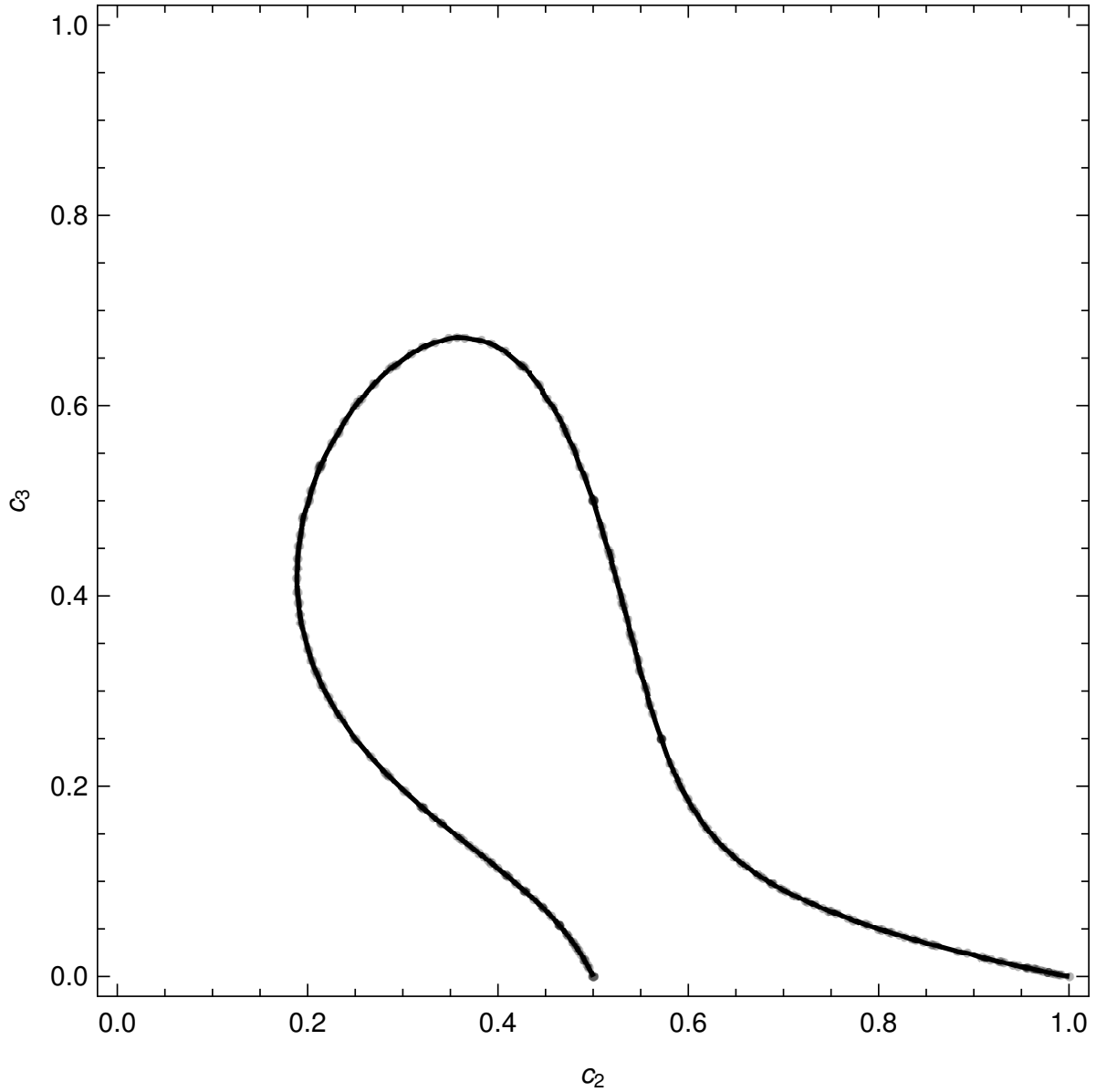


Figure 4.2. Choices of  $c_2$  and  $c_3$  on these lines result in the 4th order RMIS method, otherwise the method is 3rd order

standard RK order condition relates to the local truncation error terms. In order to approach this optimization problem, we consider representing the fmGARK order conditions as a system of equations linear in  $\mathbf{b}^{\{f\}}$ ,  $A_4\mathbf{b}^{\{f\}} = \mathbf{r}_4$  for the order conditions up to order 4 as in Equation 2.53, and  $A_5\mathbf{b}^{\{f\}} = \mathbf{r}_5$  for the 39 additional order conditions required for order 5 as in Equation 2.54. The RMIS method described in Section 4.2 is an example of the existence of a solution to the linear system formulation of the third-order conditions with  $\hat{\mathbf{b}}^{\{f\}}$ . If the conditions for a fourth-order RMIS are met, then there exists a solution to the linear system formulation of the fourth-order conditions with  $\hat{\mathbf{b}}^{\{f\}}$ . Given that we have a solution to a set of order conditions, we look for a solution to the linear system formulation of the fourth-order conditions with  $\hat{\mathbf{b}}^{\{f\}} = \mathbf{b}^{\{f\}} + W\boldsymbol{\alpha}$ ,  $\boldsymbol{\alpha} \in \mathbb{R}^{39}$ .  $W$  is a matrix whose columns are the basis vectors for  $\text{Nul}(A_4)$ . We set up a minimization problem to determine our optimized method,

$$\min_{\boldsymbol{\alpha}} \left\| \begin{bmatrix} \mathbf{r}_4 \\ \mathbf{r}_5 \end{bmatrix} - \begin{bmatrix} A_4 \\ A_5 \end{bmatrix} \mathbf{b}^{\{f\}} \right\|_2 = \min_{\boldsymbol{\alpha} \in \mathbb{R}^{39}} \left\| \begin{bmatrix} \mathbf{r}_4 \\ \mathbf{r}_5 \end{bmatrix} - \begin{bmatrix} A_4 \\ A_5 \end{bmatrix} (b + W\boldsymbol{\alpha}) \right\|_2 \quad (4.18)$$

$$= \min_{\boldsymbol{\alpha} \in \mathbb{R}^{39}} \left\| \begin{bmatrix} \mathbf{r}_4 \\ \mathbf{r}_5 \end{bmatrix} - \begin{bmatrix} A_4 \\ A_5 \end{bmatrix} \mathbf{b}^{\{f\}} - \begin{bmatrix} A_4 \\ A_5 \end{bmatrix} W\boldsymbol{\alpha} \right\|_2 \quad (4.19)$$

$$= \min_{\boldsymbol{\alpha} \in \mathbb{R}^{39}} \left\| \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_5 \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ A_5\mathbf{b}^{\{f\}} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ A_5W\boldsymbol{\alpha} \end{bmatrix} \right\|_2 \quad (4.20)$$

$$= \min_{\boldsymbol{\alpha} \in \mathbb{R}^{39}} \left\| \mathbf{r}_5 - A_5\mathbf{b}^{\{f\}} - A_5W\boldsymbol{\alpha} \right\|_2 \quad (4.21)$$

We developed methods using Matlab's built in optimization methods in order to minimize the fifth-order error terms. We investigate whether weighting optimization so that the linear system accounted for differences between the elementary weights. Our objective function (4.21) is wrapped with another function which ensures we only optimize over the bf components which are non-zero for the RFSMR. Our initial guess is a vector of zeros, we use `fminsearch` on Matlab version R2015b with the following options set: `TolFun=1e-9, TolX=1e-`



9,MaxIter=250000,MaxFunEvals=250000. The iteration limiting is set to a high value so that the optimization ends when a local minimum is found. Regardless of how we set up our optimization problem for choosing  $\mathbf{b}_p^{\{f\}}$ , we developed an algorithm which is aware of how those  $\mathbf{b}_p^{\{f\}}$  line up with each fast sub-step.

An example of how this could be implemented with the pseudocode from Section 3.3 is by choosing to use Algorithm 6 instead of Algorithm 3.

---

**Algorithm 6:** Solution update for fmGARK

---

**Data:**  $\mathbf{f}_{1,\dots,s^I}^{\{f,i\}}, \mathbf{f}_{1,\dots,s^O}^{\{s\}}, h, \boxed{i}, \boxed{k}, \boxed{\text{flagCase}}, \boxed{\mathbf{b}^{\{f\}}}, T^{\{I\}}, T^O$   
; // i.e.  $T^O \implies \mathbf{A}^{\{O\}}, \mathbf{b}^{\{O\}}, \mathbf{c}^{\{O\}}, s^{\{O\}}$  and  $T^I \implies \mathbf{A}^{\{I\}}, \mathbf{b}^{\{I\}}, \mathbf{c}^{\{I\}}, s^{\{I\}}$

**Result:**  $\boxed{y}, \boxed{\tilde{y}}, \boxed{\hat{\mathbf{k}}_1^{\{f,\tilde{f}\}}}$

1 if flagCase == 0 ; // Incremental solution updates within macro-step

2 then

3  $\hat{\mathbf{k}}_1^{\{f,\tilde{f}\}} = \hat{\mathbf{k}}_1^{\{f,\tilde{f}\}} + \sum_{j=1}^{s^I} h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (b_j^{\{I_k\}}) \right) \mathbf{f}_j^{\{f,\tilde{f}\}};$

4  $p = (i - 1) * n_i * s^I + (k - 1) * s^I;$

5  $y = y + \sum_{j=1}^{s^I} h(b_{p+j}^{\{f\}}) \mathbf{f}_j^{\{f,\tilde{f}\}};$

6 else if flagCase == 1 ; // Final solution updates

7 then

8  $\tilde{y} = \hat{\mathbf{k}}_1^{\{f,\tilde{f}\}} + \sum_{j=1}^{s^I} h \left( \frac{(c_{i+1}^{\{O\}} - c_i^{\{O\}})}{n_i} (b_j^{\{I_k\}}) \right) \mathbf{f}_j^{\{f,\tilde{f}\}};$

9  $\tilde{y} = \tilde{y} + \sum_{j=1}^{s^O} h \left( (b_j^{\{O\}}) \right) \mathbf{f}_j^{\{s\}};$

10  $p = (i - 1) * n_i * s^I + (k - 1) * s^I;$

11  $y = y + \sum_{j=1}^{s^I} h(b_{p+j-s^I}^{\{f\}}) \mathbf{f}_j^{\{f,\tilde{f}\}};$

12  $y = y + \sum_{j=1}^{s^O} h \left( (b_j^{\{O\}}) \right) \mathbf{f}_j^{\{s\}};$

---

#### 4.4. Comparisons

We can compare the restrictions for the outer method on the RFSMR and on the RMIS in Figure 4.3. This allows a visual confirmation that there are two 4-stage explicit RK methods which yield a 3rd order RFSMR and a 4th order RMIS. We can also compare the types of coefficients available if we require  $c_3 \geq c_2$ . A strong contribution of the new fmGARK structure, and especially of the RMIS methods we developed is that the order of accuracy will be at least as good, and sometimes better than the existing RFSMR and MIS methods. Another benefit of these newly developed methods is the opportunity to use the RFSMR as an embedding for estimation of the local truncation error. Having an approximation of the error allows us to evaluate how to choose our step size as we proceed through the integration. It also can allow for adaptive splittings, especially for component-wise splittings where certain components can be marked as fast or slow based on whether the current time-scale separation target of the method is appropriate for the amount of error found.

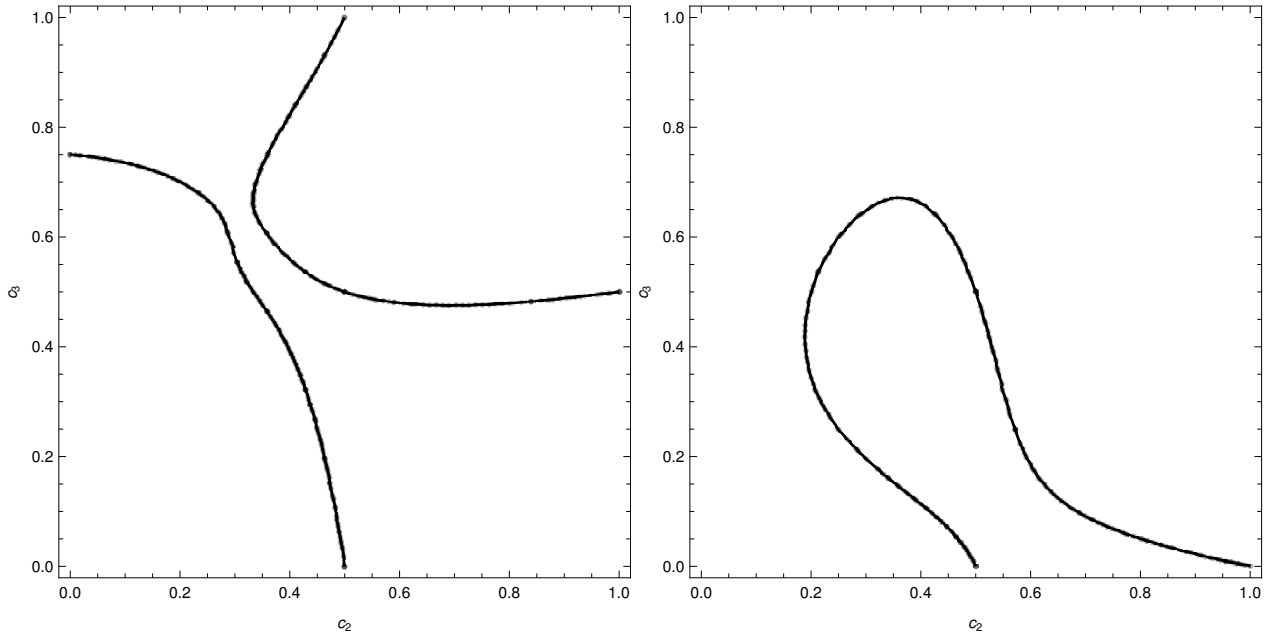


Figure 4.3. Choices of  $c_2$  and  $c_3$  on these lines result in the higher order method, otherwise the method is lower order

## Chapter 5

### Numerical Tests: Observed Order

In this work, we consider three test problems, the inverter-chain problem, a stability system problem, and the brusselator. These numerical tests are meant to verify the correctness of our software implementation, and to compare our computational results against their theoretically expected orders of convergence. The numerical tests also allow us to perform efficiency comparison tests, where we use metrics based on typical error norms and use function calls to define the cost of the integration.

#### 5.1. Test Problems

In the following 3 subsections, we will briefly describe each of the three test problems. We include the application-based and mathematical motivation. We specify the system of ODEs and the fast/slow partitioning. We indicate the time-scale separation of the test problem.

##### 5.1.1. Inverter-chain Test Problem

The inverter-chain problem is so named because it models a chain of MOSFET (metal oxide semiconductor field effect transistor) inverters. These inverters transform a signal from 0 to 1 and from 1 to 0. This physical model is appropriate because it is regularized, tunable, and scalable [18]. The solutions are very regular, and approach equilibrium points. It was first described from a physical perspective in Kværno and Rentrop's 1999 paper [34], which also gives a mathematical model which is formulated as a partitioned system of ordinary differential equations. The specific system of ODEs that we use to exercise our multirate methods is primarily based on Kværno and Rentrop's 1999 paper, although a scaling term is added as in the more recent test problem defined by Bartel and used by Constantinescu and Sandu [2, 9].

The mathematical model is based on Kirchoff's laws, namely: Kirchoff's current law which states "At any node in an electrical circuit, the sum of the currents flowing into that node is equal to the sum of the currents flowing out of that node" and Kirchoff's voltage law which states "The signed sum of the voltages around any closed circuit must be zero". Kværno identifies the following modeling principles for this setup:

- There are five basic elements: resistor, capacitor, inductor (in analogue circuits), voltage and current sources.
- The (non)linear controlled voltage or current source and the resistor describe the static behavior.
- The (non)linear controlled capacitors or inductors describe the dynamic behavior. [34]

Specifically, a voltage controlled current source is selected in order to simplify the resulting model. "The current source fits the drain current with respect to the gate voltage,"

$$\begin{aligned} I_{GS} = I_{GD} = I_{GB} = 0, \quad B : \text{bulk}, D : \text{drain}, G : \text{gate}, S : \text{source}, \\ I_{BD} = I_{BS} = 0, \\ I_{DS} = K \cdot g(y_G, y_D, y_S), \end{aligned}$$

where  $g$  is defined piecewise:

$$g(y_G, y_D, y_S) = (\max(y_G - y_S - y_T, 0))^2 - (\max(y_G - y_D - y_T, 0))^2$$

The technical parameters are:  $K = 2 \cdot 10^{-4} [AV^{-2}]$ ,  $y_T = 1 [V]$ ,  $y_{op} = 5 [V]$ .

Applying Ohms law, assuming a constant capacitor and using  $y_0 = 0$  gives

$$y_1' = (y_{op} - y_1) / (RC) - K \cdot g(y_{in}, y_1, y_0) / C.$$

The runtime of this circuit is measured in nanoseconds, so we scale by a factor  $10^9$ . Realistic values for the resistor  $R = 5 \cdot 10^3$  and for the capacitor  $C = .2 \cdot 10^{-12}$  we end up with the normalized equation

$$y_1' = (y_{op} - y_1) - g(y_{in}, y_1, y_0).$$

The output of an inverter can be fed as the gate-input of a next inverter and so on in order to create an inverter chain[34]. We convert this into a partitioned problem by considering some of the inverters as fast  $\mathbf{y}_{1,\dots,b}^{\{f\}} = [\mathbf{y}_1, \dots, \mathbf{y}_b]$  and the rest as slow  $\mathbf{y}_{1,\dots,n_i-b}^{\{s\}} = [\mathbf{y}_{b+1}, \dots, \mathbf{y}_{n_i}]$ . This component solution partitioning also partitions the differential equations.

$$\begin{aligned} [\mathbf{y}'(t)]_{b+1,\dots,n_i} &= [\mathbf{y}^{\{s\}}(t)]'_{1,\dots,n_i-b} = f^{\{s\}}(t, \mathbf{y}) \\ [\mathbf{y}'(t)]_{1,\dots,b} &= [\mathbf{y}^{\{f\}}(t)]'_{1,\dots,b} = f^{\{f\}}(t, \mathbf{y}) \end{aligned}$$

Substituting in the fast and slow functions as follows from the model, the following system is obtained,[34]

$$\begin{aligned} [\mathbf{y}'(t)]_k &= [\mathbf{y}^{\{s\}}(t)]'_{(k-b)} = (y_{op} - y_k(t)) - g(y_{k-1}(t), y_k(t), y_0) \quad \text{for } k = b+1, \dots, n_i \\ [\mathbf{y}'(t)]_k &= [\mathbf{y}^{\{f\}}(t)]'_{(k)} = \begin{cases} (y_{op} - y_1(t)) - g(y_{in}(t), y_1(t), y_0), & k = 1 \\ (y_{op} - y_k(t)) - g(y_{k-1}(t), y_k(t), y_0), & k = 2, \dots, b. \end{cases} \end{aligned}$$

One of the reasons this model was selected as a test problem is that the inverter-chain problem is consistently used to exercise multirate methods for multiphysics applications which use Runge-Kutta base methods [2, 9, 18, 24, 36, 47, 46, 48, 49, 54, 55]. In addition to the 1999 paper by Kværno and Rentrop, it has been extend by Günther and by Constantinescu to be more tunable [9, 19, 17, 34]. Making a similar modification to the Kværno version of the model adds a parameter  $\gamma$  which controls the strength of the forcing/source term  $g$ .

$$\begin{aligned} [\mathbf{y}'(t)]_k &= [\mathbf{y}^{\{s\}}(t)]'_{(k-b)} = (y_{op} - y_k(t)) - \gamma g(y_{k-1}(t), y_k(t), y_0) \quad \text{for } k = b+1, \dots, n_i \\ [\mathbf{y}'(t)]_k &= [\mathbf{y}^{\{f\}}(t)]'_{(k)} = \begin{cases} (y_{op} - y_1(t)) - \gamma g(y_{in}(t), y_1(t), y_0), & k = 1 \\ (y_{op} - y_k(t)) - \gamma g(y_{k-1}(t), y_k(t), y_0), & k = 2, \dots, b. \end{cases} \end{aligned}$$

This tunable parameter makes determining the time-scale separation much more straightforward. We choose  $\gamma = 100$  so that the maximum time-scale separation is 100. Here, the piecewise-constant forcing function  $y_{in}(t)$  is defined as

$$y_{in}(t) = \begin{cases} 0 & 0 \leq t < 5, \\ t - 5 & 5 \leq t \leq 10. \end{cases}$$

We integrate this problem from  $0 \leq t \leq 7$ , and have chosen our forcing function as in [34]. For our tests, we choose our partitioning so that the fast components are the first  $b = 3$  components (which correspond to the first 3 inverters) out of  $n_i = 100$ . In initial testing, we selected  $b = 20$  based on Günther’s claim that over the entire time integration interval, there are about 20 active inverters, with a large latent part [18]. However, perhaps due to the solutions appearing to follow a boundary layer in the beginning of the time integration interval, this proved unstable for some larger choices of the time-step size, such as  $h = .01$ . The distinction between which components we mark as fast and which are marked as slow is shown in Figure 5.1. For added detail, we note the importance of using a suitable time-step, especially when the solutions are first diverging from the boundary layer as shown in Figure 5.2

Kværno and Rentrop develop their component-based partitioning for this system adaptively throughout the integration window. Each component is marked stiff or non-stiff, and also marked as partitioned into a latent (slow) or active (fast) component. To summarize their partitioning strategy, first new step sizes are proposed for all active components using two different error approximations, then components where the more stable approximation suggests a larger step size is required are marked as stiff, then based on the active proposed time-steps and the stiff marking, a macro step size is proposed for the latent components, then this macro step is compared with the proposed fast step sizes in order to find an integer time-scale separation  $m$ . This inverter chain problem and its adaptivity strategy for time-step selection motivate our future goal of developing time-step adaptivity selection based on using Algorithm (3) to create an embedded solution to compare against our Algorithm (5) solution.

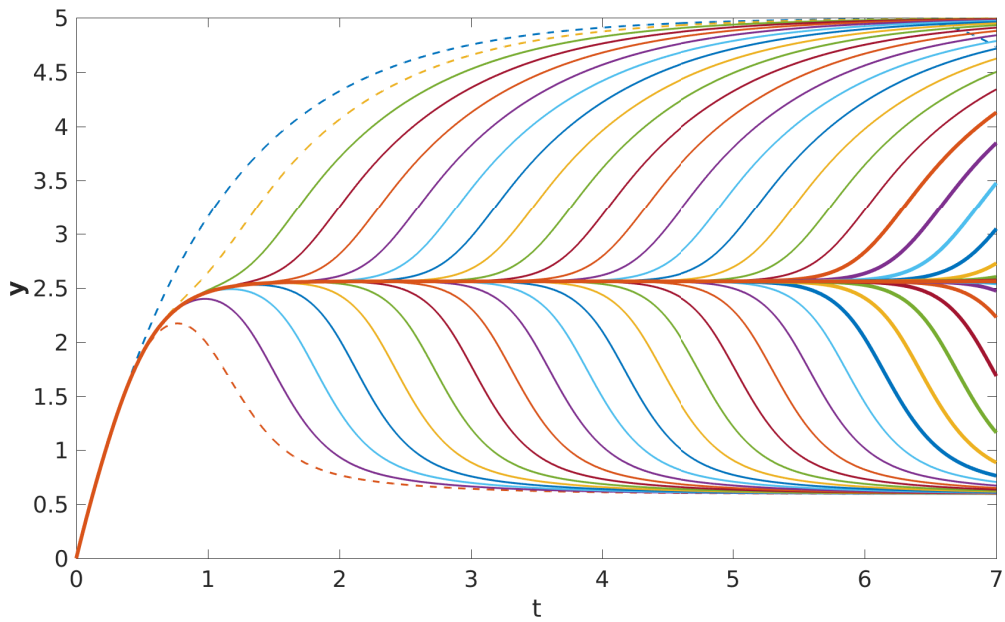


Figure 5.1. Solutions for the inverter chain problem with  $n_i = 100$ : Dotted lines represent the fastest components, while thick lines represent the slowest, i.e.  $y_1$  is the blue dotted curve at the top-left and  $y_{100}$  remains at the value 2.5 at the final time,  $t = 7$ .

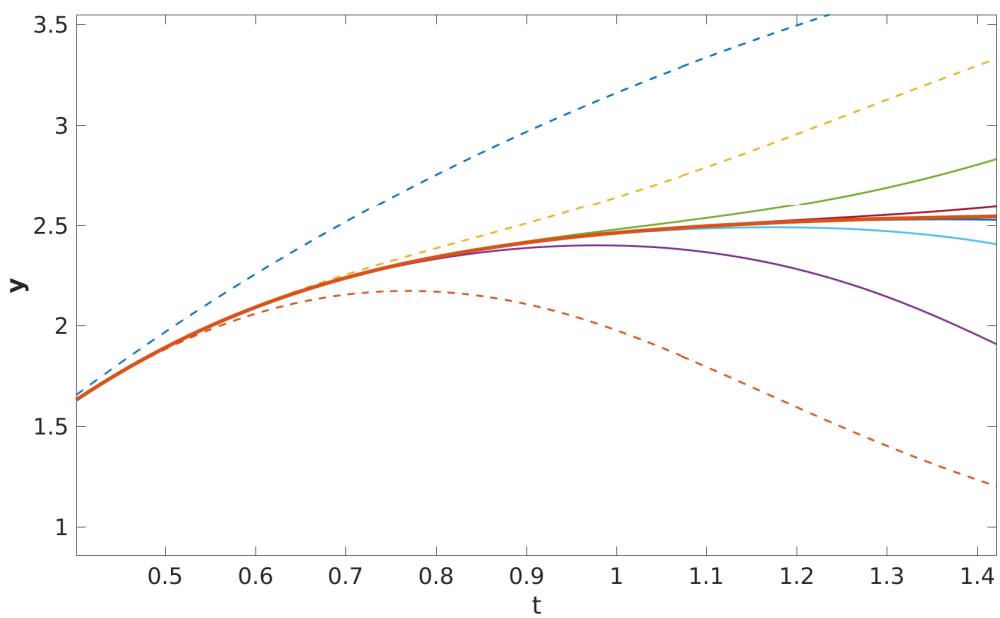


Figure 5.2. Small differences in the initial integration solution may result in disparate solution values



### 5.1.2. Kuhn Linear Test Problem

This test problem was chosen because it demonstrates a strongly-coupled problem which is linear. This is meant to exercise the performance of the error terms which correspond to the coupling matrices  $\mathbf{A}^{\{f,s\}}$  and  $\mathbf{A}^{\{s,f\}}$ . Kuhn and Lang proposed the following two-component initial value problem when examining the linear stability of a multirate system with a splitting [31]:

$$\begin{aligned} \begin{pmatrix} [\mathbf{y}^{\{f\}}(t)]' \\ [\mathbf{y}^{\{s\}}(t)]' \end{pmatrix} &= \begin{pmatrix} -5 & -1900 \\ 5 & -50 \end{pmatrix} \begin{pmatrix} \mathbf{y}^{\{f\}}(t) \\ \mathbf{y}^{\{s\}}(t) \end{pmatrix}, \quad t \in [0, 1], \\ \begin{pmatrix} \mathbf{y}^{\{f\}}(0) \\ \mathbf{y}^{\{s\}}(0) \end{pmatrix} &= \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \end{aligned} \tag{5.1}$$

Kuhn and Lang reference the ratio of the off-diagonal entries as a measure of the stiffness of the system,  $\kappa = 10$  for this particular method, which follows the original description by Kværno [31, 33]. Constantinescu and Sandu rescale their interpretation of the  $2 \times 2$  linear stability problem, and measure the time-scale separation by dividing the diagonal entries, and assume that  $a_{11}/a_{22} > 1$  [9]. Although this problem is linear, there is an interesting multirate structure in that  $\mathbf{y}^{\{f\}}(t)$  has a faster time-scale mainly based on the coupled data from  $\mathbf{y}^{\{s\}}(t)$ . In order to transform Equation (5.1) into a form similar to scaled problem Constantinescu and Sandu propose, the order of the equations and the solution vector must be scaled. Equations (5.2)-(5.4) show this transformation using the assumption Kuhn and Lang make about which component is slow and which is fast. Equations (5.5)-(5.6) show this transformation respecting the assumption Constantinescu and Sandu make about how the diagonal elements of a linear system determine which components are fast and which components are slow. The eigenvalues of this system are complex conjugates,  $-\frac{55}{2} \pm \frac{5}{2}\sqrt{1439}$ , indicating that the solutions to this problem are oscillatory and rapidly decay to zero, as shown below in Figure 5.3.

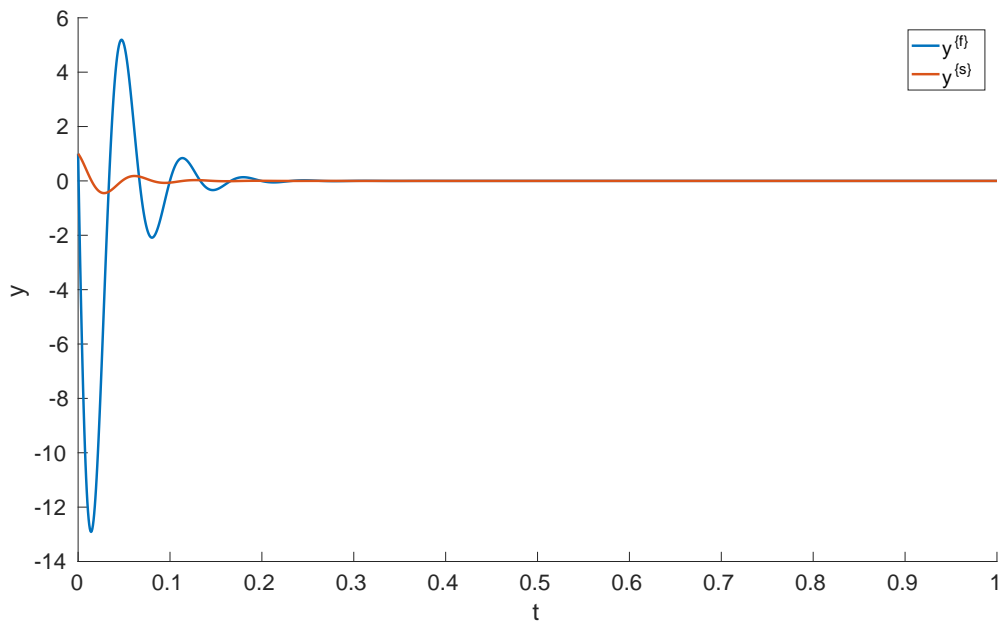


Figure 5.3. Solutions for the Kuhn test problem (5.1): note that the second component reaches equilibrium much more rapidly than, and does not vary by as much as, the first component.

$$\begin{pmatrix} [\mathbf{y}^{\{f\}}(t)]' \\ [\mathbf{y}^{\{s\}}(t)]' \end{pmatrix} = \begin{pmatrix} -5 & -1900 \\ 5 & -50 \end{pmatrix} \begin{pmatrix} \mathbf{y}^{\{f\}}(t) \\ \mathbf{y}^{\{s\}}(t) \end{pmatrix}, \quad t \in [0, 1], \quad (5.2)$$

$$\begin{pmatrix} \mathbf{y}^{\{f\}}(0) \\ \mathbf{y}^{\{s\}}(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} [\mathbf{y}^{\{s\}}(t)]' \\ [\mathbf{y}^{\{f\}}(t)]' \end{pmatrix} = \begin{pmatrix} -50 & 5 \\ -1900 & -5 \end{pmatrix} \begin{pmatrix} \mathbf{y}^{\{s\}}(t) \\ \mathbf{y}^{\{f\}}(t) \end{pmatrix} \quad (5.3)$$

$$\begin{pmatrix} \mathbf{y}^{\{s\}}(0) \\ \mathbf{y}^{\{f\}}(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} [\mathbf{y}^{\{\hat{s}\}}(t)]' \\ [\mathbf{y}^{\{f\}}(t)]' \end{pmatrix} = \begin{pmatrix} -1 & 1/10 \\ -38 & 1/10 \end{pmatrix} \begin{pmatrix} \mathbf{y}^{\{\hat{s}\}}(t) \\ \mathbf{y}^{\{f\}}(t) \end{pmatrix}, \quad t \in [0, 1], \quad (5.4)$$

$$\begin{pmatrix} \mathbf{y}^{\{f\}}(0) \\ \mathbf{y}^{\{\hat{s}\}}(0) \end{pmatrix} = \begin{pmatrix} 1/50 \\ 1/50 \end{pmatrix},$$

$$\begin{pmatrix} [\mathbf{y}^{\{\bar{s}\}}(t)]' \\ [\mathbf{y}^{\{\bar{f}\}}(t)]' \end{pmatrix} = \begin{pmatrix} -10 & 1 \\ -380 & -1 \end{pmatrix} \begin{pmatrix} \mathbf{y}^{\{\bar{s}\}}(t) \\ \mathbf{y}^{\{\bar{f}\}}(t) \end{pmatrix}, \quad t \in [0, 1], \quad (5.5)$$

$$\begin{pmatrix} \mathbf{y}^{\{\bar{f}\}}(0) \\ \mathbf{y}^{\{\bar{s}\}}(0) \end{pmatrix} = \begin{pmatrix} 1/5 \\ 1/5 \end{pmatrix}.$$

### 5.1.3. Brusselator Test Problem

We consider a system of ordinary differential equations which captures some of the physical problems of the brusselator chemical reaction network problem first described as a 1D PDE by Prigogine in 1967 [37]. More recent usages of this test problem have been seen in a

multiphysics paper by Estep in 2008, and a general computational multiphysics review paper in 2013 [12, 26]. Our version of the problem is a tunable two-rate initial-value problem represented as a system of Ordinary Differential Equations, [22]. It is very similar to the ODE system Hairer proposed as a simplification from 6 derivative terms to 2 derivative terms [22]. For our system, we include  $\mathbf{y}_3$  which has the fast initial transient, whereas Hairer neglects this term [22]. It may be formulated as an additive problem as follows:

$$\begin{aligned}
\frac{d\mathbf{y}}{dt} &= f^{\{f\}} + f^{\{s\}} \\
&= \begin{bmatrix} 0 \\ 0 \\ \frac{b-\mathbf{y}_3}{\varepsilon} \end{bmatrix} + \begin{bmatrix} a - (\mathbf{y}_3 + 1)\mathbf{y}_1 + \mathbf{y}_2\mathbf{y}_1^2 \\ \mathbf{y}_3\mathbf{y}_1 - \mathbf{y}_2\mathbf{y}_1^2 \\ -\mathbf{y}_3\mathbf{y}_1 \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ 0 \\ \frac{2.5-\mathbf{y}_3}{10^{-2}} \end{bmatrix} + \begin{bmatrix} 1.2 - (\mathbf{y}_3 + 1)\mathbf{y}_1 + \mathbf{y}_2\mathbf{y}_1^2 \\ \mathbf{y}_3\mathbf{y}_1 - \mathbf{y}_2\mathbf{y}_1^2 \\ -\mathbf{y}_3\mathbf{y}_1 \end{bmatrix}. \tag{5.6}
\end{aligned}$$

We integrate over the interval  $0 \leq t \leq 10$ , with the initial conditions  $\mathbf{y} = [3.9, 1.1, 2.8]$ . Although  $a$ ,  $b$ , and  $\varepsilon$  can be used to represent a variety of problems, we choose  $a = 1.2$ ,  $b = 2.5$ , and  $\varepsilon = 10^{-2}$ . This is a model of a chemical reaction network with three different reactants, where 5.6 defines the evolution of the concentrations. As shown in the partitioning, the fast function  $f^{\{f\}}$  contains the term which is scaled by  $\varepsilon$ . The problem time-scale separation is approximately  $1/\varepsilon = 10^2 = 100$ . With this particular choice of parameters and initial conditions, the problem exhibits a sudden change in the solution at the start of the simulation for  $t < 0.2$ , with slower variation for the remainder of the time interval, as shown in Figure 5.4. Because our tests are carried out with a fixed macro time-step  $h$ , this slower variation made our tests less sensitive to small variations after the initial sudden change. This particular test also exercises the nonlinear error terms.

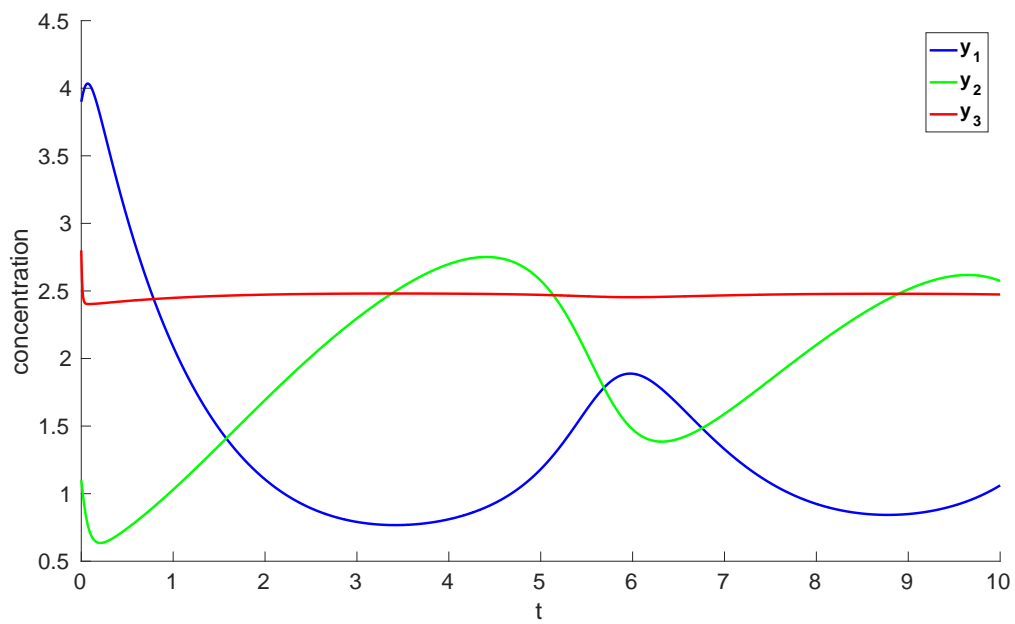


Figure 5.4. Solutions for the Brusselator test problem (5.6):  $y_3$  initially rapidly approaches  $b = 2.5$ , then varies slowly thereafter.

## 5.2. Test Setup

In order to test our methods numerically, we implemented Algorithms 2-6 in MATLAB. For these tests, we only consider explicit inner base methods and explicit outer base methods. Specifically, we consider the 3-stage third order method with equally spaced stage times KW3 from Equation (4.13) and the 4-stage fourth order method with equally spaced stage times 3/8 – Rule from Equation (4.17). The KW3 base method is of interest because the RFSMR which used the KW3 as the inner and outer base method performed the best in our initial comparative study, and because it is used in the literature for comparisons with newly developed methods [29, 50, 49, 57]. We chose the 3/8 – Rule method because it is generated from one of the  $c_2, c_3$  pairs for which a RFSMR method will be third order and a RMIS method will be fourth order. This method also has the benefit of the stage times being equally spaced, which simplifies determining how many subcycles are necessary between each slow outer stage time in order to target a particular time-scale separation. It also has historical significance because Runge and König include it in their 1924 book [40]. Given these two base methods, we will use the previously mentioned test problems to investigate the convergence and efficiency properties of the fmGARK method types enumerated in Chapter 4. As these are explicit methods and we have not investigated efficient time-step adaptivity strategies the tests here will use a fixed time-step  $h$ . We have chosen  $h$  values so that all step sizes are the same, and the final time step ends at  $t_F$ . Specifically, we choose  $h$  values so that integration intervals of integer length would have an integer number of time-steps  $h$  between  $t_0$  and  $t_F$ .

$$h \in \{0.1, 0.05, 0.025, 0.0125, 0.01, 0.005, 0.0025, 0.00125, \dots \\ 0.001, 0.0005, 0.00025, 0.000125, 0.0001, 0.00005, 0.000025, 0.0000125\}$$

To reiterate, the stage and solution values are represented by

$$\begin{aligned}
k_j^{\{f\}} &= y_n + h \sum_{l=1}^{j-1} a_{jl}^{\{f,f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, k_l^{\{f\}} \right) + h \sum_{l=1}^{s^{\{s\}}} a_{jl}^{\{f,s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, k_l^{\{s\}} \right), \\
k_i^{\{s\}} &= y_n + h \sum_{l=1}^{s^{\{f\}}} a_{il}^{\{s,f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, k_l^{\{f\}} \right) + h \sum_{l=1}^{i-1} a_{il}^{\{s,s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, k_l^{\{s\}} \right), \\
y_{n+1} &= y_n + h \sum_{l=1}^{s^{\{f\}}} b_l^{\{f\}} f^{\{f\}} \left( t + c_l^{\{f\}} h, k_l^{\{f\}} \right) + h \sum_{l=1}^{s^{\{s\}}} b_l^{\{s\}} f^{\{s\}} \left( t + c_l^{\{s\}} h, k_l^{\{s\}} \right).
\end{aligned}$$

Specifically, we will show results for a total of five different combinations of these two base methods with our three fmGARK method types each of which has its own specific intermediate solution calculation and final step solution update as represented by Algorithms 3-6. As stated in Section 3.3, the implementation we suggest requires that we store  $\max_{i=1}^{s^{\{O\}}} s^{\{I_i\}} + s^{\{O\}} + 2$  vectors the size of our solution  $\mathbf{y}$ . Note that Algorithm 6 requires the storage of the entire vector  $\mathbf{b}^f$ , which is a vector the same length as the total number of fast stage functions computed, whereas for the other two algorithms we must store only the inner and outer base method. Algorithm 3 using the Knoth-Wolke 3 base method was chosen to be used as a baseline to compare against which theoretically should demonstrate 3rd order behavior. Algorithm 5 using the Knoth-Wolke 3 base method was chosen to demonstrate a method where the outer base method assumptions for the RMIS method are not satisfied, which theoretically should demonstrate 3rd order behavior. Algorithm 3 using the case I base method was chosen to demonstrate a method where the outer base method assumptions for the fmGARK slow order condition (2.51) are satisfied, but the underlying algorithm should cause it to theoretically demonstrate 3rd order behavior. Algorithm 5 using the case I base method was chosen to demonstrate a method where the outer base method assumptions for the fmGARK slow order condition (2.51) are satisfied, and the underlying algorithm should cause it to theoretically demonstrate 4th order behavior. Algorithm 6 using the case I base method was chosen to demonstrate a method where the outer base method assumptions for the fmGARK slow order condition (2.51) are satisfied, and the underlying algorithm should cause it to theoretically demonstrate 4th order behavior.

In order to evaluate the convergence and efficiency of these methods, we must derive an exact solution to the test problems or compute a reference solution as appropriate. For the Kuhn problem, we compute error by comparison against the analytical solution,

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \mathbf{y}^{\{f\}} \\ \mathbf{y}^{\{s\}} \end{pmatrix} = e^{\{-55\frac{t}{2}\}} \begin{pmatrix} \cos\left(\frac{5\sqrt{1439}t}{2}\right) - \frac{751}{\sqrt{1439}} \sin\left(\frac{5\sqrt{1439}t}{2}\right) \\ \cos\left(\frac{5\sqrt{1439}t}{2}\right) - \frac{7}{\sqrt{1439}} \sin\left(\frac{5\sqrt{1439}t}{2}\right) \end{pmatrix}.$$

For the inverter chain and brusselator test problems we compute error by comparison against reference solutions. For these problems, we generated reference solutions by using the 12th order accurate implicit Runge-Kutta Gauss method, taking fixed time-steps which are 4 times smaller than the smallest  $h$  value tested for our multirate methods. Using these reference and exact solutions, we can approximate or measure the error in the solution computed using our method. Denote our computed solution at time  $t_n = t_0 + hn$  as  $\mathbf{y}_{n,h}$  and the reference or exact solution at time  $t_n = t_0 + hn$  as  $\mathbf{y}_{n,ref}$ , our approximation for the method's error at  $t_n = t_0 + hn$  is

$$\mathbf{err}_{n,h} = \mathbf{y}_{n,h} - \mathbf{y}_{n,ref}.$$

The canonical root-mean-square norm modifies the vector two-norm by normalizing by the number of observations to allow for a better comparison of error. Specifically, the vector root-mean-squared norm for a vector  $\mathbf{v}$  with  $N$  components is

$$\|\mathbf{v}\|_{RMS} = \left( \frac{1}{N} \sum_{i=1}^N v_i^2 \right)^{1/2} = \frac{1}{N^{1/2}} \|\mathbf{v}\|_2.$$

We base our error metric for each method and for each  $h$  value on this norm. Note that our bounds of integration are  $t \in [t_0, t_F]$  with a fixed macro step size  $h$ . Therefore, the number of error vectors calculated is  $\frac{t_F - t_0}{h}$ . Equation (5.7) below shows how to compute **RMSerror** as a metric which measures the error for a particular multirate method,  $h$  choice, and test problem integrated for  $t \in [t_0, t_F]$  based on the reference solutions.



$$\begin{aligned}
\text{RMSError} &= \left( \sum_{n=1}^{\frac{t_F-t_0}{h}} \left( \frac{1}{\left(\frac{t_F-t_0}{h}\right)} \right) \|\mathbf{err}_{n,h}\|_{RMS}^2 \right)^{1/2} \\
&= \left( \sum_{n=1}^{\frac{t_F-t_0}{h}} \left( \frac{1}{\left(\frac{t_F-t_0}{h}\right)} \right) \left( \frac{1}{N} \sum_{i=1}^N ([\mathbf{err}_{n,h}]_i)^2 \right) \right)^{1/2} \tag{5.7}
\end{aligned}$$

The `TotalFunctionCalls` metric for the efficiency analysis is based on  $\frac{t_F-t_0}{h}$  : the number of macro steps taken,  $n_i$  : the number of fast step subcycles per slow stage,  $s^{\{I\}}$  : inner base method stages and  $s^{\{O\}}$  : outer base method stages. Specifically, it adds the number of outer base method stages which require a new slow function call to the number of inner base method stages which require a new fast function call multiplied by the number of substeps per slow stage.

$$\text{TotalFunctionCalls} = \frac{t_F - t_0}{h} \left( s^{\{O\}} + \sum_{i=1}^{s^{\{O\}}} s^{\{I\}} n_i \right)$$

Note that  $\frac{t_F-t_0}{h} \left( s^{\{O\}} + \sum_{i=1}^{s^{\{O\}}} s^{\{I\}} n_i \right) = h * \frac{t_F-t_0}{h^2} \left( s^{\{O\}} + \sum_{i=1}^{s^{\{O\}}} s^{\{I\}} n_i \right)$ . Requiring the first stage time  $c_1 = 0$  and the last stage time  $c_4 = 1$  is a necessary, but not sufficient condition for Runge-Kutta methods which are explicit with four stages and satisfy the row-sum assumption to be fourth order [22]. Because of the structure of the fmGARK method,  $\mathbf{A}^{\{s^{\{O\}}, s^{\{O\}}\}}$  is equal to the zero matrix because  $1 - c_{(s^{\{O\}})} = 0$ . So total function calls per macro step for our particular methods which use \ref{1/3} are  $4 + (4*33 + 4*33 + 4*33) = 400$ , and which use 4.13 are  $3 + (3*35 + 3*35 + 3*35) = 318$ . The fast stage to slow stage ratio for 4.17 is  $408/4 = 102$ , and for 4.13 is  $315/3 = 105$ .

### 5.3. Numerical Convergence and Efficiency Results

As stated in Section 5.2, the metrics we use to compute observed numerical order are the fixed time-step size  $h$ , the RMS-norm based error metric `RMSError` and the total number of function calls `TotalFunctionCalls`. In order to investigate order of convergence, we compare our metric `RMSError` with  $h$  values for each of our test problems. We will present

Table 5.1. Table of Method Efficiency Properties: The base method choice determines the number of stages, and affects the ratio of fast stages to slow stages in the overall GARK table.

Method Efficiency	$s^{\{I\}} = s^{\{O\}}$	$\mathbf{s}^{\{f\}}/\mathbf{s}^{\{s\}}$	$\left(s^{\{O\}} + \sum_{i=1}^{s^{\{O\}}} s^{\{I\}}\eta_i\right)$
Algorithm 6 w/ Base Method 4.17	4	102	400
Algorithm 5 w/ Base Method 4.17	4	102	400
Algorithm 5 w/ Base Method 4.13	3	105	318
Algorithm 3 w/ Base Method 4.17	4	102	400
Algorithm 3 w/ Base Method 4.13	3	105	318

tables of observed numerical order based on a least-squares fit of the  $h$  vs `RMSerror` data where outliers are ignored, that is, we only include simulations for numerical order calculation when  $10^{-9} \leq \text{RMSerror} \leq 1$ . We also plot `RMSerror` vs  $h$  in order to examine convergence behavior not captured by the best-fit line. To read the convergence plots, the slope on the log-log plot corresponds to the order of convergence. As shown in Table 5.1, the cost per macro time step  $h$  is affected by how many stages the base methods have. In order to investigate the efficiency of these methods, we plot `RMSerror` vs `TotalFunctionCalls`. These plots are meant to give a fairer comparison of efficiency and meant to be general. If a user knew the relative cost of computing  $f^{\{f\}}$  vs computing  $f^{\{s\}}$ , a problem-specific metric for efficiency could be developed. To read the efficiency plots, the most efficient method will have a line that is furthest towards the bottom-left corner.

### 5.3.1. Inverter-chain Results

As we described in Subsection 5.1.1, the inverter-chain problem we're interested in has a multirate structure which exhibits weakly-coupled behavior between the time-scales. This

test problem has boundary-layer behavior near the beginning of the time integration window. As described in Table 5.2, all methods tested had the expected order of accuracy except for Algorithm 6 w/ Base Method 4.17. The order of accuracy plot Figure 5.5

Table 5.2. Table of Inverter-Chain Convergence: The order of accuracy is expected, although our measure of observed order is limited by the inclusion of all results with error between 1 and  $10^{-9}$

Multirate Method	Expected Order	Observed Order
Algorithm 6 w/ Base Method 4.17	4	1.74
Algorithm 5 w/ Base Method 4.17	4	4.07
Algorithm 5 w/ Base Method 4.13	3	2.93
Algorithm 3 w/ Base Method 4.17	3	2.98
Algorithm 3 w/ Base Method 4.13	3	2.98

shows that the points excluded from our best fit line calculation do not have substantially increasing `RMSerror` for increasing  $h$ . Algorithm 6 w/ Base Method 4.17 initially (for  $h \in \{0.05, 0.025, 0.0125\}$ ) has a slope which is sharper than Algorithm 5 w/ Base Method 4.17, however for this pattern does not hold for the simulations with the smaller  $h$  values. There's a small increase in `RMSerror` and then the slope observed from `RMSerror` decreased for  $h \in \{0.005, 0.0025, 0.00125, 0.001\}$ , which could be taken to indicate convergence which is at least 3rd order. Finally, the smallest  $h$  values tested have notably similar `RMSerror` values, for  $h \in \{0.00025, 0.000125, 0.0001, 0.00005, 0.000025, 0.0000125\}$ . Our speculation for this unusual behavior is that the lack of structure in  $\mathbf{b}^{\{f\}}$  for this optimized method caused the double-precision order to suffer for smaller  $h$ . We speculate that the method performed at 4th order for larger  $h$  values, before the accumulated inaccuracy brought about by this lack of structure caused one of the 4th order error terms to no longer be canceled

out, resulting in the observed third order behavior. However, this method still converged to a solution that is  $10^{-8}$  away from our reference solution.

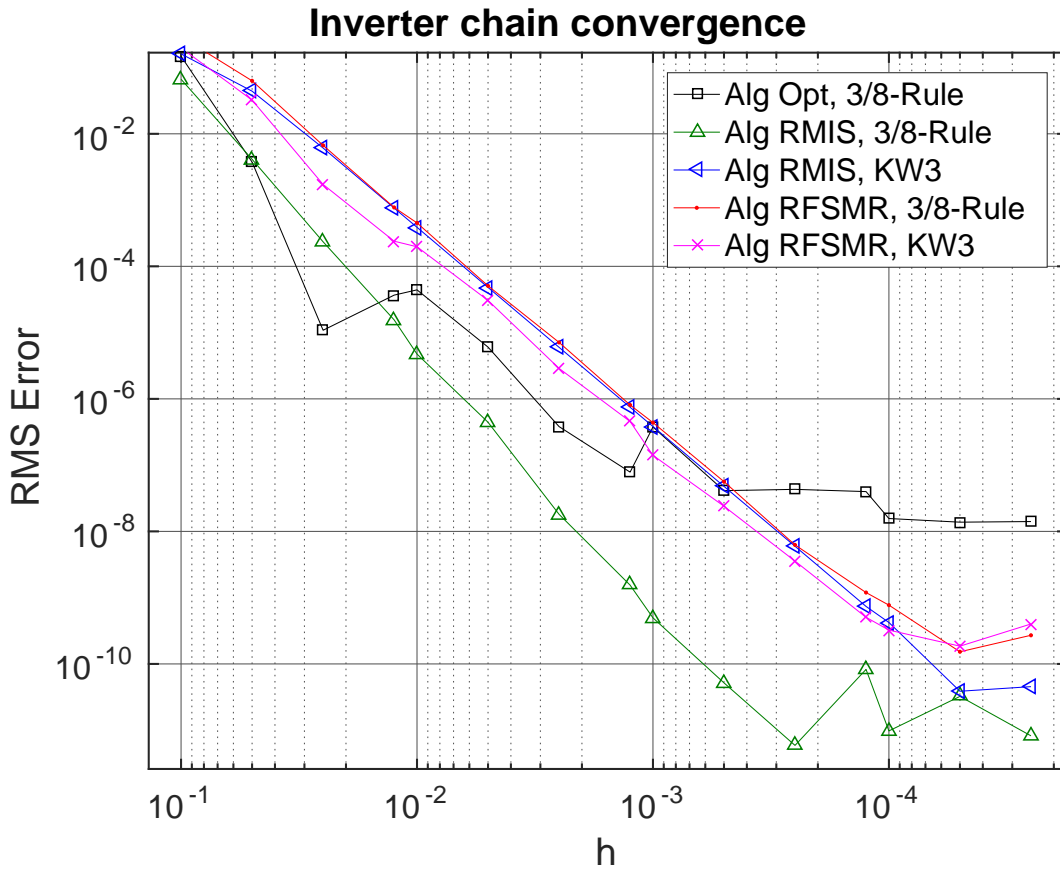


Figure 5.5. Convergence for the inverter chain problem: note that for this problem the slope for order of accuracy is consistent for all methods except Algorithm 6 w/ Base Method 4.17. The flattening of the error for this method may be due to inaccurate capturing of the initial integration window causing this method to converge to a solution which is approximately  $10^{-8}$  away from the reference solution.

We can see in Figure 5.6 that changing the x-axis from  $h$  to `TotalFunctionCalls` rescales the tested methods which were based on Base Method 4.17 by  $\frac{t_F - t_0}{h^2} \left( s^{\{O\}} + \sum_{i=1}^{s^{\{O\}}} s^{\{I\}} n_i \right) = \frac{7}{h^2} \times 400$  and those which were based on Base method (4.13) by  $\frac{7}{h^2} \times 318$ . When considering the efficiency for all  $h$  values, and therefore for all `RMSerror` observed, we see that Algorithm 5 w/ Base Method 4.17 is the clear winner.

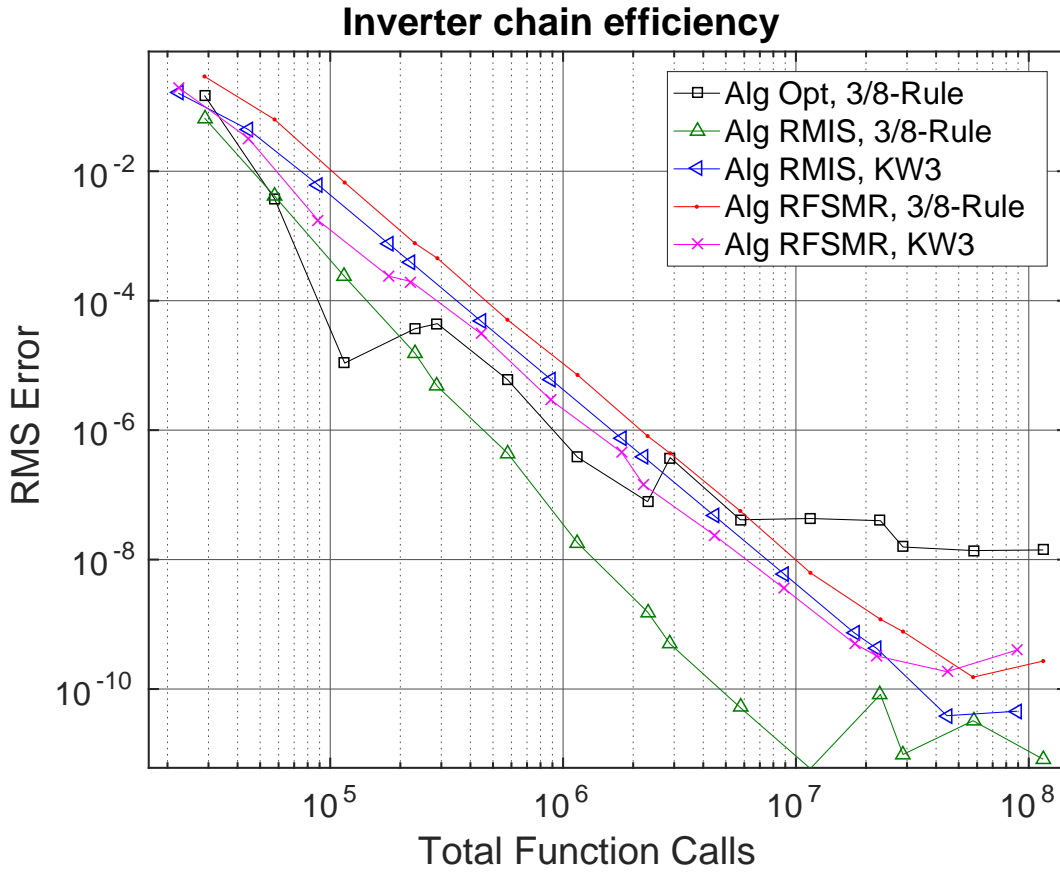


Figure 5.6. Efficiency for the inverter chain problem: note that for this problem the most efficient methods for larger error values are Algorithm 5 w/ Base Method 4.17 and Algorithm 6 w/ Base Method 4.17, while for smaller errors Algorithm 5 w/ Base Method 4.17 is the clear winner; all other methods perform comparably well.

### 5.3.2. Kuhn Linear Results

As we described in Subsection 5.1.2, the Kuhn linear problem we’re interested in has a multirate structure which exhibits strongly-coupled linear behavior between the time-scales. As described in Table 5.3, all methods tested had the expected order of accuracy.

Table 5.3. Table of Kuhn Convergence: The observed order of accuracy is slightly better than expected based on theory

Multirate Method	Expected Order	Observed Order
Algorithm 6 w/ Base Method 4.17	4	4.22
Algorithm 5 w/ Base Method 4.17	4	4.22
Algorithm 5 w/ Base Method 4.13	3	3.09
Algorithm 3 w/ Base Method 4.17	3	3.18
Algorithm 3 w/ Base Method 4.13	3	3.09

The order of accuracy plot Figure 5.7 shows that the points excluded from our best fit line calculation do not have increasing **RMSerror** for increasing  $h$ . The linear problem shows very straight order of accuracy lines for all methods except Algorithm 6 w/ Base Method 4.17, which begins to level off before it achieves the limit of double-precision representation of the solution.

Note that Figure 5.8 shows the change in x-axis from  $h$  to **TotalFunctionCalls** rescales the tested methods which were based on Base Method 4.17 by  $\frac{t_F-t_0}{h^2} \left( s^{\{O\}} + \sum_{i=1}^{s^{\{O\}}} s^{\{I\}} n_i \right) = \frac{1}{h^2} \times 400$  and those which were based on Base method (4.13) by  $\frac{1}{h^2} \times 318$ . Considering the efficiency for all  $0.01 \leq h \leq 10^{-4}$ , we see that Algorithm 5 w/ Base Method 4.17 and Algorithm 6 w/ Base Method 4.17 perform similarly and are the farthest towards the bottom-left corner. Algorithm 6 w/ Base Method 4.17 begins flattening as **RMSerror**  $\rightarrow 10^{-10}$ .

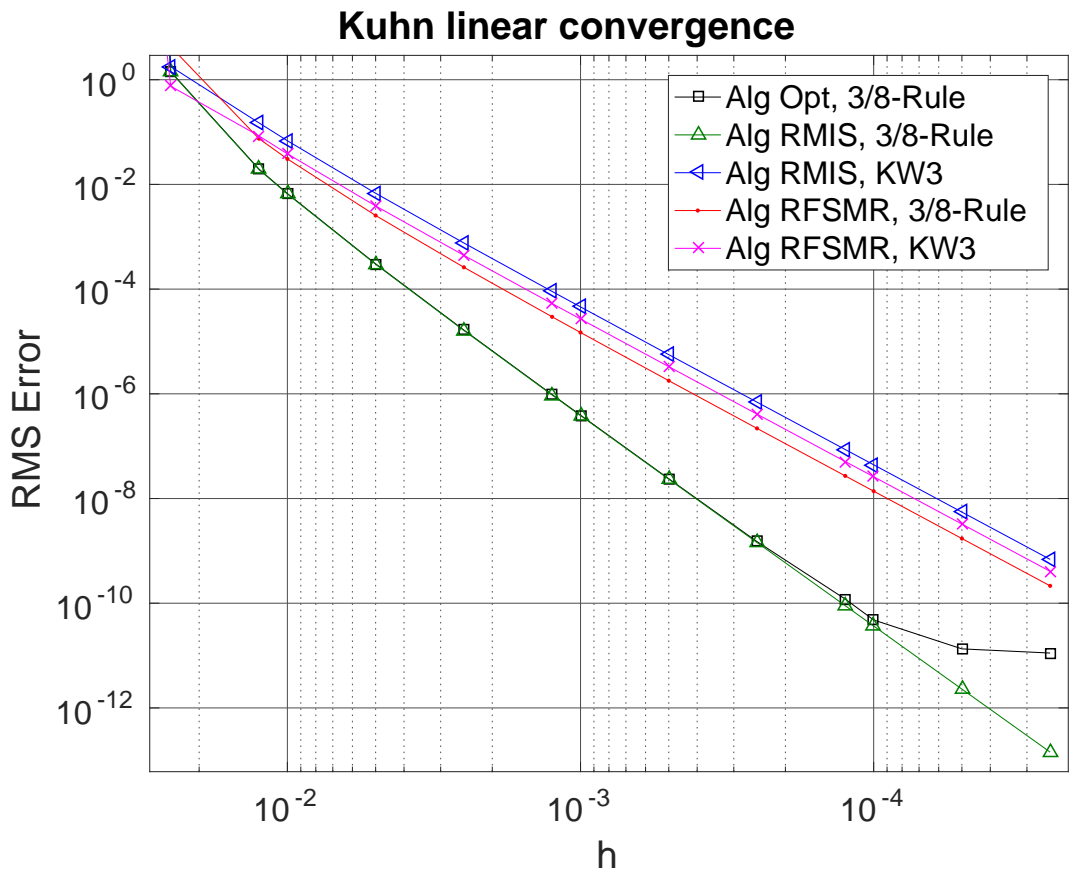


Figure 5.7. The numerical order observed is close to the expected values

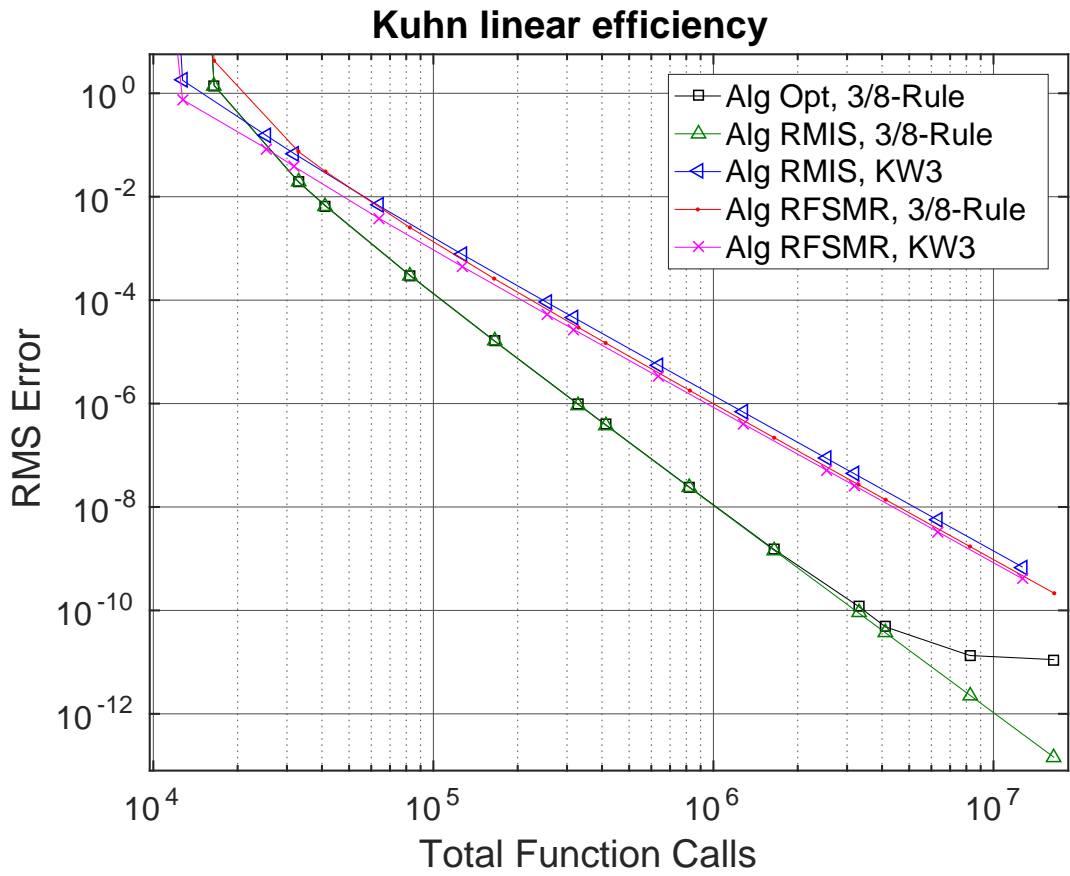


Figure 5.8. The higher order of accuracy methods are more efficient for almost all  $h$  values tested



### 5.3.3. Brusselator Results

As we described in Subsection 5.1.3, the Brusselator problem we’re interested in has a multirate structure which exhibits weakly-coupled nonlinear behavior between the time-scales. Table 5.4 describes the theoretically expected order of accuracy and the observed order of accuracy to make a comparison. We note that Algorithm 6 w/ Base Method 4.17 has an observed order of accuracy of 5.83, which suggests that the optimization successfully minimized the dominant error terms for this problem. The other methods have an observed numerical order of accuracy which is consistent with the theoretically expected order of accuracy.

Table 5.4. Table of Brusselator Convergence: The observed order of accuracy is slightly better than expected based on theory, and the observed order for the optimized method is significantly better

Multirate Method	Expected Order	Observed Order
Algorithm 6 w/ Base Method 4.17	4	5.83
Algorithm 5 w/ Base Method 4.17	4	4.16
Algorithm 5 w/ Base Method 4.13	3	3.30
Algorithm 3 w/ Base Method 4.17	3	3.28
Algorithm 3 w/ Base Method 4.13	3	3.02

Figure 5.9 shows that all methods perform at least as well as the expected order of accuracy. All methods tested leveled out at an error floor around  $10^{-9}$ , which is likely the accuracy of our reference solution. We can see in Figure 5.10 that changing the x-axis from  $h$  to **TotalFunctionCalls** rescales the tested methods which were based on Base Method 4.17 by  $\frac{t_F-t_0}{h^2} \left( s^{\{O\}} + \sum_{i=1}^{s^{\{O\}}} s^{\{I\}} n_i \right) = \frac{10}{h^2} \times 400$  and those which were based on Base method (4.13)

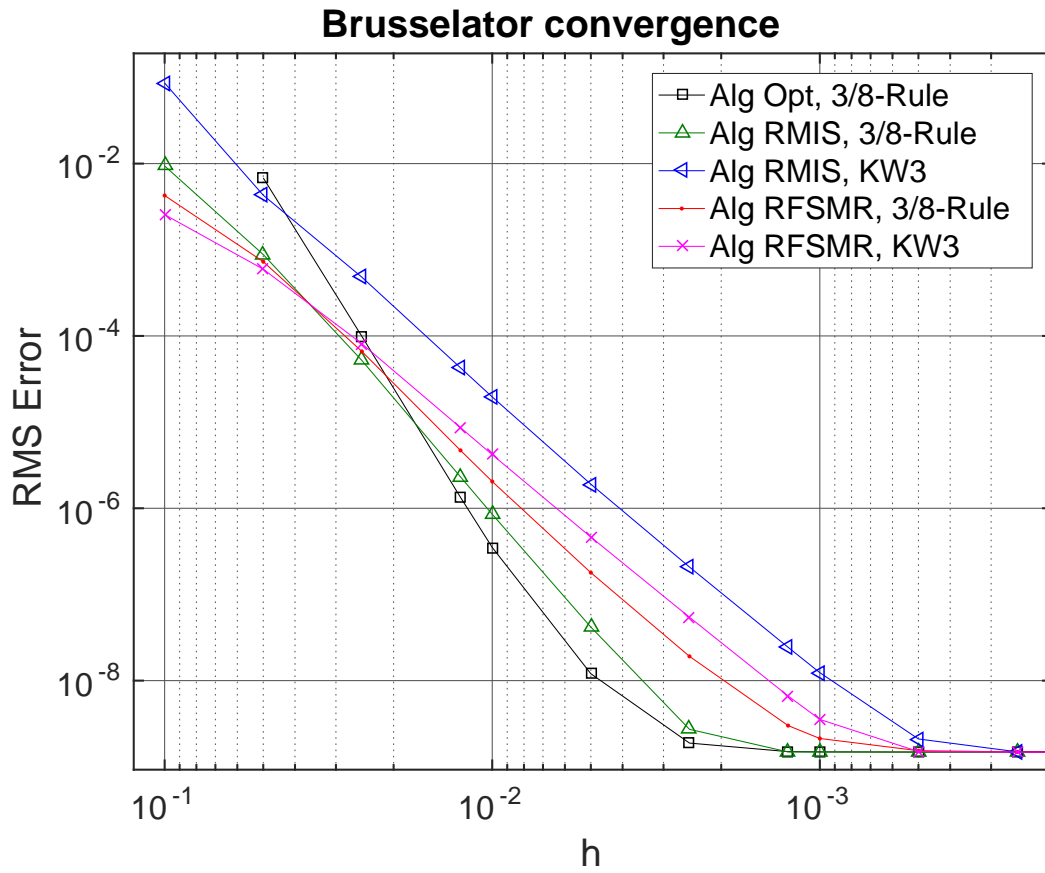


Figure 5.9. The optimized method and the RMIS seem to do best

by  $\frac{10}{h^2} \times 318$ . When considering the efficiency, we observe that around  $\text{RMSError} = 10^{-5}$ , the higher order methods begin to outperform the lower order methods. I don't know why the constant multiplier  $ch^4$  is so large for these particular methods for this particular problem. We do see that when a smaller error metric is required, the higher order methods Algorithm 5 w/ Base Method 4.17 and Algorithm 6 w/ Base Method 4.17 are more efficient.

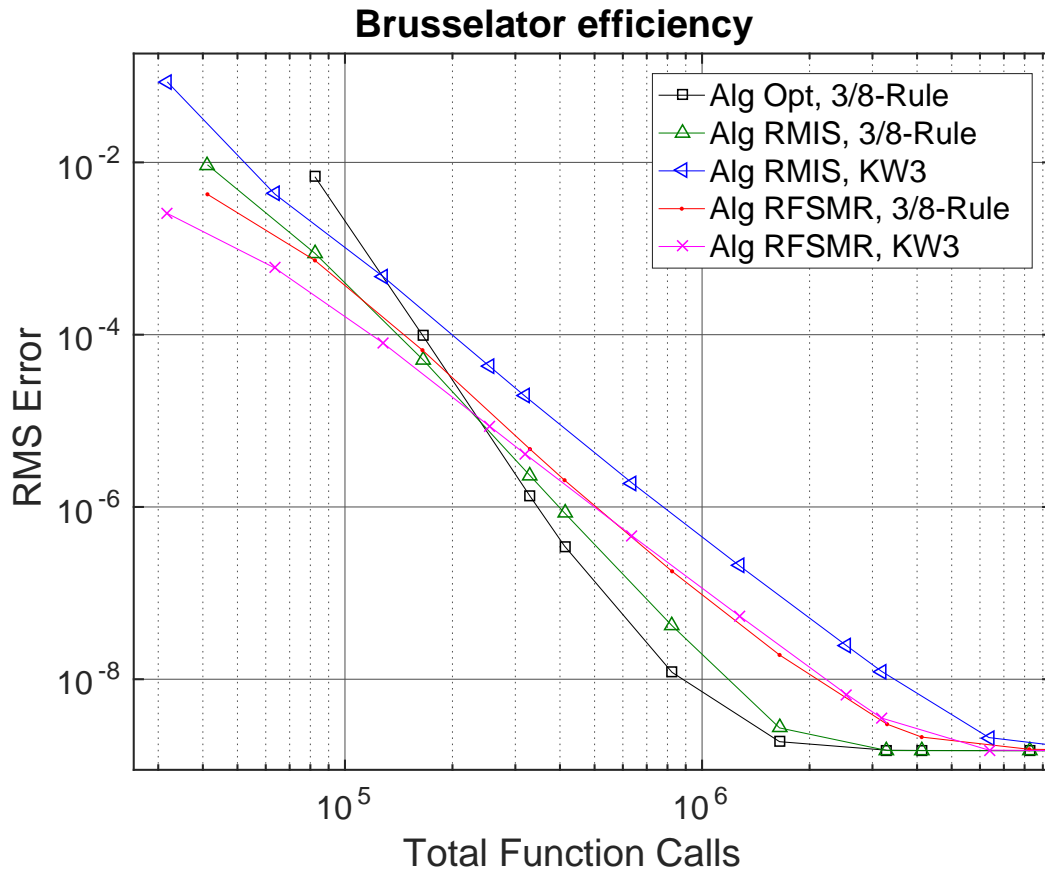


Figure 5.10. Method efficiency for this problem is dependent on both the order of accuracy and on the constant multiplier on the error terms

In general, our numerical tests suggest that the methods tested perform at least as well as the theoretical order of accuracy. Algorithm 5 w/ Base Method 4.17 is consistently efficient for all test problems, except for the Brusselator test problem for large values of  $\text{RMSError}$ .

## Chapter 6

### Stability Results: Theoretical & Numerical

In this chapter, we consider the linear stability of our proposed set of numerical methods. We first derive a stability recursion for GARK methods by applying existing stability theory. We then consider two different approaches to investigating the stability properties of our new methods by comparing them to the technique represented by Algorithm 3. The first approach will compare the two base methods for which Algorithm 3 gives a 3rd order method and Algorithm 5 gives a 4th order method. The second approach optimizes over 4-stage methods for which Algorithm 5 gives a 4th order method and for which Algorithm 3 gives a 3rd order method. The “largest” stability area for the highest possible order is then compared.

#### 6.1. Deriving a Stability Concept for GARK methods based on Kværno’s stability

As mentioned in Chapter 2, we derive a stability function for  $2 \times 2$  GARK methods based on the stability concept first proposed by Kværno, and later reintroduced by Savcenco and by Constantinescu [33, 47, 45, 9]. Recall the  $2 \times 2$  linear stability test problem:

$$\begin{pmatrix} y'_f(t) \\ y'_s(t) \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix} \begin{pmatrix} y_f(t) \\ y_s(t) \end{pmatrix}, \quad \begin{pmatrix} y_f(0) \\ y_s(0) \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Savcenco assumes a partitioned splitting when deriving his stability concept, which we rewrite as an additive splitting [45]. Let us first define

$$\mathbf{y}_n = \begin{pmatrix} y_{f,n} \\ y_{s,n} \end{pmatrix}, \quad \mathbf{k}_j^{\{f\}} = \begin{pmatrix} k_{f,j}^{\{f\}} \\ k_{s,j}^{\{f\}} \end{pmatrix}, \quad \text{and} \quad \mathbf{k}_j^{\{s\}} = \begin{pmatrix} k_{f,j}^{\{s\}} \\ k_{s,j}^{\{s\}} \end{pmatrix},$$

and write the additively-split right hand side functions as

$$f^{\{f\}}(t, \mathbf{y}) = \begin{pmatrix} g_{11} & g_{12} \\ 0 & 0 \end{pmatrix} \mathbf{y}, \quad \text{and} \quad f^{\{s\}}(t, \mathbf{y}) = \begin{pmatrix} 0 & 0 \\ g_{21} & g_{22} \end{pmatrix} \mathbf{y}.$$

Combining the fast stage updates,

$$\begin{aligned} \mathbf{k}_j^{\{f\}} &= y_n + \sum_{l=1}^{\mathbf{s}\{f\}} h * a_{jl}^{\{f,f\}} * f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) \\ &\quad + \sum_{l=1}^{\mathbf{s}\{s\}} h * a_{jl}^{\{f,s\}} * f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right), \end{aligned}$$

with this test problem makes

$$\begin{aligned} f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) &= \begin{pmatrix} g_{11} & g_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} k_{f,l}^{\{f\}} \\ k_{s,l}^{\{f\}} \end{pmatrix} \\ &= \begin{pmatrix} g_{11} k_{f,l}^{\{f\}} + g_{12} k_{s,l}^{\{f\}} \\ 0 \end{pmatrix} \\ f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) &= \begin{pmatrix} 0 & 0 \\ g_{21} & g_{22} \end{pmatrix} \begin{pmatrix} k_{f,l}^{\{s\}} \\ k_{s,l}^{\{s\}} \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ g_{21} k_{f,l}^{\{s\}} + g_{22} k_{s,l}^{\{s\}} \end{pmatrix}. \end{aligned}$$

So the fast stages for this test problem have the following update:

$$\begin{aligned} \mathbf{k}_j^{\{f\}} &= \mathbf{1}_{2s^{\{f\}}} \otimes \begin{pmatrix} y_{f,n} \\ y_{s,n} \end{pmatrix} + \sum_{l=1}^{s^{\{f\}}} h * a_{jl}^{\{f,f\}} * \begin{pmatrix} g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}} \\ 0 \end{pmatrix} \\ &\quad + \sum_{l=1}^{s^{\{s\}}} h * a_{jl}^{\{f,s\}} * \begin{pmatrix} 0 \\ g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}} \end{pmatrix} \end{aligned}$$

or equivalently,

$$\begin{aligned} \begin{pmatrix} k_{f,j}^{\{f\}} \\ k_{s,j}^{\{f\}} \end{pmatrix} &= \begin{pmatrix} y_{f,n} \\ y_{s,n} \end{pmatrix} + \begin{pmatrix} \sum_{l=1}^{s^{\{f\}}} h * a_{jl}^{\{f,f\}} * (g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}}) \\ \sum_{l=1}^{s^{\{s\}}} h * a_{jl}^{\{f,s\}} * (g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}}) \end{pmatrix} \\ &= \begin{pmatrix} y_{f,n} + \sum_{l=1}^{s^{\{f\}}} h * a_{jl}^{\{f,f\}} * (g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}}) \\ y_{s,n} + \sum_{l=1}^{s^{\{s\}}} h * a_{jl}^{\{f,s\}} * (g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}}) \end{pmatrix} \end{aligned}$$

Similarly for the slow stage updates:

$$\begin{aligned} \mathbf{k}_j^{\{s\}} &= \mathbf{1}_{2s^{\{s\}}} \otimes \begin{pmatrix} y_{f,n} \\ y_{s,n} \end{pmatrix} + \sum_{l=1}^{s^{\{f\}}} h * a_{jl}^{\{s,f\}} * \begin{pmatrix} g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}} \\ 0 \end{pmatrix} \\ &\quad + \sum_{l=1}^{s^{\{s\}}} h * a_{jl}^{\{s,s\}} * \begin{pmatrix} 0 \\ g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}} \end{pmatrix} \end{aligned}$$

or equivalently,

$$\begin{aligned} \begin{pmatrix} k_{f,j}^{\{s\}} \\ k_{s,j}^{\{s\}} \end{pmatrix} &= \begin{pmatrix} y_{f,n} \\ y_{s,n} \end{pmatrix} + \begin{pmatrix} \sum_{l=1}^{s^{\{f\}}} h * a_{jl}^{\{s,f\}} * (g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}}) \\ \sum_{l=1}^{s^{\{s\}}} h * a_{jl}^{\{s,s\}} * (g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}}) \end{pmatrix} \\ &= \begin{pmatrix} y_{f,n} + \sum_{l=1}^{s^{\{f\}}} h * a_{jl}^{\{s,f\}} * (g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}}) \\ y_{s,n} + \sum_{l=1}^{s^{\{s\}}} h * a_{jl}^{\{s,s\}} * (g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}}) \end{pmatrix} \end{aligned}$$

So the solution variable  $\mathbf{y}$  for this test problem has the following update:

$$\begin{aligned} \mathbf{y}_{n+1} &= \mathbf{y}_n + \sum_{l=1}^{\mathbf{s}\{f\}} h * b_l^{\{f\}} * f^{\{f\}} \left( t + c_l^{\{f\}} h, \mathbf{k}_l^{\{f\}} \right) + \sum_{l=1}^{\mathbf{s}\{s\}} h * b_l^{\{s\}} * f^{\{s\}} \left( t + c_l^{\{s\}} h, \mathbf{k}_l^{\{s\}} \right) \\ &= \mathbf{y}_n + \sum_{l=1}^{\mathbf{s}\{f\}} h * b_l^{\{f\}} * \begin{pmatrix} g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}} \\ 0 \end{pmatrix} + \sum_{l=1}^{\mathbf{s}\{s\}} h * b_l^{\{s\}} * \begin{pmatrix} 0 \\ g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}} \end{pmatrix} \end{aligned}$$

Given these equations, we can formulate the linear stability of this test problem in a similar way to how the original Runge Kutta stability problem is formulated:

$$\begin{aligned} \begin{pmatrix} k_{f,j}^{\{f\}} \\ k_{s,j}^{\{f\}} \end{pmatrix} &= \begin{pmatrix} y_{f,n} + \sum_{l=1}^{\mathbf{s}\{f\}} h * a_{jl}^{\{f,f\}} * \left( g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}} \right) \\ y_{s,n} + \sum_{l=1}^{\mathbf{s}\{s\}} h * a_{jl}^{\{f,s\}} * \left( g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}} \right) \end{pmatrix} \\ \begin{pmatrix} k_{f,j}^{\{s\}} \\ k_{s,j}^{\{s\}} \end{pmatrix} &= \begin{pmatrix} y_{f,n} + \sum_{l=1}^{\mathbf{s}\{f\}} h * a_{jl}^{\{s,f\}} * \left( g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}} \right) \\ y_{s,n} + \sum_{l=1}^{\mathbf{s}\{s\}} h * a_{jl}^{\{s,s\}} * \left( g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}} \right) \end{pmatrix} \\ \begin{pmatrix} y_{f,n+1} \\ y_{s,n+1} \end{pmatrix} &= \begin{pmatrix} y_{f,n} \\ y_{s,n} \end{pmatrix} + \sum_{l=1}^{\mathbf{s}\{f\}} h * b_l^{\{f\}} * \begin{pmatrix} g_{11}k_{f,l}^{\{f\}} + g_{12}k_{s,l}^{\{f\}} \\ 0 \end{pmatrix} + \sum_{l=1}^{\mathbf{s}\{s\}} h * b_l^{\{s\}} * \begin{pmatrix} 0 \\ g_{21}k_{f,l}^{\{s\}} + g_{22}k_{s,l}^{\{s\}} \end{pmatrix}. \end{aligned}$$

Converting these to use the matrix and vector form for the coefficients, we have

$$\begin{aligned} \begin{pmatrix} \mathbf{k}_f^{\{f\}} \\ \mathbf{k}_s^{\{f\}} \end{pmatrix} &= \begin{pmatrix} y_{f,n} \mathbf{1}_{s\{f\}} + hA^{\{f,f\}} \left( g_{11}\mathbf{k}_f^{\{f\}} + g_{12}\mathbf{k}_s^{\{f\}} \right) \\ y_{s,n} \mathbf{1}_{s\{f\}} + hA^{\{f,s\}} \left( g_{21}\mathbf{k}_f^{\{s\}} + g_{22}\mathbf{k}_s^{\{s\}} \right) \end{pmatrix} \\ \begin{pmatrix} \mathbf{k}_f^{\{s\}} \\ \mathbf{k}_s^{\{s\}} \end{pmatrix} &= \begin{pmatrix} y_{f,n} \mathbf{1}_{s\{s\}} + hA^{\{s,f\}} \left( g_{11}\mathbf{k}_f^{\{f\}} + g_{12}\mathbf{k}_s^{\{f\}} \right) \\ y_{s,n} \mathbf{1}_{s\{s\}} + hA^{\{s,s\}} \left( g_{21}\mathbf{k}_f^{\{s\}} + g_{22}\mathbf{k}_s^{\{s\}} \right) \end{pmatrix} \\ \begin{pmatrix} y_{f,n+1} \\ y_{s,n+1} \end{pmatrix} &= \begin{pmatrix} y_{f,n} + h\mathbf{b}^{\{f\}} \left( g_{11}\mathbf{k}_f^{\{f\}} + g_{12}\mathbf{k}_s^{\{f\}} \right) \\ y_{s,n} + h\mathbf{b}^{\{s\}} \left( g_{21}\mathbf{k}_f^{\{s\}} + g_{22}\mathbf{k}_s^{\{s\}} \right) \end{pmatrix}. \end{aligned}$$

We then substitute  $Z = hG = \begin{pmatrix} hg_{11} & hg_{12} \\ hg_{21} & hg_{22} \end{pmatrix}$  into the above result,

$$\begin{pmatrix} \mathbf{k}_f^{\{f\}} \\ \mathbf{k}_s^{\{f\}} \end{pmatrix} = \begin{pmatrix} y_{f,n} \mathbf{1}_{s\{f\}} + A^{\{f,f\}} (z_{11} \mathbf{k}_f^{\{f\}} + z_{12} \mathbf{k}_s^{\{f\}}) \\ y_{s,n} \mathbf{1}_{s\{f\}} + A^{\{f,s\}} (z_{21} \mathbf{k}_f^{\{s\}} + z_{22} \mathbf{k}_s^{\{s\}}) \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{k}_f^{\{s\}} \\ \mathbf{k}_s^{\{s\}} \end{pmatrix} = \begin{pmatrix} y_{f,n} \mathbf{1}_{s\{s\}} + A^{\{s,f\}} (z_{11} \mathbf{k}_f^{\{f\}} + z_{12} \mathbf{k}_s^{\{f\}}) \\ y_{s,n} \mathbf{1}_{s\{s\}} + A^{\{s,s\}} (z_{21} \mathbf{k}_f^{\{s\}} + z_{22} \mathbf{k}_s^{\{s\}}) \end{pmatrix}$$

$$\begin{pmatrix} y_{f,n+1} \\ y_{s,n+1} \end{pmatrix} = \begin{pmatrix} y_{f,n} + \mathbf{b}^{\{f\}} (z_{11} \mathbf{k}_f^{\{f\}} + z_{12} \mathbf{k}_s^{\{f\}}) \\ y_{s,n} + \mathbf{b}^{\{s\}} (z_{21} \mathbf{k}_f^{\{s\}} + z_{22} \mathbf{k}_s^{\{s\}}) \end{pmatrix}.$$

We then rearrange this result in order to solve for  $\mathbf{k}$ :

$$\begin{pmatrix} \mathbf{k}_f^{\{f\}} \\ \mathbf{k}_s^{\{f\}} \\ \mathbf{k}_f^{\{s\}} \\ \mathbf{k}_s^{\{s\}} \end{pmatrix} = \begin{pmatrix} y_{f,n} \mathbf{1}_{s\{f\}} + A^{\{f,f\}} (z_{11} \mathbf{k}_f^{\{f\}} + z_{12} \mathbf{k}_s^{\{f\}}) \\ y_{s,n} \mathbf{1}_{s\{f\}} + A^{\{f,s\}} (z_{21} \mathbf{k}_f^{\{s\}} + z_{22} \mathbf{k}_s^{\{s\}}) \\ y_{f,n} \mathbf{1}_{s\{s\}} + A^{\{s,f\}} (z_{11} \mathbf{k}_f^{\{f\}} + z_{12} \mathbf{k}_s^{\{f\}}) \\ y_{s,n} \mathbf{1}_{s\{s\}} + A^{\{s,s\}} (z_{21} \mathbf{k}_f^{\{s\}} + z_{22} \mathbf{k}_s^{\{s\}}) \end{pmatrix},$$

or equivalently

$$\begin{pmatrix} y_{f,n} \mathbf{1}_{s\{f\}} \\ y_{s,n} \mathbf{1}_{s\{f\}} \\ y_{f,n} \mathbf{1}_{s\{s\}} \\ y_{s,n} \mathbf{1}_{s\{s\}} \end{pmatrix} = \left[ \begin{pmatrix} I_{s\{f\}} & & & \\ & I_{s\{f\}} & & \\ & & I_{s\{s\}} & \\ & & & I_{s\{s\}} \end{pmatrix} - \begin{pmatrix} A^{\{f,f\}} z_{11} & A^{\{f,f\}} z_{12} & & \\ & & A^{\{f,s\}} z_{21} & A^{\{f,s\}} z_{22} \\ A^{\{s,f\}} z_{11} & A^{\{s,f\}} z_{12} & & \\ & & A^{\{s,s\}} z_{21} & A^{\{s,s\}} z_{22} \end{pmatrix} \right] \begin{pmatrix} \mathbf{k}_f^{\{f\}} \\ \mathbf{k}_s^{\{f\}} \\ \mathbf{k}_f^{\{s\}} \\ \mathbf{k}_s^{\{s\}} \end{pmatrix}.$$



Solving this equation for the stage vectors, we have

$$\begin{pmatrix} \mathbf{k}_f^{\{f\}} \\ \mathbf{k}_s^{\{f\}} \\ \mathbf{k}_f^{\{s\}} \\ \mathbf{k}_s^{\{s\}} \end{pmatrix} = \left[ \begin{pmatrix} I_{s\{f\}} & & & \\ & I_{s\{f\}} & & \\ & & I_{s\{s\}} & \\ & & & I_{s\{s\}} \end{pmatrix} - \begin{pmatrix} A^{\{f,f\}}_{z_{11}} & A^{\{f,f\}}_{z_{12}} & & \\ & & A^{\{f,s\}}_{z_{21}} & A^{\{f,s\}}_{z_{22}} \\ A^{\{s,f\}}_{z_{11}} & A^{\{s,f\}}_{z_{12}} & & \\ & & & A^{\{s,s\}}_{z_{21}} & A^{\{s,s\}}_{z_{22}} \end{pmatrix} \right]^{-1} \begin{pmatrix} y_{f,n} \mathbf{1}_{s\{f\}} \\ y_{s,n} \mathbf{1}_{s\{f\}} \\ y_{f,n} \mathbf{1}_{s\{s\}} \\ y_{s,n} \mathbf{1}_{s\{s\}} \end{pmatrix}.$$

Finally, we insert this result into our solution update equation and rearrange to obtain the equation:

$$S(Z) = \left[ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + B_z * (\mathbf{I} - A_z)^{-1} * \begin{pmatrix} \mathbf{1}_{s\{f\}} \\ \mathbf{1}_{s\{f\}} \\ \mathbf{1}_{s\{s\}} \\ \mathbf{1}_{s\{s\}} \end{pmatrix} \right] \quad (6.1)$$

where

$$B_z = \begin{pmatrix} \mathbf{b}^{\{f\}}_{z_{11}} & \mathbf{b}^{\{f\}}_{z_{12}} & & \\ & & \mathbf{b}^{\{s\}}_{z_{21}} & \mathbf{b}^{\{s\}}_{z_{22}} \end{pmatrix},$$

$$A_z = \begin{pmatrix} A^{\{f,f\}}_{z_{11}} & A^{\{f,f\}}_{z_{12}} & & \\ & & A^{\{f,s\}}_{z_{21}} & A^{\{f,s\}}_{z_{22}} \\ A^{\{s,f\}}_{z_{11}} & A^{\{s,f\}}_{z_{12}} & & \\ & & & A^{\{s,s\}}_{z_{21}} & A^{\{s,s\}}_{z_{22}} \end{pmatrix}, \quad \text{and}$$

$$\mathbf{I} = \begin{pmatrix} I_{s\{f\}} & & & \\ & I_{s\{f\}} & & \\ & & I_{s\{s\}} & \\ & & & I_{s\{s\}} \end{pmatrix}.$$

## 6.2. Comparing Stability for Base Methods with High Order

Figures 6.1–6.4 directly compare between the stability plots for the two different base methods which satisfy both Equation (2.28) and Equation (2.51), and therefore make Algorithm 3 3rd order and Algorithm 5 4th order. We are interested in studying all 4 of these sets of stability plots in order to thoroughly investigate comparisons of stability for Algorithm 3 and Algorithm 5.

The following plots demonstrate the situation where the problem-based time-scale separation  $\kappa$  is the same as the method-based time-scale separation. Results are shown for  $\kappa = m = 10$  and  $\kappa = m = 100$  to demonstrate the expected scaling relationship along the  $\xi$  axis. Differences in where the methods are stable are also more easily observed in Figure 6.1 and Figure 6.3 where  $\kappa = m = 10$ . First, in Figure 6.1-6.2 we compare the stability plots for the RMIS and the RFSMR using the 3/8 – Rule from Section 5.2 as shown in Equation (4.17). We recall that this base method has equally-spaced stage times  $\mathbf{c}$ , and automatically makes the RMIS 4th order and the RFSMR 3rd order. Figure 6.1 investigates  $\kappa = 10$  and this particular base method. We see that the location of the added stability is similar, and the most negative  $\xi$  for  $-.4 < \eta < 1$  where the multirate method will be stable is also similar.

Figure 6.2 investigates  $\kappa = 100$  and this particular base method. We see that the location of the added stability is similar, and the most negative  $\xi$  for  $-.6 < \eta < 1$  where the multirate method will be stable is also similar. The stability results for  $\kappa = 100$  are shown in order to demonstrate how relatively insignificant the extra stability is for  $\xi < -.03$  as well as to show stability properties for the methods tested in Chapter 5.

Next, in Figure 6.3-6.4 we compare the stability plots for our RMIS method and the RFSMR method using the base method we found by a symbolic solve for the intersections of the order graphs from Figure 4.3 which satisfies the conditions so that both the RMIS and the RFSMR have their maximum order for a 4-stage method. The specific numerical values for this base method are described by Equations (6.2)-(6.15).

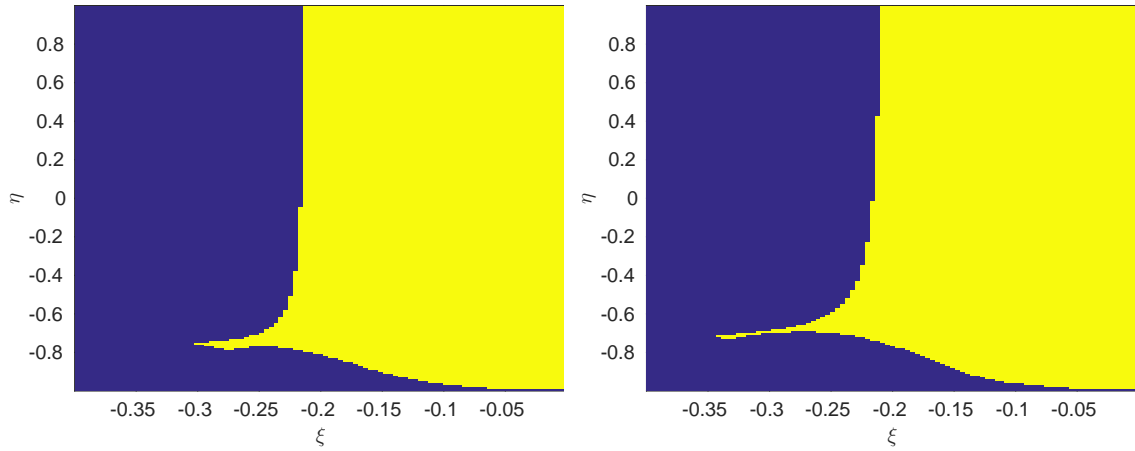


Figure 6.1. Linear stability plots for time-scale separation and multirate-method factors  $\kappa = m = 10$  using  $3/8 - \text{Rule}$ : our RMIS method is on the left and the RFSMR method is on the right.

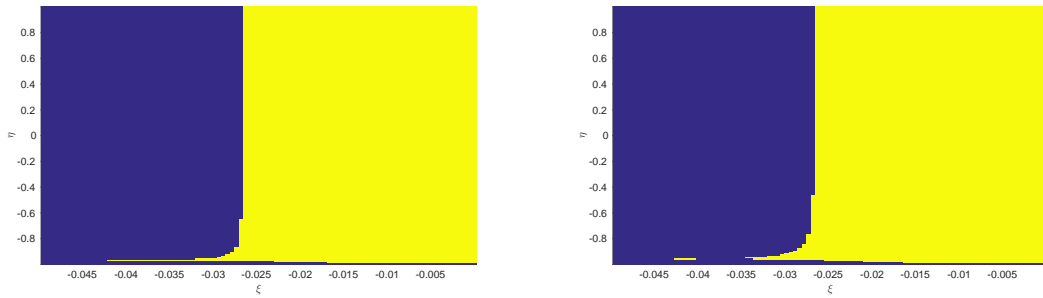


Figure 6.2. Linear stability plots for time-scale separation and multirate-method factors  $\kappa = m = 100$ , using  $3/8 - \text{Rule}$ : our RMIS method is on the left and the RFSMR method is on the right.

$$a_{2,1} = \frac{2502984374488603}{9007199254740992}, \quad (6.2)$$

$$a_{3,1} = -\frac{2049798293874988098690044461751131390159023391}{7517271464469377352257191894615983679002902528}, \quad (6.3)$$

$$a_{3,2} = \frac{3018117031057894889503793574809}{3338339144884628840629222376986}, \quad (6.4)$$

$$a_{4,1} = \frac{22765692084681232760997989792354552558780331504607913975583345}{19998492846806357615697413355127699632803177178677941346341489}, \quad (6.5)$$

$$a_{4,2} = -\frac{94926537801716237058245396328766122453369223328055560635416576}{67181335943891087403606309330764345048833854489406728444975161}, \quad (6.6)$$

$$a_{4,3} = \frac{32427491595433235837136108367314853114729405718844372760395776}{25440922267341045412987192917951158593602244095128793560304173}, \quad (6.7)$$

$$b_1 = \frac{527733938855414179796741258573}{4744937406134690332764136798874}, \quad (6.8)$$

$$b_2 = \frac{48009933710805118264642043178729372929801322496}{155513464357343682343746756582997662207410178171}, \quad (6.9)$$

$$b_3 = \frac{6762883003043953633003226591911056199068418048}{15030509768155253218042489127098340903297421993}, \quad (6.10)$$

$$b_4 = \frac{8429402175443019162849978345397}{64783217009677962262953073299306}, \quad (6.11)$$

$$c_1 = 0, \quad (6.12)$$

$$c_2 = \frac{2502984374488603}{9007199254740992} \approx 0.27788708828342423285, \quad (6.13)$$

$$c_3 = \frac{2843567935040037}{4503599627370496} \approx 0.63139891871345210639, \quad (6.14)$$

$$c_4 = 1. \quad (6.15)$$

Figure 6.4 investigates  $\kappa = 100$  and this particular base method. We see that the location of the added stability is similar, and the most negative  $\xi$  for  $-.8 < \eta < 1$  where the multirate method will be stable is also similar. The stability results for  $\kappa = 100$  are shown in order to demonstrate how relatively insignificant the extra stability is for  $\xi < -.03$  as well as to show stability properties for the methods tested in Chapter 5.

Recall that the intent of Figures 6.1–6.4 is to determine how our newly developed methods compare in terms of stability to the existing method we compare to in our numerical order

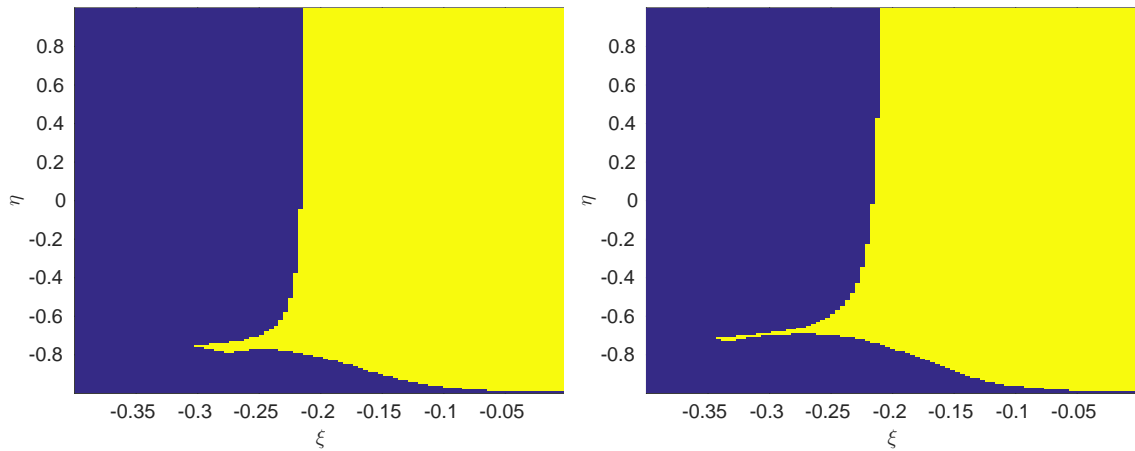


Figure 6.3. Linear stability plots for time-scale separation and multirate-method factors  $\kappa = m = 10$ : our RMIS method is on the left and the RFSMR method is on the right. Note that the stability areas are slightly larger than in Figure 6.1, although they have similar shape, indicating that multirate problems can potentially take a larger time-step due to the increased region of stability. The different placement of the region which is extended indicates that some multirate problems which would be stable for  $\kappa = 10$  using the  $3/8$  – Rule would not be stable for this method, and vice versa.

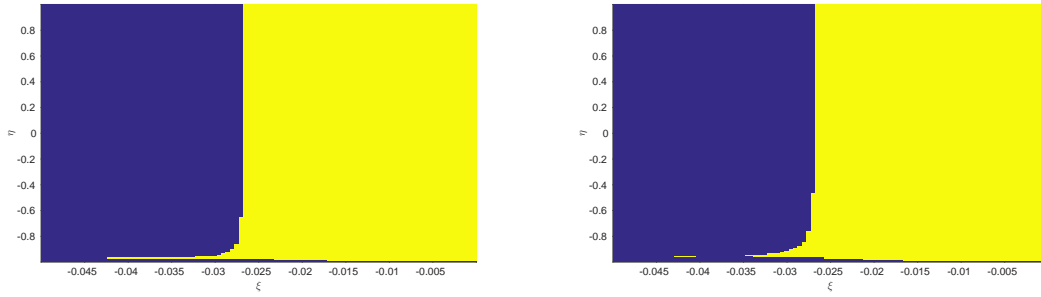


Figure 6.4. Linear stability plots for time-scale separation and multirate-method factors  $\kappa = m = 10$ : our RMIS method is on the left and the RFSMR method is on the right. Note that the stability areas are slightly larger than in Figure 6.2, although they have similar shape, indicating that multirate problems can potentially take a larger time-step due to the increased region of stability. The different placement of the region which is extended indicates that some multirate problems which would be stable for  $\kappa = 100$  using the 3/8 – Rule would not be stable for this method, and vice versa.

tests in Chapter 5. Specifically, we wanted to compare the stability for multirate methods which used the same base method. We chose base methods where the order of accuracy for the RMIS was 4 and the order of accuracy for the RFSMR was 3. The result of this test was that the RMIS method performed better than the RFSMR method.

### 6.3. Comparing Stability Optimization

We performed another set of tests to investigate the relative size of the stability area for many different choices of base method. We chose to use  $\kappa = 10$  and used  $c_2 \leq c_3$  for base methods of the form described by Equations (4.2)-(4.11) which satisfy Equation (2.28) for Algorithm 3 and which satisfy Equation (2.51) for Algorithm 5. These are preliminary results which attempt to quantify what percentage of the stability region displayed is stable by adding up the number of connected stable “boxes” and dividing by the total number of stable “boxes.” This can be shown in a basic way by making a heat map of the intensity of the base methods tested. Figure 6.5 is a preliminary attempt to display this information.

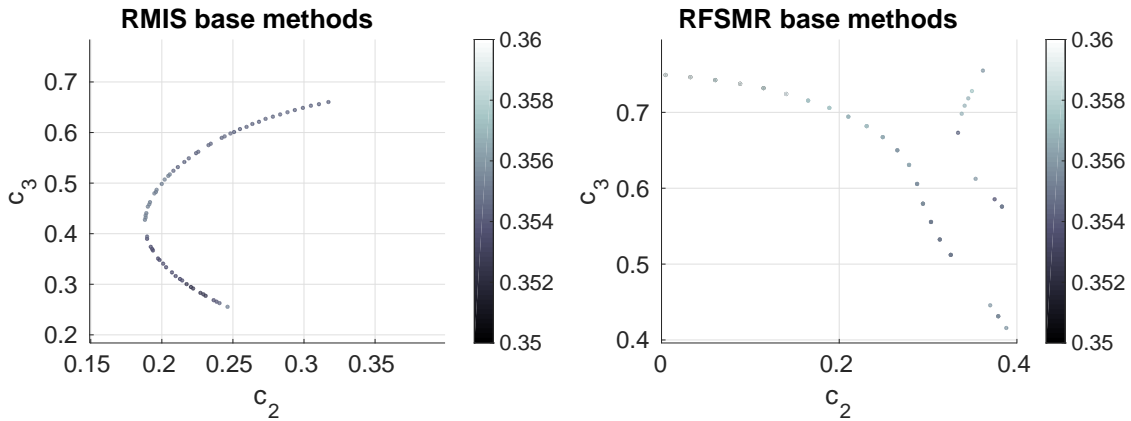


Figure 6.5. For most of the base methods tested, the RMIS methods had a stable area around 35.5% of the plotted area. The RFSMR methods, on the other hand, had a more varied percentage area, going from 35.2% to 35.8%. The total percentage area did not seem to favor either the RMIS or the RFSMR in terms of stability. We used these sets of base methods to pick a base method for both the RMIS and the RFSMR in order to compare the largest area of stability.

In Figure 6.6-6.7 we compare our RMIS method with the largest stability area we found against the RFSMR method with the largest stability area we found for  $\kappa = 10$  and  $\kappa = 100$  respectively. This is a slightly unfair comparison, since we're considering 4-stage methods here, and the RFSMR method cannot use the extra stage flexibility to increase beyond order 3.

Recall that the intent of Figures 6.6–6.7 is to determine how our newly developed RMIS methods compare in terms of stability to the existing RFSMR methods. Specifically, we wanted to compare the stability for multirate methods which have an optimized stability region by selecting the largest stability region from the set tested. We chose base methods where the order of accuracy for the RMIS was 4 and the order of accuracy for the RFSMR was 3. The result of this test was that the RMIS method performed more consistently than the RFSMR method, although for some specific base methods the RFSMR method could have a larger stability area.

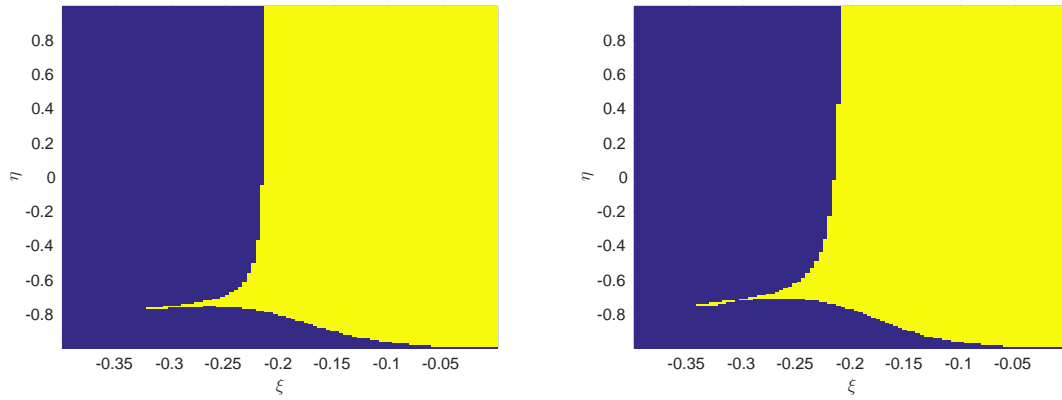


Figure 6.6. Same  $\kappa = m = 10$ , best stability area RMIS on the left and best stability area RFSMR on the right

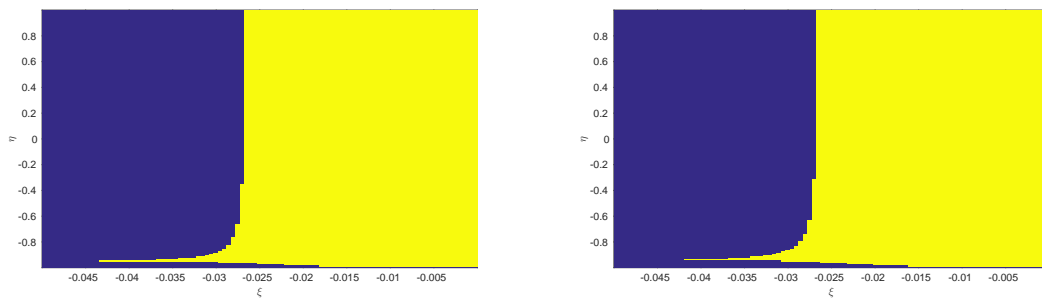


Figure 6.7. Same  $\kappa = m = 100$ , best stability area RMIS on the left and best stability area RFSMR on the right



From these stability studies, the RFSMR method and the RMIS method have similar stability regions, although not exactly the same. They also scale their stability regions in similar ways when the time-scale separation is increased. We have shown here that the improved order of accuracy obtained from RMIS over RFSMR does not hinder stability whatsoever.

## Chapter 7

### Conclusions

In conclusion, this dissertation demonstrates our newly developed fourth order multirate methods. These methods are based on Multirate Infinitesimal Step Methods. In Chapter 1, we introduce basic multirate concepts and basic historical content. In Chapter 2, we discuss more detailed background on the existing methods and existing Runge-Kutta theory which these new methods extend from. In Chapter 3 we describe the structure we developed for Flexible Multirate GARK methods, and give some related proofs. In Chapter 4 we discuss particular fourth-order implementations of that structure, including our development of the Relaxed Multirate Infinitesimal Step Methods. In Chapter 5 we investigated the numerical order properties of our methods. In Chapter 6 we investigate the stability properties of our newly developed methods compared to what we found was the most efficient multirate method tested on our test problems.

More specifically, in Chapter 1, we described the broad context of multirate methods for time integration. Multirate methods aim to more efficiently solve problems which have functions or processes that have differing characteristic time scales yet still have meaningful coupling between them. The first recorded example of the simplest multirate method is a 1980 preprint by Gear [15]. Advances in multirate integrators in the 1990s which remain commonly cited today mostly focused on examining multirate stability, and restrict discussions on the order of accuracy to specific problem and method contexts [30, 53, 58]. More complicated and higher order multirate integrators began to be developed in the early 2000s, specifically focusing on order of accuracy and addressing coupling error through interpolation [11, 34, 33, 18]. Most advances in the field of multirate integrators in the last decade have been in increasing efficiency by increasing the overall method order [7, 8, 9, 14, 21, 44, 47, 46, 50, 51, 57].

More specifically, in Chapter 2, we discussed the GARK framework which Sandu and Günther first introduced in 2013, as well as the more thoroughly developed Runge-Kutta based Multirate Infinitesimal Step and Recursive Flux-Splitting Multirate methods. MIS methods have been interpreted as exponential integrators and as ARK methods, and the Multirate GARK paper by Sandu and Günther uses GARK theory to account for order of accuracy, including coupling error [20]. The GARK framework has robust theory regarding order of accuracy conditions, and allows for more straightforward linear stability analysis for multirate methods. One benefit of developing multirate methods using GARK theory is the standardization in approach towards interpolation between the fast process and the slow process. Another benefit is the strong basis in Runge-Kutta theory this brings to the coupling error conditions. Finally, using GARK theory reduces the extra initial work to implement multirate methods which rely on base Runge Kutta methods. Multirate Infinitesimal Step methods were extended with a specific splitting or partitioning concept by Schlegel as Recursive-Flux Splitting Multirate methods. These methods are telescopic, and have at best third order and at worst second order accuracy. In order for RFSMR methods to be third-order, a specific condition based on the slow base method must be satisfied. These methods can be implemented so that the time-scale separation is a parameter of the method which can be changed to suit a particular problem’s time-scale separation or be changed dynamically between macro-steps during integration. These methods are constructed from base Runge Kutta methods, and can be subcycled as described by Schlegel [49].

More specifically, in Chapter 3 we develop a structure based on GARK and on MIS methods called fmGARK. fmGARK methods can be specifically determined depending on which base methods are chosen, on the number of subcycles, and on the final fast solution weighting vector  $\mathbf{b}^{\{f\}}$ . We show that the RFSMR is an example of an fmGARK method. We discuss pseudocode and implementation details for fmGARK methods, as well as memory-efficiency. Specifically, in order to avoid recomputing fast and slow function calls, fmGARK methods require at most  $s^{\{I\}} + s^{\{O\}} + 2$  vectors stored of size  $y$ . The algorithm which allows this efficiency-saving measure is described in the pseudocode, which include subcycling. We

also discuss how the fmGARK structure satisfies the slow order conditions up to order four given base methods of at least third order, and a specific condition based on the slow base method must be satisfied. We also discuss briefly how the fast order conditions up to order three apply to the fmGARK structure in general, and the RFSMR in particular.

In Chapter 4 we discuss two specific families of fmGARK methods we have developed, and their associated order conditions. The Relaxed Multirate Infinitesimal Step methods are based on the MIS methods, but relax the assumption about how  $\mathbf{b}^{\{f\}}$  is constructed. The same weights  $\mathbf{b}^{\{f\}}$  are used as  $\mathbf{b}^{\{s\}}$  at the same stage times, but the stage vectors are linear combinations of the fast function values at all preceding stage times and linear combinations of the slow function values at all preceding stage times respectively. RMIS methods have the benefits RFSMR and MIS methods have which were passed on to the fmGARK structure. In addition, RMIS methods have the benefit that each fast coupling order conditions is satisfied when the corresponding slow coupling order condition is satisfied. The structure of the fmGARK forces the slow order conditions (including the slow coupling order conditions) up to third order to be automatically satisfied if the base methods are at least third order. So RMIS methods are at worst third order with third order base methods. If a specific condition based on the slow base method is satisfied, the only slow coupling condition at order four which is not automatically satisfied by the fmGARK structure will be satisfied. This then forces the corresponding fast coupling conditions up to fourth order to be satisfied, and therefore given this condition, the RMIS method will be fourth order. We include equations and plots for coefficient choices based on the minimal number of base method stages while getting third order RFSMR or fourth order RMIS. Another possible family of fmGARK methods are methods which are optimized for a particular target objective. We develop methods optimized for a specific time-scale separation which minimize the fifth-order error coefficients. Other optimization goals are possible. Since a RFSMR is a specific instance of an fmGARK, we can define an embedded solution for any fmGARK method based on the RFSMR solution. The RFSMR solution is automatically generated while progressing through the fmGARK step.

In Chapter 5 we introduce three test problems and our testing methodology. We have a nonlinear coupled multirate problem, a linear strongly coupled multirate problem, and a linear weakly couple multirate problem. We show convergence plots which suggest that the numerical order calculated based on our implementation of the multirate methods tested matches the theoretical order predicted. We show efficiency plots which suggest that the higher order methods are more efficient for tighter error tolerances. We see more benefit from higher-order methods when taking smaller step-sizes (for smaller  $h$  values) partly because of the asymptotic nature of order of convergence. In Chapter 6 we performed a linear stability study by leveraging GARK theory and using a 2 by 2 linear system stability test problem which has been used for examining multirate stability since it was first introduced by Kvaerno in 2000 [33]. These tests showed that our newly created methods performed no worse than the RFSMR methods which are already in use that we use for comparison.

Possible future directions of inquiry include using embedded solutions to create an adaptive time-step size strategy. Another possible future direction is to consider using a fast base method which is implicit, but has an explicit first stage. Another possible future direction is to investigate numerical results for methods with more than two time-scales. Another possible future direction may include developing multirate methods in which the fast base method changes within an integration step, and multirate methods which substantially change the initial condition for each fast sub-step.

## Bibliography

- [1] ASCHER, U. M., RUUTH, S. J., AND SPITERI, R. J. Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics* 25, 2 (1997), 151–167.
- [2] BARTEL, A., AND GÜNTHER, M. A multirate W-method for electrical networks in statespace formulation. *Journal of Computational and Applied Mathematics* 147, 2 (Oct. 2002), 411–425.
- [3] BOGACKI, P., AND SHAMPINE, L. F. A 3 (2) pair of runge-kutta formulas. *Applied Mathematics Letters* 2, 4 (1989), 321–325.
- [4] BREMICKER-TRÜBELHORN, S., AND ORTLEB, S. On Multirate GARK Schemes with Adaptive Micro Step Sizes for FluidStructure Interaction: Order Conditions and Preservation of the Geometric Conservation Law. *Aerospace* 4, 1 (Feb. 2017), 8.
- [5] BUTCHER, J. C. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008.
- [6] CELLEDONI, E., MARTHINSEN, A., AND OWREN, B. Commutator-free Lie group methods. *Future Generation Computer Systems* 19, 3 (2003), 341–352.
- [7] CONSTANTINESCU, E. M., AND SANDU, A. Multirate timestepping methods for hyperbolic conservation laws. *Journal of Scientific Computing* 33, 3 (2007), 239–278.
- [8] CONSTANTINESCU, E. M., AND SANDU, A. Extrapolated implicit-explicit time stepping. *SIAM J.Sci.Comput* 31, 6 (2010), 4452–4477.
- [9] CONSTANTINESCU, E. M., AND SANDU, A. Extrapolated multirate methods for differential equations with multiple time scales. *Journal of Scientific Computing* 56, 1 (2013), 28–44.
- [10] DORMAND, J. R., AND PRINCE, P. J. Practical RungeKutta Processes. *SIAM Journal on Scientific and Statistical Computing* 10, 5 (1989), 977–989.

- [11] ENGSTLER, C., AND LUBICH, C. Mur8: a multirate extension of the eighth-order dormand-prince method. *Applied Numerical Mathematics* 25, 2 (1997), 185–192.
- [12] 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 (2008), 1116–1146.
- [13] ESTEP, D., GINTING, V., AND TAVENER, S. A Posteriori analysis of a multirate numerical method for ordinary differential equations. *Computer Methods in Applied Mechanics and Engineering* 223 (2012), 10–27.
- [14] FOK, P. W. A Linearly Fourth Order Multirate RungeKutta Method with Error Control. *Journal of Scientific Computing* 66 (2016), 177–195.
- [15] GEAR, C. Multirate methods for ordinary differential equations. Tech. Rep. UIUCDCS-F-74-880, Department of Computer Sciences, University of Illinois, 1980.
- [16] GEAR, C. W., AND WELLS, D. Multirate linear multistep methods. *BIT Numerical Mathematics* 24, 4 (1984), 484–502.
- [17] GÜNTHER, M., AND HOSCHEK, M. Partitioning Strategies in Circuit Simulation. 1999, pp. 343–352.
- [18] GÜNTHER, M., KVÆRNØ, A., AND RENTROP, P. Multirate partitioned runge-kutta methods. *Bit Numerical Mathematics* 41, 3 (2001), 504–514.
- [19] GÜNTHER, M., AND RENTROP, P. Mathematical modelling and simulation of electrical circuits and semiconductor devices. In *Mathematical modelling and simulation of electric circuits and semiconductor devices: proceedings of a conference held at the Mathematisches Forschung*. Birkhauser, 1990, pp. 33–60.
- [20] GÜNTHER, M., AND SANDU, A. Multirate generalized additive Runge Kutta methods. *Numerische Mathematik* 133, 540 (2013), 497–524.

- [21] GÜNTHER, M., AND SANDU, A. Multirate generalized additive Runge Kutta methods. *Numerische Mathematik* 133, 3 (July 2016), 497–524.
- [22] HAIRER, E., NØRSETT, S. P., AND WANNER, G. *Solving ordinary differential equations*, vol. 8-. Springer-Verlag, New York; Berlin, 1993.
- [23] HAIRER, E., AND OSTERMANN, A. Dense output for extrapolation methods. *Numerische Mathematik* 58 (1990), 419–439.
- [24] HUNSDORFER, W., AND SAVCENCO, V. Analysis of a multirate theta-method for stiff ODEs. *Applied Numerical Mathematics* 59, 3 (2009), 693–706.
- [25] KENNEDY, C. A., AND CARPENTER, M. H. Additive rungekutta schemes for convectiondiffusionreaction equations. *Applied Numerical Mathematics* 44, 1 (2003), 139–181.
- [26] KEYES, D., MCINNES, L., 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., RANGLES, A., REYNOLDS, D., RIVIERE, B., RUDE, 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. *INTERNATIONAL JOURNAL OF HIGH PERFORMANCE COMPUTING APPLICATIONS* 27, 1 (2013), 4–83.
- [27] KLEMP, J. B., AND WILHELMSON, R. B. *The Simulation of Three-Dimensional Convective Storm Dynamics*, 1978.
- [28] KNOTH, O., AND ARNOLD, M. Numerical solution of multiscale problems in atmospheric modeling. *Applied numerical mathematics* 62 (2012).



- [29] KNOTH, O., AND WENSCH, J. Generalized split-explicit runge-kutta methods for the compressible euler equations. *Monthly Weather Review* 142, 5 (2014), 2067.
- [30] KNOTH, O., AND WOLKE, R. Implicit-explicit runge-kutta methods for computing atmospheric reactive flows. *Applied Numerical Mathematics* 28, 2 (1998), 327–341.
- [31] KUHN, K., AND LANG, J. Comparison of the asymptotic stability for multirate Rosenbrock methods. *Journal of Computational and Applied Mathematics* 262 (2014), 139–149.
- [32] KUTTA, W. Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys.* 46 (1901), 435–453.
- [33] KV ÆRNO, A. Stability of multirate runge-kutta schemes. *International Journal of Differential Equations and Applications* 1 (2000), 97–105.
- [34] KVAERNØ, A., AND RENTROP, P. Low order multirate runge-kutta methods in electric circuit simulation.
- [35] LAMBERT, J. D. *Computational methods in ordinary differential equations*. Wiley, New York; London, 1973.
- [36] OLIVEIRA, J. F., AND PEDRO, J. C. Radio frequency numerical simulation techniques based on multirate Runge-Kutta schemes. *Journal of Applied Mathematics* 2012 (2012).
- [37] PRIGOGINE, I., AND NICOLIS, G. On SymmetryBreaking Instabilities in Dissipative Systems. *The Journal of Chemical Physics* 46, 9 (1967), 3542–3550.
- [38] REYNOLDS, D. ARKode: A library of high order implicit/explicit methods for multi-rate problems. SIAM Conference on Parallel Processing for Scientific Computing.
- [39] REYNOLDS, D. The ARKode Solver. <http://faculty.smu.edu/reynolds/ARKode/>, 2014.

- [40] RUNGE, C., AND KONIG, H. *Vorlesungen über numerisches rechnen*. J. Springer, Berlin, 1924.
- [41] SANDU, A., AND CONSTANTINESCU, E. M. Multirate Explicit Adams Methods for Time Integration of Conservation Laws. *J Sci Comput* 38 (2009), 229–249.
- [42] SANDU, A., AND GÜNTHER, M. A class of generalized additive runge-kutta methods. Tech. Rep. CSL-TR-5/2013, Computational Science Laboratory, Virginia Tech, 2013.
- [43] SANDU, A., AND GÜNTHER, M. Multirate generalized additive runge kutta methods. Tech. Rep. CSL-TR-6/2013, Computational Science Laboratory, Virginia Tech, 2013.
- [44] SANDU, A., AND GÜNTHER, M. A generalized-structure approach to additive runge–kutta methods. *SIAM Journal on Numerical Analysis* 53, 1 (2015), 17–42.
- [45] SAVCENCO, V. Comparison of the asymptotic stability properties for two multirate strategies. *Journal of Computational and Applied Mathematics* 220, 1-2 (Oct. 2008), 508–524.
- [46] SAVCENCO, V. Construction of a multirate RODAS method for stiff ODEs. *Journal of Computational and Applied Mathematics* 225, 2 (Mar. 2009), 323–337.
- [47] SAVCENCO, V., HUNSDORFER, W., AND VERWER, J. G. A multirate time stepping strategy for stiff ordinary differential equations. *BIT Numerical Mathematics* 47, 1 (2007), 137–155.
- [48] SAVCENCO, V., AND MATTHEIJ, R. M. M. Multirate Numerical Integration for Stiff ODEs. *Progress in Industrial Mathematics at ECMI ...* (2010), 1–6.
- [49] SCHLEGEL, M. *A Class of General Splitting Methods for Air Pollution Models: Theory and Practical Aspects*. PhD thesis, 2012.

- [50] SCHLEGEL, M., KNOTH, O., ARNOLD, M., AND WOLKE, R. Multirate rungekutta schemes for advection equations. *Journal of Computational and Applied Mathematics* 226, 2 (2009), 345–357.
- [51] SCHLEGEL, M., KNOTH, O., ARNOLD, M., AND WOLKE, R. Implementation of multirate time integration methods for air pollution modelling. *Geoscientific Model Development* 5 (2012), 1395–1405.
- [52] SCHLEGEL, M., KNOTH, O., ARNOLD, M., AND WOLKE, R. Numerical solution of multiscale problems in atmospheric modeling. *Applied Numerical Mathematics* 62, 10 (2012), 1531–1543.
- [53] SKAMAROCK, W. C., AND KLEMP, J. B. The stability of time-split numerical methods for the hydrostatic and the nonhydrostatic elastic equations. *Monthly Weather Review* 120, 9 (1992), 2109–2127.
- [54] STRIEBEL, M., AND GÜNTHER, M. Hierarchical mixed multirating in circuit simulation. In *Scientific Computing in Electrical Engineering SCEE 2006* (2007), no. 11, pp. 221–228.
- [55] VERHOEVEN, A. *Redundancy Reduction of IC Models by Multirate Time-Integration and Model Order Reduction*. 2008.
- [56] WELLS, D. R. *MULTIRATE LINEAR MULTISTEP METHODS FOR THE SOLUTION OF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS*. PhD thesis, 1982.
- [57] WENSCH, J., KNOTH, O., AND GALANT, A. Multirate infinitesimal step methods for atmospheric flow simulation. *BIT Numerical Mathematics* 49, 2 (2009), 449–473.
- [58] WICKER, L. J., AND SKAMAROCK, W. C. A time-splitting scheme for the elastic equations incorporating second-order runge-kutta time differencing. *Monthly Weather Review* 126, 7 (1998), 1992–1999.

- [59] ZHANG, H., SANDU, A., AND BLAISE, S. Partitioned and implicitexplicit general linear methods for ordinary differential equations. *Journal of Scientific Computing* 61, 1 (2014), 119–144.