



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

**EXTRAPOLATED MULTIRATE METHODS FOR
HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS**

by

Patrick Mugg

September 2021

Dissertation Supervisors:

Francis X. Giraldo
Jeremy E. Kozdon

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 2021	3. REPORT TYPE AND DATES COVERED Dissertation		
4. TITLE AND SUBTITLE EXTRAPOLATED MULTIRATE METHODS FOR HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS			5. FUNDING NUMBERS	
6. AUTHOR(S) Patrick Mugg				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Within atmospheric earth system models, accurately resolving the various physical multi-scale processes is critical. For example, the single-layered shallow water equations are widely used to model ocean waves in shallow depth, which contain both fast gravity and slow advective waves. In this dissertation we describe the development of an extrapolated multirate time-integration scheme using explicit Runge-Kutta methods for advancing the numerical solutions of hyperbolic partial differential equations with multiple processes. Our time-integration technique is based on Richardson extrapolation and will be tested using high-order accurate continuous and discontinuous Galerkin methods with an upwind-biased Rusanov flux. The benefit in developing an extrapolated multirate time-integration method is that we can solve the fast and slow processes using different time-steps with high order accuracy. Results are shown for the non-linear shallow water equations in conservation form and the non-hydrostatic, atmospheric Euler equations. We further analyze and compare the extrapolated multirate method against other single-rate and multirate time-splitting schemes for problems with multiple time scales.				
14. SUBJECT TERMS Richardson extrapolation, Gragg extrapolation, continuous Galerkin methods, discontinuous Galerkin methods, time-integration, multirate infinitesimal step methods, partial differential equations, shallow water equations, Euler equations, Runge-Kutta methods, Nonhydrostatic Unified Model of the Atmosphere			15. NUMBER OF PAGES 175	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**EXTRAPOLATED MULTIRATE METHODS FOR HYPERBOLIC PARTIAL
DIFFERENTIAL EQUATIONS**

Patrick Mugg
Lieutenant Colonel, United States Army
BS, U.S. Military Academy, 2003

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN APPLIED MATHEMATICS

from the

**NAVAL POSTGRADUATE SCHOOL
September 2021**

Approved by:	Francis X. Giraldo Department of Applied Mathematics Dissertation Supervisor and Chair	Jeremy E. Kozdon Department of Applied Mathematics Dissertation Supervisor
	Carlos F. Borges Department of Applied Mathematics	Wei Kang Department of Applied Mathematics
	Mara S. Orescanin Department of Oceanography	Hong Zhou Department of Applied Mathematics

Approved by: Francis X. Giraldo
Chair, Department of Applied Mathematics

Michael E. Freeman
Vice Provost of Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Within atmospheric earth system models, accurately resolving the various physical multi-scale processes is critical. For example, the single-layered shallow water equations are widely used to model ocean waves in shallow depth, which contain both fast gravity and slow advective waves. In this dissertation we describe the development of an extrapolated multirate time-integration scheme using explicit Runge-Kutta methods for advancing the numerical solutions of hyperbolic partial differential equations with multiple processes.

Our time-integration technique is based on Richardson extrapolation and will be tested using high-order accurate continuous and discontinuous Galerkin methods with an upwind-biased Rusanov flux. The benefit in developing an extrapolated multirate time-integration method is that we can solve the fast and slow processes using different time-steps with high order accuracy. Results are shown for the non-linear shallow water equations in conservation form and the non-hydrostatic, atmospheric Euler equations. We further analyze and compare the extrapolated multirate method against other single-rate and multirate time-splitting schemes for problems with multiple time scales.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1	Introduction and Background	1
1.1	Continuous PDE System	2
1.2	Overview of Element-Based Galerkin (EBG) Methods	6
1.3	Theory of Extrapolation Methods for Time Integration	13
1.4	Multirate Extrapolation Literature Review	40
2	Single-Rate and Multirate Runge-Kutta Methods	47
2.1	Single-Rate Explicit Runge-Kutta (ERK) and Additive RK (ARK) Methods .	47
2.2	Single-Rate ERK and ARK Results	51
2.3	Multirate ERK and ARK Methods	54
2.4	Multirate ERK and ARK Results	59
3	Extrapolated Multirate Methods	69
3.1	Multirate Infinitesimal Step (MIS) ERK Method with Extrapolation	69
3.2	MIS ERK Method with Extrapolation Results	70
3.3	Extrapolated MIS ERK Method with Substepping	78
3.4	Extrapolated MIS ERK with Substepping Results	81
3.5	Parallelizing the Extrapolated MIS ERK Methods	85
4	Nonhydrostatic Unified Model of the Atmosphere (NUMA)	93
4.1	Governing Equations for the 2D Shallow Water Model	93
4.2	NUMA 2D Shallow Water Results	96
4.3	Governing Equations for the Nonhydrostatic Navier-Stokes (Euler) Equations	108
4.4	NUMA 2D Euler Results	111
4.5	NUMA 3D Euler Results	116
5	Conclusion	121

Appendix A	Butcher Tableau Coefficients for Single-Rate ERK, ARK and LSRK Methods	123
Appendix B	Analytic Solution to 1D Shallow Water Equations	129
Appendix C	Derivation of Linearly Implicit Euler Method	133
Appendix D	Numerical Results of Non-Linear SWE	135
	List of References	141
	Initial Distribution List	147

List of Figures

Figure 1.1.1	Two Dimensional Shallow Water Diagram.	3
Figure 1.3.1	Work-Precision Plot for Linear SWE Using Explicit-Euler (Richardson) vs. Explicit-Midpoint (Gragg) Extrapolation.	30
Figure 1.3.2	Work-Precision Plot for Non-Linear SWE Using Richardson vs. Gragg Extrapolation.	31
Figure 1.3.3	Extrapolation Work-Precision Plot for Non-Linear SWE using Inexact DG, with 32^{nd} Order Polynomials.	32
Figure 1.3.4	Order of Accuracy / Convergence Rates Plot for Linear SWE Using Richardson vs. Gragg Extrapolation.	34
Figure 1.3.5	Order of Accuracy / Convergence Rates Plot for Non-Linear SWE Using Richardson vs. Gragg Extrapolation.	35
Figure 1.3.6	Stability Regions for P^{th} Order Richardson and Gragg Extrapolation Methods.	39
Figure 2.2.1	Error Plots and Total RHS Evaluations for Non-Linear SWE Using Single-Rate ARK & ERK Methods ($g = h_B = 1$).	52
Figure 2.2.2	Error Plots and Total RHS Evaluations for Non-Linear SWE Using Single-Rate ARK & ERK Methods ($g = 10, h_B = 1$).	53
Figure 2.3.1	Multirate Infinitesimal Step Stage Evaluation Diagram for MIS ERK(1, 1) Through MIS ERK(1, 4).	58
Figure 2.4.1	Error Plots and Total RHS Evaluations for Non-Linear SWE Using 1^{st} Order MIS ARK & ERK Methods ($g = h_B = 1$).	60
Figure 2.4.2	Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1^{st} Order MIS ARK & ERK Methods ($g = h_B = 1$).	62
Figure 2.4.3	Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1^{st} Order MIS ARK & ERK Methods ($g = 10, h_B = 1$).	62

Figure 2.4.4	Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1 st Order MIS ERK Methods ($g = 10, h_B = 1$).	63
Figure 2.4.5	Error Plots and Slow RHS Evaluations for Non-Linear SWE Using 2 nd Order MIS ARK & ERK Methods ($g = h_B = 1$).	64
Figure 2.4.6	Error Plots and Slow RHS Evaluations for Non-Linear SWE Using 2 nd Order MIS ARK & ERK Methods ($g = 10, h_B = 1$).	65
Figure 2.4.7	Error Plots and Slow RHS Evaluations for Non-Linear SWE using 2 nd Order MIS ERK Methods ($g = 10, h_B = 10$).	66
Figure 3.2.1	Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1 st Order MIS ERK(1, J , p) Method w/Extrapolation ($J = 1, 2; g = 10, h_B = 1$).	71
Figure 3.3.1	Example Diagram for the 3 rd Order MIS ERK(1,1,2,3) Method using Harmonic Sequence.	80
Figure 3.3.2	Example Diagram for the 4 th Order MIS ERK(1,2,2,4) Method using Harmonic Sequence.	81
Figure 3.5.1	Single Core Implementation for the 4 th Order MIS ERK(1,2,2,4) Method Using Harmonic Sequence.	86
Figure 3.5.2	Two Core Parallel Implementation for the 4 th Order MIS ERK(1,2,2,4) Method Using Harmonic Sequence.	87
Figure 4.2.1	Convergence Rates for the NUMA 2D Shallow Water Model for RHS Splitting Option 1.	100
Figure 4.2.2	LSRK4-5 Versus MIS ERK(1,1,50,4) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 0$ Seconds Using Splitting Option One.	101
Figure 4.2.3	LSRK4-5 Versus MIS ERK(1,1,50,4) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 1.5$ Seconds Using Splitting Option One.	101
Figure 4.2.4	LSRK4-5 Versus MIS ERK(1,1,50,4) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 3$ Seconds Using Splitting Option One.	102

Figure 4.2.5	Convergence Rates for the NUMA 2D Shallow Water Model for RHS Splitting Option Two.	105
Figure 4.2.6	LSRK4-5 Versus MIS ERK(1,MIS ERK(1,1,2,4),2,1) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 0$ Seconds Using Splitting Option Two.	106
Figure 4.2.7	LSRK4-5 Versus MIS ERK(1,MIS ERK(1,1,2,4),2,1) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 2$ Seconds Using Splitting Option Two.	107
Figure 4.2.8	LSRK4-5 Versus MIS ERK(1,MIS ERK(1,1,2,4),2,1) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 4$ Seconds Using Splitting Option Two.	107
Figure 4.4.1	Convergence Rates for the NUMA 2D Euler (Rising Thermal Bubble) Model for RHS Splitting Option One.	114
Figure 4.4.2	Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 2D Euler Model at $t = 0$ Seconds Using Splitting Option One.	115
Figure 4.4.3	Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 2D Euler Model at $t = 350$ Seconds Using Splitting Option One.	115
Figure 4.4.4	Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 2D Euler Model at $t = 700$ Seconds Using Splitting Option One.	116
Figure 4.5.1	Convergence Rates for the NUMA 3D Euler (Rising Thermal Bubble) Model for RHS Splitting Option One.	118
Figure 4.5.2	Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 3D Euler Model at $t = 0$ Seconds Using Splitting Option One.	118
Figure 4.5.3	Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 3D Euler Model at $t = 400$ Seconds Using Splitting Option One.	119
Figure 4.5.4	Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 3D Euler Model at $t = 500$ Seconds Using Splitting Option One.	119

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 1.3.1	Tableau of $T_{j,k}$ Solutions with Corresponding Base Method Orders	20
Table 1.3.2	$f(Y_{j,k})$ Functional Evaluation Comparison for Richardson vs. Gragg Extrapolation	29
Table 1.3.3	RHS Evaluation Comparison for Gragg Extrapolation	36
Table 2.2.1	Percent Increase/Decrease of Total Number of RHS Evaluations for SR ERK vs. SR ARK Methods up to Order 3.	54
Table 2.4.1	Percent Increase/Decrease of Slow RHS Evaluations for MIS ERK Methods of Order 1 vs. SR ERK (Forward Euler) Method.	64
Table 2.4.2	Percent Increase/Decrease of Slow RHS Evaluations for MIS ERK Methods of Order 2 vs. SR RK2 and ARK-2 (3 stage) Methods.	66
Table 3.2.1	Comparison of SR ERK4 Versus MIS ERK(1, J , 4) Methods for $1 \leq J \leq 4$ and $gh_B = 10, 100$ Using Algorithm 3.1.1.	74
Table 3.2.2	Comparison of SR ERK4 Versus MIS ERK(1, J , 4) Methods for $1 \leq J \leq 4$ and $gh_B = 10, 100$ Using Algorithm 3.2.1.	76
Table 3.2.3	Comparison of SR ERK4 Versus MIS ERK(2, J , 4) Methods for $1 \leq J \leq 4$ and $gh_B = 10$ Using Algorithm 3.2.1.	77
Table 3.4.1	Comparison of SR ERK4 Versus MIS ERK(1, 1, m , 4) Methods for $1 \leq m \leq 6$ Using Algorithm 3.3.1.	82
Table 3.4.2	Comparison of SR ERK5 Versus MIS ERK(1, 1, m , 5) Methods for $1 \leq m \leq 6$ Using Algorithm 3.3.1.	83
Table 3.5.2	MIS ERK(1, 2, 2, 4) Method: Parallel Comparison for Total Number of RHS Evaluations Per Time-Step	86
Table 3.5.3	Minimum Number of Stage Evaluations and Order Conditions Required for SR ERK Methods up to Order 8.	89

Table 4.2.1	4 th Order Comparison Using the MIS ERK(1, 1, m , 4) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 1$ and $\nu = 0.001$ Using Splitting Option One.	97
Table 4.2.2	4 th Order Comparison Using the MIS ERK(1, 1, m , 4) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 50$ and $\nu = 0.001$ Using Splitting Option One.	99
Table 4.2.3	4 th Order Comparison Using the MIS ERK(1, 1, 1, m_1 , 2, 4) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 75$ and $\nu = 0.001$ Using Splitting Option Two.	103
Table 4.2.4	Single-rate Versus Multirate Comparison Using the MIS ERK(1, MIS ERK(1, 1, 2, 4), m_1 , 1) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 20$ and $\nu = 0.001$ Using Splitting Option Two. . .	104
Table 4.4.1	Single-rate Versus Multirate Comparison for NUMA's 2D Euler Model for $K = 1$ and $\nu = 1$ Using Linearized Splitting Option One.	113
Table 4.4.2	Single-rate Versus Multirate Comparison for NUMA's 2D Euler Model for $K = 1$ and $\nu = 1$ Using Splitting Option Two.	114
Table 4.5.1	Single-rate Versus Multirate Comparison for NUMA's 3D Euler Model for $K = 1$ and $\nu = 1$ Using Linearized Splitting Option One.	117
Table 4.5.2	Single-rate Versus Multirate Comparison for NUMA's 3D Euler Model for $K = 1$ and $\nu = 1$ Using Linearized Splitting Option Two.	117
Appendix A		123
Table A.0.1	Coefficient Tableau for the 1 st Order Forward Euler Method . . .	123
Table A.0.2	Coefficient Tableau for the 2 nd Order Explicit Midpoint Method .	123
Table A.0.3	Coefficient Tableau for the 3 rd Order (Kutta) RK Method	123
Table A.0.4	Coefficient Tableau for the 4 th Order (Classical) RK Method . . .	124
Table A.0.5	Coefficient Tableau for the 4 th Order (3/8 Rule) RK Method . . .	124

Table A.0.6	Coefficient Tableau for the 5 th Order (Butcher) RK Method	124
Table A.0.7	Coefficient Tableau for the 5 th Order (Dormand-Prince) RK Method	125
Table A.0.8	Coefficient Tableau for the ARK1 (1 st Order, 2-Stage) Method . .	125
Table A.0.9	Coefficient Tableau for the ARK2 (2 nd Order, 3-Stage) Method . .	125
Table A.0.10	Coefficient Tableau for the ARK3 (3 rd Order, 4-Stage) Method . .	126
Table A.0.11	Coefficient Tableau for the ARK4 (4 th Order, 6-Stage) Method . .	126
Table A.0.12	Coefficient Tableau for the ARK5 (5 th Order, 8-Stage) Method . .	126
Table A.0.13	Coefficient Tableau for the LSRK4-5 (4 th Order, 5-Stage) Method	128

Appendix D 135

Table D.0.1	Extrapolated MIS ERK(1, 1, m, p) Results for $g = 10, h_B = 1$. . .	136
Table D.0.2	Extrapolated MIS ERK(1, 2, m, p) Results for $g = 10, h_B = 1$. . .	137
Table D.0.3	Extrapolated MIS ERK(1, 3, m, p) Results for $g = 10, h_B = 1$. . .	138
Table D.0.4	Extrapolated MIS ERK(1, 1, m, p) Results for $g = 10, h_B = 10$. . .	139
Table D.0.5	Extrapolated MIS ERK(1, 2, m, p) Results for $g = 10, h_B = 10$. . .	140

THIS PAGE INTENTIONALLY LEFT BLANK

List of Algorithms

1.3.1	Explicit-Euler (Richardson) Extrapolation	23
1.3.2	Explicit-Midpoint (Gragg) Extrapolation	27
2.3.1	2-Partition General Explicit MIS Method	57
2.3.2	2-Partition General Explicit MIS Method (Non-Autonomous) . . .	57
3.1.1	2-Partition General Explicit MIS Method with Extrapolation	70
3.2.1	2-Partition General Explicit MIS Method w/ Extrapolation (Ver. 2)	75
3.3.1	2-Partition General Explicit Extrapolated MIS Method w/ Substepping	79
3.4.1	2-Partition General Explicit MIS Method with Substepping	84

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AMR	adaptive mesh refinement
ARK	additive Runge-Kutta
BC	boundary conditions
CG	continuous Galerkin
CFL	Courant–Friedrichs–Lewy
DG	discontinuous Galerkin
EBG	element-based Galerkin
ERK	explicit Runge-Kutta
ESM	earth system model
FDM	finite-difference method
FEM	finite-element method
FVM	finite-volume method
GBS	Gragg-Bulirsch-Stöer
GE	Gragg-extrapolation
IC	initial conditions
IMEX	implicit-explicit
IBVP	initial boundary value problem
LHS	left hand side
LSRK	low-storage Runge-Kutta

MIS	multirate infinitesimal step
MOL	method of lines
MR	multirate
MREX	m -rate multirate explicit Euler extrapolation
MRP2RK	multirate process-partitioned Runge-Kutta
MURX	multirate extrapolation
NUMA	Nonhydrostatic Unified Model of the Atmosphere
($\mathcal{O}$)	Big- $\mathcal{O}$ notation
ODEs	ordinary differential equations
PDEs	partial differential equations
RHS	right hand side
RK	Runge-Kutta
SSP	strong stability preserving
SWE	shallow water equations
SR	single-rate
TVD	total variation diminishing
WS	Wicker-Skamarock
1D	one-dimensional
2D	two-dimensional
3D	three-dimensional

Executive Summary

Our research is focused on developing a multirate (MR) time-integration approach for accurately resolving the various physical multi-scale processes which exist in systems of hyperbolic partial differential equations (PDEs). Throughout this dissertation we primarily use the single-layered shallow water equations (SWE) as our hyperbolic PDE system of choice since it is widely used to model ocean waves in shallow depth and contains both fast gravity and slow advective waves. In order to approximate the solutions to these systems using numerical methods, choices must be made for how to discretize both space and time.

Our specific MR time-integration technique is based on Richardson extrapolation and is tested using high-order accurate continuous and discontinuous Galerkin methods with an upwind-biased Rusanov flux. The benefit in developing an extrapolated MR time-integration method is to take advantage of solving the fast and slow processes using different time-steps, in addition to achieving high order accuracy in both space and time. In the following chapters we describe the development of our extrapolated MR scheme using explicit Runge-Kutta (ERK) methods. In Chapters 1-3, results are shown for the one-dimensional (1D), non-linear SWE in conservation form. In Chapter 4 we use our numerical method to approximate a more robust two-dimensional (2D) shallow water case, and both a two and three-dimensional (3D) Euler equations case within the Nonhydrostatic Unified Model of the Atmosphere (NUMA), which is used for simulating nonhydrostatic atmospheric processes. Throughout the entire dissertation we further analyze and compare our extrapolated MR method against other single-rate (SR) and MR time-splitting schemes specifically designed for solving problems with multiple time scales.

Chapter 1 begins with a brief introduction to the hyperbolic PDEs we are interested in approximating. In Sections 1.1-1.2 we demonstrate how to discretize the 1D SWE initial boundary value problem (IBVP) using element-based discontinuous Galerkin (DG) methods, where we focus on splitting the right hand side (RHS) of the system into its respective processes. Section 1.3 introduces the theory for time-integration schemes based on Richardson and Gragg extrapolation methods, which use local error estimation to remove lower order error terms in order to generate higher-order methods. We further compare the two extrapolation schemes using our discretized SWE IBVP from Section 1.2 and discuss the

pros and cons for each. Section 1.4 concludes the chapter with a literature review on how extrapolation methods have been used in conjunction with MR time-integration schemes and how our extrapolated multirate method differs from these previous works.

In Chapter 2 we provide a general overview of SR and MR ERK and additive Runge-Kutta (ARK) time-integration schemes. Section 2.2 compares the SR performance (computational efficiency) for schemes developed in Chapter 1, where we demonstrate that SR ERK methods are more cost efficient than SR ARK methods. Sections 2.3-2.4 extend the use of ERK methods within the multirate infinitesimal step (MIS) method, where we further demonstrate how the overall MIS ERK method is computationally more efficient than SR methods of equivalent order of accuracy in terms of the number of slow and fast RHS evaluations. Results and analysis from Chapter 2 are then applied to the development and construction of our extrapolated MR scheme in Chapter 3.

Chapter 3 describes our approach to the development of a generalized extrapolated multirate time-integration scheme, which combines the extrapolation process from Chapter 1 with the MIS ERK method from Chapter 2. In Section 3.1 we derive the generalized algorithm for our extrapolated MIS ERK method with numerical results shown in Section 3.2. For consistency in our comparison between SR and MR methods we developed an efficiency metric based on the slow to fast computational ratio of the multirate scheme compared to a single-rate scheme of equivalent order of accuracy. We further improve upon our method in Section 3.3 through the use of substepping the fast approximations within each slow approximation, where our extrapolated MIS ERK with substepping scheme results are shown in Section 3.4. We end the chapter with a brief discussion and example on a way to introduce a parallel implementation of the extrapolation process, which can further improve the overall computational efficiency of the method.

In Chapter 4 we use our extrapolated multirate time-integration scheme developed in Chapter 3 to solve both a 2D non-linear SWE model with artificial viscosity and both a 2D and 3D compressible Euler equations model. Both models are currently used within NUMA, which is used for simulating atmospheric processes (i.e weather prediction). Various RHS splitting options are considered within both models and a unified continuous/discontinuous Galerkin (CG/DG) method for the spatial discretization. Section 4.1 describes the 2D SWE governing equations using two different splitting options for either a two or three process splitting,

with numerical results shown in Section 4.2. Next we show the governing equations for the nonhydrostatic Euler equations model in Section 4.3. Similar to the 2D SWE model, we also considered two different RHS splitting options with results displayed in Section 4.4 and 4.5 for the 2D and 3D models, respectively. Finally, we provide our conclusion and recommendations for future work in Chapter 5.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

This dissertation is dedicated to my wife, Starrissa, and my daughter, Sierra, who have been my rock to lean on during both my time working on my PhD and throughout my military career. None of this would have been possible if not for their support, patience and love. I also owe a lot of thanks to both my parents for their love, support and prayers. Most importantly, I thank God for giving me this opportunity and helping me to persevere through all of the many challenges I faced along the way.

I want to acknowledge both of my dissertation advisors, Dr. Giraldo and Dr. Kozdon, for their encouragement, guidance and friendship. I owe a lot to both of you for pushing me when I needed to be pushed, and helping me to believe in myself when I had moments of doubt. Thank you for your honest and constructive feedback over the past three years. There truly are not enough words to say thank you for helping me accomplish this dissertation.

I also want to thank all of the staff and faculty I have had the pleasure of working with at the Naval Postgraduate School. Specifically within the Department of Applied Mathematics, I want to thank all of the professors who served on both my exam and dissertation committees: Francis Giraldo, Jeremy Kozdon, Carlos Borges, Wei Kang, Hong Zhou, and Christopher Frenzen, as well as Mara Orescanin within the Department of Oceanography. Each of these members provided valuable assistance and feedback during preparation for my exams and throughout my dissertation research.

I also want to recognize all of the postdoctoral research assistants: Maciej Waruszewski, Felipe Alves and Sohail Reddy, for their friendship and valuable discussions about my research during our weekly lunches. Felipe's assistance was specifically critical in implementing our methods into the NUMA framework.

Lastly, to Jeff Parker, my PhD peer within the Operations Research Department, your friendship over the past three years has meant everything. It was great having a battle buddy within another department that could relate with me about life as a PhD student. You helped make my time at NPS more enjoyable and I wish you all the best.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:

Introduction and Background

Although much work has been done in the areas of oceanic and atmospheric modeling, these areas will remain a source of continued research as we push the boundaries and forge newer and more innovative techniques for solving large-scale, real-world problems on computers. Specifically, the development of better numerical methods is required, as these physical problems continue to present both complex and unique challenges for how to accurately and efficiently resolve the physical processes that exist in nature. We represent these physical processes using mathematical models, which usually involve solving systems of nonlinear partial differential equations (PDEs). These equations help describe the interactions between the various model parameters, such as temperature, pressure, momentum, mass, and energy. However, the complexity within these models typically results in problems where no analytical solution can be found; therefore, the solution must be solved using a numerical method. Before attempting to solve the more complicated models, we typically begin by solving a simplified version of the problem, where a known analytical solution exists. The simpler problem is beneficial as it provides insight into the numerical model itself, such as the accuracy of the numerical methods through verification using a known analytic solution.

When using numerical methods to solve PDEs, certain choices must be made for how to perform both the spatial and temporal discretizations, such that the overall numerical method we choose is consistent with the underlying continuous problem we are modeling. Within this work, we are interested in solving initial boundary value problems of the form:

$$\begin{aligned}\frac{\partial \mathbf{q}(x, t)}{\partial t} &= \mathbf{P}_1(t, \mathbf{q}) + \mathbf{P}_2(t, \mathbf{q}) + \cdots + \mathbf{P}_N(t, \mathbf{q}) \text{ for } t \geq t_0, \\ \mathbf{q}(x, t_0) &= \mathbf{q}_0, \text{ and } x_L \leq x \leq x_R,\end{aligned}\tag{1.0.1}$$

where the right hand side (RHS) of the PDE has been split into processes P_i for $i = 1, \dots, N$. The left and right spatial boundaries are defined by x_L and x_R , where boundary conditions

must be applied. For simplicity, we assume that $x \in \mathbb{R}^1$ and that there are only two processes:

$$\frac{\partial \mathbf{q}(x, t)}{\partial t} = \mathbf{F}(t, \mathbf{q}) + \mathbf{S}(t, \mathbf{q}), \quad (1.0.2)$$

where $\mathbf{F}$ and $\mathbf{S}$ represent a fast and slow process, respectively.

In order to discretize the PDE, we use high order, discontinuous element-based Galerkin (EBG) methods to represent the spatial derivatives, in conjunction with an explicit multirate (MR) extrapolation time-integration scheme to evaluate the time-dependent derivatives of the multi-process PDE from Equation (1.0.2). Note, the spatial approximations define the discrete fast and slow processes, $\mathbf{F}$ and $\mathbf{S}$.

In this chapter, we define the continuous PDE system for our test problem in Section 1.1, using the one-dimensional (1D) nonlinear shallow water equations (SWE) in conservation form. Section 1.2 describes the spatial discretization method used. Section 1.3 provides an overview on the theory of extrapolated time-integration methods and Section 1.4 ends the chapter with a brief review of the prior development of MR extrapolation schemes. In the next chapter we provide a general overview of both single-rate (SR) and MR explicit Runge-Kutta (RK) multi-stage time-integration schemes, which provide the foundation for the extrapolated MR time-integration method we develop in Chapter 3.

1.1 Continuous PDE System

Models of the ocean are constructed using the fundamental conservation laws. The simplest of models might only contain a few physical components; however, more complicated models can contain a large number of physical processes [1]. Throughout this dissertation, our research will compare our MR extrapolation scheme with other MR time-splitting methods used for resolving these physical processes. Specifically, we will consider a simplified SWE model described in [2].

As a first step, consider the SWE model in conservation form as:

$$\frac{\partial h}{\partial t} + \nabla \cdot \mathbf{U} = 0 \quad (1.1.1a)$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \left(\frac{\mathbf{U} \otimes \mathbf{U}}{h} \right) + h \nabla (gh_s) = \mathbf{0} , \quad (1.1.1b)$$

where the momentum is defined as $\mathbf{U} = h\mathbf{u}$, $\otimes$ represents the tensor product operator, the nabla operator, ∇ , represents a vector comprised of partial derivatives $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y})^\top$, where $\top$ is the transpose operator, h is the total height of the water (or viscous incompressible fluid) column, and g is the gravity constant. Additionally, we make the following simplifying assumptions:

- The solution wavelengths of interest are significantly larger than the depth, so that the velocity, $\mathbf{u} = (u, v)^\top$, can reasonably be assumed to be constant in the entire column.
- The system is non-rotational and we can disregard Coriolis effects.
- We assume no wind stress and a frictionless bottom surface.
- Total height of the water column is $h = h_s + h_B$ where h_s is the surface wave height measured from mean sea level and h_B is the height from the bottom surface to mean sea level as depicted in Figure 1.1.1.

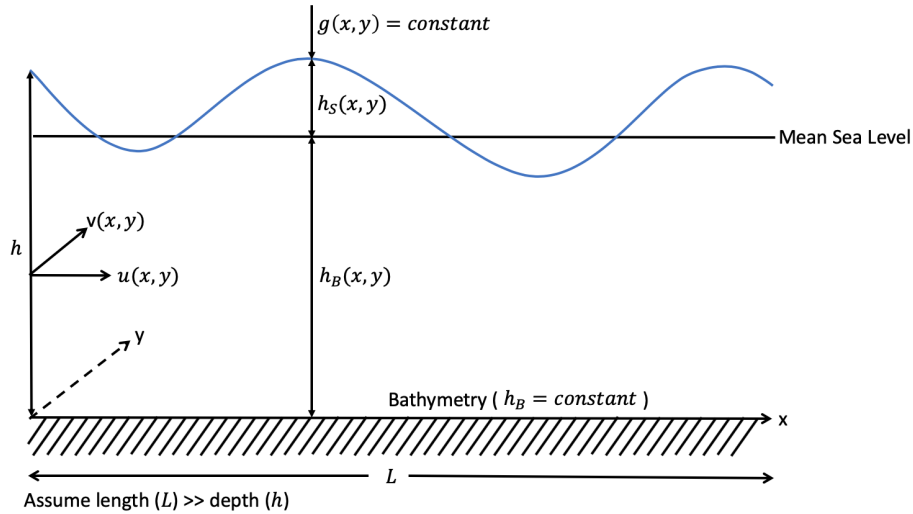


Figure 1.1.1. Two Dimensional Shallow Water Diagram.

1.1.1 Separating the Continuous Equations

Let us now separate the SWE PDE system into its fast (gravity) and slow (advection) processes. Notice that the third term in Equation (1.1.1b) can be written as

$$h\nabla(gh_s) = \nabla(ghh_s) - gh_s\nabla(h), \quad (1.1.2)$$

where since we have defined $h = h_s + h_B$, then $hh_s = (h_s + h_B)h_s = h_s^2 + h_Bh_s$. The right hand side of Equation (1.1.2) becomes:

$$h\nabla(gh_s) = \nabla(gh_s^2 + gh_Bh_s) - gh_s\nabla(h_s + h_B). \quad (1.1.3)$$

Assuming that h_B and g are constant, then the terms in (1.1.3) simplify to

$$2gh_s\nabla h_s + gh_B\nabla h_s - gh_s\nabla h_s = \nabla\left(\frac{1}{2}gh_s^2\right) + gh_B\nabla h_s. \quad (1.1.4)$$

Substituting the RHS of Equation (1.1.4) into Equation (1.1.1), allows Equation (1.1.1b) to be written as:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \left[\frac{\mathbf{U} \otimes \mathbf{U}}{h} + \frac{1}{2}gh_s^2\mathbf{I}_2 + gh_Bh_s\mathbf{I}_2 \right] = \mathbf{0}, \quad (1.1.5)$$

where $\mathbf{I}_2$ is the rank-2 identity matrix, and we can now rewrite Equations (1.1.1a) - (1.1.1b) as follows:

$$\frac{\partial h_s}{\partial t} + \nabla \cdot \mathbf{U} = 0 \quad (1.1.6a)$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \left[\frac{\mathbf{U} \otimes \mathbf{U}}{h} + \frac{1}{2}gh_s^2\mathbf{I}_2 + gh_Bh_s\mathbf{I}_2 \right] = \mathbf{0}, \quad (1.1.6b)$$

where $\frac{\partial h}{\partial t} = \frac{\partial h_s}{\partial t}$ in Equation (1.1.6a), since h_B is assumed to be constant.

Now if we let $\mathbf{q} = (h_s, \mathbf{U})^T$, then we can rewrite Equations (1.1.6a-b) in compact form as the following PDE:

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{S}(\mathbf{q}) + \mathbf{F}(\mathbf{q}), \quad (1.1.7)$$

where the slow term is:

$$\mathbf{S}(\mathbf{q}) = - \begin{pmatrix} \nabla \cdot \mathbf{U} \\ \nabla \cdot \left(\frac{\mathbf{U} \otimes \mathbf{U}}{h} \right) \end{pmatrix}, \quad (1.1.8)$$

and the fast term is:

$$\mathbf{F}(\mathbf{q}) = - \begin{pmatrix} 0 \\ \nabla \cdot (gh_B h_s \mathbf{I}_2 + \frac{1}{2}gh_s^2 \mathbf{I}_2) \end{pmatrix}. \quad (1.1.9)$$

Notice that the fast gravity waves are only contained within the function $\mathbf{F}(\mathbf{q})$, while the slower advection waves are only within the function $\mathbf{S}(\mathbf{q})$; therefore, the fast and slow waves have been separated. Furthermore, if we define $\mathbf{R}(\mathbf{q}) = \mathbf{S}(\mathbf{q}) + \mathbf{F}(\mathbf{q})$, then the PDE system can be written as follows:

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{R}(\mathbf{q}), \quad (1.1.10)$$

where the right hand side vector, $\mathbf{R}(\mathbf{q})$, contains both processes.

Our aim is to define an extrapolated MR time-integration scheme that can take advantage of solving the fast and slow solutions within the system using different time scales, Δt_f and Δt_s , which represent the fast and slow time steps, respectively. The advance of the overall solution in time is then considered a MR solution. In order for the numerical method to be useful, we must ensure that the MR time-integration method is numerically stable and consistent. Additionally, it is often required that certain physical properties of the solution are respected, such as conservation of mass and momentum. Finally, we seek a method that is computationally efficient, and thus useful for practical simulations.

1.1.2 One-Dimensional (1D) Shallow Water Equations

In 1D, the governing Equations (1.1.7) - (1.1.9) reduce to:

$$\frac{\partial}{\partial t} \begin{pmatrix} h_s \\ U \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} U \\ \frac{U^2}{h} \end{pmatrix} = - \frac{\partial}{\partial x} \begin{pmatrix} 0 \\ gh_B h_s + \frac{1}{2}gh_s^2 \end{pmatrix}. \quad (1.1.11)$$

Note that the PDE system in Equation (1.1.11) is hyperbolic and represents an initial boundary value problem (IBVP) for the following initial conditions (IC):

$$\begin{aligned} h_s(x, 0) &= \frac{1}{2} \cos(c\pi x), \\ U(x, 0) &= 0, \end{aligned} \quad (1.1.12)$$

where c in Equation (1.1.12) is a constant and represents the wave frequency. Additionally, we use no flux boundary conditions (BC) at the endpoints of our spatial domain, such that:

$$U(-1, t) = U(1, t) = 0. \quad (1.1.13)$$

Therefore, the 1D IBVP is defined using Equations (1.1.11) - (1.1.13). We solve this IBVP using the explicit extrapolation MR time-integration scheme we develop in Chapter 3.1, with results shown in Chapter 3.2. In the next section we provide a general overview of EBG methods and then use them to discretize the IBVP. Additionally, Appendix B provides the analytic solution to Equations (1.1.11) - (1.1.13).

1.2 Overview of Element-Based Galerkin (EBG) Methods

EBG methods are numerical methods for locally solving PDEs within a finite number of elements, which rely on the integral form of the continuous PDEs. The choice in using EBG methods is that they have the desirable characteristics (assuming the continuous problem is well-posed) of being consistent with the underlying continuous problem, as well as algebraically and geometrically flexible; see [3] for example. The last quality is important as various types of unstructured and/or adaptive spatial meshes could be considered. For example, EBG methods can be built directly on an element of a pre-specified geometry, such that the spatial grids can be any structured or unstructured mesh made up of various geometric shapes from quadrilaterals and triangles in 2D, to tetrahedra, hexahedra or pyramids in 3D [3].

When using EBG methods, we must also make a choice of using either continuous Galerkin (CG) or discontinuous Galerkin (DG) methods for the spatial discretization of our IBVP. Although both choices have their benefits, we will use the DG method as it is locally conservative, the solution space is less restrictive, and the well-known Rusanov flux approach for handling continuity of the solution across each elemental boundary is often suitable for hyperbolic problems [3]. The following sections assume the reader has a general understanding of Galerkin methods, where the DG approach specifically consists of dividing up the overall spatial grid into a specified number of elements. Within each element, a smaller local problem is numerically solved, such that the global solution to the problem is constructed by summing up all of the local elemental solutions. In contrast

to the CG method, DG methods do not require the local solutions within each element to be continuous at each elemental boundary. However, continuity at elemental interfaces is weakly enforced via a numerical flux term. Therefore, each element communicates with its neighboring elements what its respective contribution is to the global solution at the elemental boundary.

1.2.1 Discontinuous Galerkin (DG) Method

Starting from Equation (1.1.11), let us first simplify the problem by placing it into the following form:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{S}(\mathbf{q})}{\partial x} + \frac{\partial \mathbf{F}(\mathbf{q})}{\partial x} = \mathbf{0}, \quad (1.2.1)$$

where $\mathbf{q} = (h_s, U)^\top$, $\mathbf{S}(\mathbf{q}) = (U, \frac{U^2}{h})^\top$, and $\mathbf{F}(\mathbf{q}) = (0, gh_B h_s + \frac{1}{2}gh_s^2)^\top$. Now, multiplying the continuous form of the PDE in Equation (1.2.1) by a basis (test) function, ψ , integrating over each locally defined elemental domain, Ω_e , within the 1D mesh, and replacing the continuous variables with their respective numerical approximation via an N^{th} order polynomial, we can write Equation (1.2.2) as follows:

$$\int_{\Omega_e} \psi_i \left(\frac{\partial \mathbf{q}_N^{(e)}}{\partial t} + \frac{\partial \mathbf{S}(\mathbf{q}_N^{(e)})}{\partial x} + \frac{\partial \mathbf{F}(\mathbf{q}_N^{(e)})}{\partial x} \right) d\Omega_e = \mathbf{0} = \int_{\Omega_e} \psi \epsilon d\Omega_e, \quad (1.2.2)$$

where the numerical solution for $\mathbf{q}(x, t)$ as an N^{th} order polynomial expansion is $\mathbf{q}_N^{(e)}(x, t) = \sum_{j=0}^N \psi_j(x) \mathbf{q}_j^{(e)}(t)$, such that ψ_j represents the local polynomial basis functions and $\mathbf{q}_j^{(e)}$ are the expansion coefficients.

Now, since we have chosen to use the DG method for our spatial discretization, then the vector solution space will be less restrictive than that of the CG method, such that the set of polynomial basis functions, ψ , we choose can be any N^{th} order interpolating polynomial that belongs within the finite-dimensional L^2 Sobolev space, satisfying the following criteria [3]:

1. The solution, $\mathbf{q}_N^{(e)}(x, t)$, and basis functions, $\psi_j(x)$, are square-integrable and continuous within each element; however, continuity is not required at the elemental boundaries.
2. The basis functions must be orthogonal to the numerical error, ϵ , incurred by approx-

imating $\mathbf{q}(x, t)$ with $\mathbf{q}_N^{(e)}(x, t)$ as shown within the last term of Equation (1.2.2).

Continuing with the DG discretization of Equation (1.2.2) yields,

$$\int_{\Omega_e} \psi_i \frac{\partial \mathbf{q}_N^{(e)}}{\partial t} d\Omega_e + \int_{\Omega_e} \psi_i \frac{\partial \mathbf{S}(\mathbf{q}_N^{(e)})}{\partial x} d\Omega_e = - \int_{\Omega_e} \psi_i \frac{\partial \mathbf{F}(\mathbf{q}_N^{(e)})}{\partial x} d\Omega_e. \quad (1.2.3)$$

Using the product rule, we can rewrite both the second and third terms in Equation (1.2.3) using:

$$\psi_i \frac{\partial \mathbf{S}(\mathbf{q}_N^{(e)})}{\partial x} = \frac{\partial}{\partial x} \left(\psi_i \mathbf{S}(\mathbf{q}_N^{(e)}) \right) - \frac{d\psi_i}{dx} \mathbf{S}(\mathbf{q}_N^{(e)}), \quad (1.2.4)$$

$$\psi_i \frac{\partial \mathbf{F}(\mathbf{q}_N^{(e)})}{\partial x} = \frac{\partial}{\partial x} \left(\psi_i \mathbf{F}(\mathbf{q}_N^{(e)}) \right) - \frac{d\psi_i}{dx} \mathbf{F}(\mathbf{q}_N^{(e)}), \quad (1.2.5)$$

so that Equation (1.2.3) becomes the following when (1.2.4) and (1.2.5) are substituted in:

$$\begin{aligned} \int_{\Omega_e} \psi_i \frac{\partial \mathbf{q}_N^{(e)}}{\partial t} d\Omega_e + \int_{\Omega_e} \frac{\partial}{\partial x} \left(\psi_i \mathbf{S}(\mathbf{q}_N^{(e)}) \right) d\Omega_e - \int_{\Omega_e} \frac{d\psi_i}{dx} \mathbf{S}(\mathbf{q}_N^{(e)}) d\Omega_e = \\ - \int_{\Omega_e} \frac{\partial}{\partial x} \left(\psi_i \mathbf{F}(\mathbf{q}_N^{(e)}) \right) d\Omega_e + \int_{\Omega_e} \frac{d\psi_i}{dx} \mathbf{F}(\mathbf{q}_N^{(e)}) d\Omega_e. \end{aligned} \quad (1.2.6)$$

Using the Fundamental Theorem of Calculus to simplify both the second and fourth integrals provides

$$\begin{aligned} \int_{\Omega_e} \psi_i \frac{\partial \mathbf{q}_N^{(e)}}{\partial t} d\Omega_e + \sum_{k=1}^{N_f} \left[\hat{\mathbf{n}}_{\Gamma}^{(e,k)} \psi_i \mathbf{S}(\mathbf{q}_N^{(e,k)}) \right]_{\Gamma_e} - \int_{\Omega_e} \frac{d\psi_i}{dx} \mathbf{S}(\mathbf{q}_N^{(e)}) d\Omega_e = \\ - \sum_{k=1}^{N_f} \left[\hat{\mathbf{n}}_{\Gamma}^{(e,k)} \psi_i \mathbf{F}(\mathbf{q}_N^{(e,k)}) \right]_{\Gamma_e} + \int_{\Omega_e} \frac{d\psi_i}{dx} \mathbf{F}(\mathbf{q}_N^{(e)}) d\Omega_e, \end{aligned} \quad (1.2.7)$$

where $\mathbf{S}(\mathbf{q}_N^{(e,k)})$ and $\mathbf{F}(\mathbf{q}_N^{(e,k)})$ both represent the numerical fluxes for the slow and fast terms respectively. The summation $\sum_{k=1}^{N_f}$ adds the respective numerical flux contributions from an elements neighboring elements, where N_f represents the total number of neighbors (faces/boundaries), and $\hat{\mathbf{n}}_{\Gamma}^{(e,k)}$ is the normal vector at an elemental boundary in the direction of its k^{th} neighbor [3]. For 1D problems we have that $N_f = 2$.

In order to implement DG, we need to expand $\mathbf{q}_N^{(e)}$ and its derivatives in terms of an N^{th} order polynomial basis function. In this work we chose to use a nodal basis where $\psi_j(x)$ is

an interpolating Lagrange polynomial, such that:

$$\mathbf{q}_N^{(e)}(x, t) = \sum_{j=0}^N \psi_j(x) \mathbf{q}_j^{(e)}(t), \quad (1.2.8a)$$

$$\frac{\partial \mathbf{q}_N^{(e)}(x, t)}{\partial t} = \sum_{j=0}^N \psi_j(x) \frac{d\mathbf{q}_j^{(e)}(t)}{dt}, \quad (1.2.8b)$$

where

$$\mathbf{q}_j^{(e)} = \begin{pmatrix} h_{s,j}^{(e)} \\ U_j^{(e)} \end{pmatrix}, \quad \text{and} \quad \frac{d\mathbf{q}_j^{(e)}}{dt} = \begin{pmatrix} \frac{dh_{s,j}^{(e)}}{dt} \\ \frac{dU_j^{(e)}}{dt} \end{pmatrix}. \quad (1.2.9)$$

Therefore, by substituting (1.2.8a-b) into (1.2.7), it now simplifies to the elemental matrix form:

$$M_{ij}^{(e)} \frac{d\mathbf{q}_j^{(e)}}{dt} + F_{ij}^{(e)} \mathbf{S}(\mathbf{q}_j^{(e,k)}) - \tilde{D}_{ij}^{(e)} \mathbf{S}(\mathbf{q}_j^{(e)}) = -F_{ij}^{(e)} \mathbf{F}(\mathbf{q}_j^{(e,k)}) + \tilde{D}_{ij}^{(e)} \mathbf{F}(\mathbf{q}_j^{(e)}). \quad (1.2.10)$$

All terms within Equation (1.2.10) must now be defined. Section 1.2.2 describes the elemental matrices along with their respective basis functions, while Section 1.2.4 defines the respective slow and fast terms, along with their respective fluxes.

1.2.2 1D Elemental Matrices and Basis Functions

The elemental matrices $M_{ij}^{(e)}$, $\tilde{D}_{ij}^{(e)}$, and $F_{ij}^{(e)}$ are defined as

$$M_{ij}^{(e)} = \int_{x_0}^{x_N} \psi_i(x) \psi_j(x) dx = \frac{\Delta x^{(e)}}{2} \int_{-1}^{+1} \psi_i(\xi) \psi_j(\xi) d\xi, \quad (1.2.11a)$$

$$\tilde{D}_{ij}^{(e)} = \int_{x_0}^{x_N} \frac{d\psi_i(x)}{dx} \psi_j(x) dx = \int_{-1}^{+1} \frac{d\psi_i(\xi)}{d\xi} \psi_j(\xi) d\xi, \quad (1.2.11b)$$

$$F_{ij}^{(e)} = [\psi_i(x) \psi_j(x)]_{x_0}^{x_N} = [\psi_i(\xi) \psi_j(\xi)]_{-1}^{+1}, \quad (1.2.11c)$$

where $x \rightarrow \xi$, such that $\int_{x_0}^{x_N} dx = \int_{-1}^{+1} \frac{dx}{d\xi} d\xi$, and $\frac{dx}{d\xi} = \frac{\Delta x^{(e)}}{2}$. Further details for defining ξ are provided in Section 1.2.3. Furthermore, if we choose to build our 1D basis functions

ψ_i , and their derivatives $\frac{d\psi_i}{d\xi}$ using Lagrange polynomials, then:

$$\psi_i(\xi) = \prod_{\substack{j=0 \\ j \neq i}}^N \frac{(\xi - \xi_j)}{(\xi_i - \xi_j)}, \quad (1.2.12)$$

and

$$\frac{d\psi_i(\xi)}{d\xi} = \sum_{\substack{k=0 \\ k \neq i}}^N \left(\frac{1}{\xi_i - \xi_k} \right) \prod_{\substack{j=0 \\ j \neq i \\ j \neq k}}^N \frac{(\xi - \xi_j)}{(\xi_i - \xi_j)}. \quad (1.2.13)$$

1.2.3 Mapping from Physical Space to Computational Space

When discretizing in space, we must choose not only the order (N) of the interpolating polynomial to represent our solution, but also the total number of elements, N_e . The computational efficiency of the DG method will vary based on these choices, as the number of time-steps will increase linearly with the number of elements, but (approximately) quadratically with the polynomial order. In 1D, the total number of DG points used to approximate our solution is $\text{DG}_{\text{points}} = N_e(N + 1)$.

The reference element in computational space is usually defined as $\xi \in [-1, +1]$; therefore, the 1D mapping for an element with points $[x_0, x_N] \in \mathbb{R}$ in physical space to the reference element with points $[\xi_0, \xi_N] \in \mathbb{R}$ in computational space can be found as follows:

$$\xi_i = \frac{2(x_i - x_0)}{\Delta x^{(e)}} - 1, \quad i = 0 : N, \quad (1.2.14)$$

where $\Delta x^{(e)} = \frac{(x_{\max} - x_{\min})}{N_e}$. Furthermore, the reverse mapping in 1D from the reference element points $[-1, +1]$ to any element's physical space points $[x_0, x_N]$ can be found using:

$$x_0 = x_{\min} + \Delta x^{(e)}(e - 1), \quad e = 1 : N_e, \quad (1.2.15a)$$

$$x_i = x(\xi_i) = x_0 + \frac{\Delta x^{(e)}}{2}(\xi_i + 1), \quad i = 0 : N. \quad (1.2.15b)$$

Additionally, when using Galerkin methods to perform the spatial discretization of our

problem, we have flexibility on choosing the specific points $\xi_i \in [-1, +1]$ for $i = 0, \dots, N$, to use when evaluating the Lagrange polynomials described in Section 1.2.2. The conditioning of the matrices arising in DG depend strongly on the chosen quadrature points. Though there are a variety of well-known, high quality quadrature points, we have chosen to use the roots of Legendra-Gauss-Lobatto (LGL) polynomials (i.e., Lobatto points), since the basis functions constructed within each element are well-behaved at these points. LGL polynomials also give points at the boundary of each element; therefore, they provide an excellent approximation to our functions and their derivatives as the polynomial order (N) increases.

Finally, when using either CG or DG methods, we have options for how to perform the numerical integration (quadrature) when constructing the elemental matrices in Equations (1.2.11a) - (1.2.11c). As previously mentioned, the order N of the interpolating polynomial and choice of points to use for quadrature must be selected. For our specific choice of Lobatto points, we can accurately evaluate integrals up to $2N - 1$ degree polynomials.

Since we are using affine meshes in 1D, the mass matrix arises from the product of two degree N polynomials. Furthermore, since we use Lobatto quadrature, the mass matrix though diagonal, will be inexact as we can only integrate polynomials of degree $2N - 1$. If we wanted an exact mass matrix there are two options. We could use Legendre-Gauss points which lead to exact quadrature of degree $2N + 1$ polynomials, but then we would need to interpolate to evaluate the numerical fluxes as there are no interpolation points at the element boundaries. The other option is to continue to use Lobatto points but interpolate to a different quadrature grid for integration; however, this would result in a non-diagonal mass matrix. Though neither of these options are too problematic in 1D, in multiple dimensions both of these options introduce additional computational overheads.

For the terms on the RHS of our IBVP, often referred to as the stiffness matrices, the situation is similar. For example, Equation (1.1.11) includes the following integral that must be evaluated:

$$-\frac{\partial}{\partial x} \left(\frac{1}{2} g h_s^2 \right) \approx - \int_{\Omega_e} \psi_i \frac{\partial}{\partial x} \left(\frac{1}{2} g h_{s_N}^{(e)2} \right) d\Omega_e, \quad (1.2.16)$$

which involves a degree $3N - 1$ polynomial. Though we could interpolate to a high-

order quadrature grid to evaluate this integral exactly, we make the common choice using underintegration and aliasing all terms at the Lobatto points. Using a high-order quadrature would increase the spatial stability of the scheme, but since our focus is time-integration, this is not done here.

1.2.4 Slow and Fast Terms

From Equation 1.2.10 the slow and fast terms are defined as

$$\mathbf{S}(\mathbf{q}_j^{(e)}) = \begin{pmatrix} U_j^{(e)} \\ \frac{(U_j^{(e)})^2}{h_{s,j}^{(e)} + h_B} \end{pmatrix}, \quad (1.2.17a)$$

$$\mathbf{F}(\mathbf{q}_j^{(e)}) = \begin{pmatrix} 0 \\ gh_B h_{s,j}^{(e)} + \frac{1}{2}g(h_{s,j}^{(e)})^2 \end{pmatrix}, \quad (1.2.17b)$$

and the flux terms use the Rusanov flux

$$\mathbf{S}(\mathbf{q}_j^{(e,k)}) = \frac{1}{2} \left[\mathbf{S}^{(e)} + \mathbf{S}^{(k)} - |\lambda|(\mathbf{q}_j^{(k)} - \mathbf{q}_j^{(e)}) \right], \quad (1.2.18a)$$

$$\mathbf{F}(\mathbf{q}_j^{(e,k)}) = \frac{1}{2} \left[\mathbf{F}^{(e)} + \mathbf{F}^{(k)} - |\lambda|(\mathbf{q}_j^{(k)} - \mathbf{q}_j^{(e)}) \right], \quad (1.2.18b)$$

where $\lambda^{(k)} = \max(|U^{(e)}| + \sqrt{gh^{(e)}}, |U^{(k)}| + \sqrt{gh^{(k)}})$ to average the solutions between elements; see for example [4]. Finally, we complete the DG discretization of Equation (1.2.10) by moving the second and third terms of (1.2.10) to the RHS and then multiply all terms by the inverse mass matrix $(M_{ij}^{(e)})^{-1}$, such that

$$\mathbf{I}_N = (M_{ij}^{(e)})^{-1} M_{ij}^{(e)}, \quad (1.2.19a)$$

$$\hat{D}_{ij}^{(e)} = (M_{ij}^{(e)})^{-1} \tilde{D}_{ij}^{(e)}, \quad (1.2.19b)$$

$$\hat{F}_{ij}^{(e)} = (M_{ij}^{(e)})^{-1} F_{ij}^{(e)}, \quad (1.2.19c)$$

where $\mathbf{I}_N$ is the rank- N identity matrix.

Therefore, Equation (1.2.10) can be written as

$$\frac{d\mathbf{q}_i^{(e)}}{dt} = -\hat{F}_{ij}^{(e)} \left[\mathbf{S}(\mathbf{q}_j^{(e,k)}) + \mathbf{F}(\mathbf{q}_j^{(e,k)}) \right] + \hat{D}_{ij}^{(e)} \left[\mathbf{S}(\mathbf{q}_j^{(e)}) + \mathbf{F}(\mathbf{q}_j^{(e)}) \right], \quad (1.2.20)$$

or for our purposes, it is better to keep the slow and fast terms separated such that Equation (1.2.20) is written as the following semidiscretization:

$$\frac{d\mathbf{q}_i^{(e)}}{dt} = \mathbf{R}_S(\mathbf{q}_j^{(e)}) + \mathbf{R}_F(\mathbf{q}_j^{(e)}), \quad (1.2.21)$$

where $\mathbf{R}_S$ and $\mathbf{R}_F$ are the slow and fast RHS terms respectively, and defined as

$$\mathbf{R}_S(\mathbf{q}_j^{(e)}) = -\hat{F}_{ij}^{(e)} \mathbf{S}(\mathbf{q}_j^{(e,k)}) + \hat{D}_{ij}^{(e)} \mathbf{S}(\mathbf{q}_j^{(e)}), \quad (1.2.22a)$$

$$\mathbf{R}_F(\mathbf{q}_j^{(e)}) = -\hat{F}_{ij}^{(e)} \mathbf{F}(\mathbf{q}_j^{(e,k)}) + \hat{D}_{ij}^{(e)} \mathbf{F}(\mathbf{q}_j^{(e)}). \quad (1.2.22b)$$

Note, Equation (1.2.21) is only the discretization in space. The remainder of our dissertation concerns the discretization in time.

1.3 Theory of Extrapolation Methods for Time Integration

In this section we provide a brief discussion on the theory of extrapolation methods using both Richardson's and Gragg's methods for time integration. These methods are examples of how extrapolation can be used to generate higher-order methods using a lower-order base method. Additionally, this section discusses the link between both RK and extrapolation, as an extrapolation scheme can be written as an RK method when using a fixed number of iterations. These ideas are helpful as they set the stage for the construction of our explicit MR extrapolation method in Chapter 3.

1.3.1 Classical (Richardson) Extrapolation

A deep history of extrapolation methods can, in some sense, be traced back to Archimedes in 250 BC, with the numerical approximations for the number π [5]. Although many contributions have been made to extrapolation methods, those presented here are based on the works of Richardson [6], [7] and Romberg [8].

Richardson's work primarily focused on the use of centered divided differences for solving ordinary differential equations (ODEs), where he used extrapolation to remove the first global error term of order $\mathcal{O}(h^2)$. He termed his method " h^2 -extrapolation," as he improved his numerical method's accuracy to $\mathcal{O}(h^4)$ by using a single step of extrapolation. In 1955, Romberg improved upon this method using composite trapezoidal approximations by successively halving the step size of his method [8]. Specifically, he is responsible for deriving the general extrapolation formula by linearly combining the computed approximations using step sizes of h and $h/2$, which successfully eliminated the lower order error terms within the asymptotic expansion of the global error function.

Defining the Local Error:

Let us first start with the following scalar IBVP:

$$q'(t) = f(t, q(t)), \quad q(t_0) = q_0, \quad (1.3.1)$$

and solve for the solution at time t_{n+1} using forward Euler's method with a step size of h , such that $t_{n+1} = t_n + h$ for $n \geq 0$. The explicit forward Euler's method and its local error, d_n , are:

$$q_{n+1} = q_n + hf(t_n, q_n), \quad (1.3.2a)$$

$$d_n = q_{n+1} - q_n - hf(t_n, q_n) = C_n h^2 + \mathcal{O}(h^3), \quad (1.3.2b)$$

where the coefficient C_n is independent of the step size h and is proportional to the first higher order term not kept within the Taylor Series expansion of the numerical method used (i.e., $C_n \sim q^{(p+1)}(t_n)$), and where $q^{(p)}$ represents the p^{th} -derivative of (1.3.1). For Euler's method above, the C_n term is:

$$C_n \sim q''(t_n) = f'(t_n, q_n). \quad (1.3.3)$$

More precisely, if we assume that $q_n = q(t_n)$, then Euler's method in Equation (1.3.2a) can be written as:

$$q_{n+1} = q(t_n) + hf(t_n, q(t_n)), \quad (1.3.4a)$$

$$= q(t_n) + hq'(t_n). \quad (1.3.4b)$$

Next, we take the Taylor series expansion of the exact solution, such that:

$$q(t_{n+1}) = q(t_n) + hq'(t_n) + \mathcal{O}(h^2). \quad (1.3.5)$$

Now, if we define the local error, d_n , to be the error introduced in a single iteration of using Euler's method, then d_n corresponds to the difference between the exact and numerical solutions, such that:

$$d_n = q(t_{n+1}) - q_{n+1}, \quad (1.3.6a)$$

$$= q(t_n) + hq'(t_n) + \mathcal{O}(h^2) - (q(t_n) + hq'(t_n)), \quad (1.3.6b)$$

$$= \mathcal{O}(h^2). \quad (1.3.6c)$$

Therefore, the local error for Euler's method is shown to be $\mathcal{O}(h^2)$. Notice that if we continued with the Taylor series expansion of the exact solution, then the first few terms within $\mathcal{O}(h^2)$ are:

$$\mathcal{O}(h^2) = \frac{h^2}{2}q''(t_n) + \frac{h^3}{3!}q'''(t_n) + \frac{h^4}{4!}q^{(4)}(t_n) + \dots, \quad (1.3.7)$$

where we defined C_n in Equation (1.3.2b) to be proportional to the first term within Equation (1.3.7), such that:

$$C_n \sim q''(t_n). \quad (1.3.8)$$

To be exact, $C_n = \frac{q''(t_n)}{2}$ in the case of forward Euler's method. Therefore, if we combine Equations (1.3.6) and (1.3.7) we will arrive at the expression for d_n we had in Equation (1.3.2b), where C_n is substituted into the first term of Equation (1.3.7) and $\mathcal{O}(h^3)$ is used to represent all of the remaining higher order terms. Notice that Euler's method is first order (i.e., $p = 1$), and has local error on the order of $\mathcal{O}(h^2)$. In general, when solving the IBVP in Equation (1.3.1) with an explicit RK method of order p , the local error will be on the order of $\mathcal{O}(h^{p+1})$. We will therefore define the local error for an explicit p^{th} -order method as follows:

$$d_n = C_n h^{p+1} + \mathcal{O}(h^{p+2}). \quad (1.3.9)$$

We do not show the global error results here, as we are only concerned with the local error estimation; however, the primary difference between global and local errors is that the later

is determined by a single iteration of the numerical method, where the error in step $n + 1$ is evaluated assuming there was no error in step n . In contrast, the global error calculation does not make the assumption that there is no error in step n , and is a measurement of all the accumulated error between successive steps.

Computing Local Error Estimation:

Let the solution of the IBVP at time t_{n+1} using a step size of h be defined as q_{n+1}^h . If we also compute the solution at t_{n+1} using two half-steps of size $\frac{h}{2}$, then the second solution is defined as $q_{n+1}^{h/2}$. Since we are using the same base-method of order p , the second solution's local error at time t_{n+1} using two half-steps is expected to be 2^p times smaller. The local error schemes are defined as:

$$q_{n+1}^h - q(t_{n+1}) = C_n h^{p+1} + \mathcal{O}(h^{p+2}), \quad (1.3.10a)$$

$$q_{n+1}^{h/2} - q(t_{n+1}) = 2C_n (h/2)^{p+1} + \mathcal{O}(h^{p+2}), \quad (1.3.10b)$$

where the additional factor of 2 times C_n in Equation (1.3.10b) arises from the fact that two steps of $\frac{h}{2}$ were used instead of one step of size h . Additionally, $q(t_{n+1})$ represents the exact solution at time $t = t_0 + nh$. Notice, if we subtract Equation (1.3.10a) from Equation (1.3.10b) to remove the $q(t_{n+1})$ term on the LHS, then

$$q_{n+1}^{h/2} - q_{n+1}^h = C_n h^{p+1} (2^{-p} - 1) + \mathcal{O}(h^{p+2}). \quad (1.3.11)$$

Rearranging Equation (1.3.11) provides us with a local error estimation of our p^{th} -order method, such that

$$|d_n| = |C_n| h^{p+1} \approx \frac{|q_{n+1}^h - q_{n+1}^{h/2}|}{1 - 2^{-p}}, \quad (1.3.12)$$

where for small enough values of h we can disregard the $\mathcal{O}(h^{p+2})$ term.

Computing Higher-Order Methods:

Error estimation is important as it can be used to help determine the appropriate step size, h , for a given method; however, it can be costly to compute error estimates at each time step. However, if our goal is to create a higher-order method, then we can use the local error estimate in (1.3.12) by adding it back into either Equation (1.3.10a) or (1.3.10b).

For example, substituting (1.3.12) into (1.3.10b) results in

$$\begin{aligned}
q_{n+1}^{h/2} - q(t_{n+1}) &= 2C_n(h/2)^{p+1} + \mathcal{O}(h^{p+2}), \\
&= \frac{C_n h^{p+1}}{2^p} + \mathcal{O}(h^{p+2}), \\
&= \left(\frac{q_{n+1}^h - q_{n+1}^{h/2}}{1 - 2^{-p}} \right) \left(\frac{1}{2^p} \right) + \mathcal{O}(h^{p+2}), \\
&= \frac{q_{n+1}^h - q_{n+1}^{h/2}}{2^p - 1} + \mathcal{O}(h^{p+2}).
\end{aligned} \tag{1.3.13}$$

Finally, if we rearrange Equation (1.3.13) as follows:

$$q(t_{n+1}) = q_{n+1}^{h/2} + \frac{q_{n+1}^h - q_{n+1}^{h/2}}{2^p - 1} + \mathcal{O}(h^{p+2}), \tag{1.3.14}$$

then the RHS of the above equation now represents a new approximation for the solution q at time t_{n+1} that is on the order of $\mathcal{O}(h^{p+2})$. Comparing this to Equations (1.3.10a) - (1.3.10b), which are on the order of $\mathcal{O}(h^{p+1})$, we have now combined the schemes with h and $\frac{h}{2}$ to achieve a higher order method. We define this approximation as:

$$\hat{q}_{n+1}^{h/2} = q_{n+1}^{h/2} + \frac{q_{n+1}^h - q_{n+1}^{h/2}}{2^p - 1}. \tag{1.3.15}$$

Notice that our new method in Equation (1.3.14) is simply a linear combination of two solutions computed using any p^{th} -order “base method” with step sizes of h and $h/2$. For example, if we compute q_{n+1}^h and $q_{n+1}^{h/2}$ in Equation (1.3.15) using forward Euler as our base method, then:

$$q_{n+1} = q_n + hf(t_n, q_n), \tag{1.3.16a}$$

$$q_{n+\frac{1}{2}} = q_n + \frac{h}{2}f(t_n, q_n), \text{ and} \tag{1.3.16b}$$

$$q_{n+1} = q_{n+\frac{1}{2}} + \frac{h}{2}f(t_{n+\frac{1}{2}}, q_{n+\frac{1}{2}}), \tag{1.3.16c}$$

where Equation (1.3.16a) represents the solution for q_{n+1}^h , and Equations (1.3.16b-1.3.16c) represent the solution for $q_{n+1}^{h/2}$. Additionally, since our example uses the explicit forward

Euler method where $p = 1$, then the higher order method for the solution to our IBVP simplifies to:

$$\hat{q}_{n+1}^{h/2} = 2q_{n+1}^{h/2} - q_{n+1}^h + \mathcal{O}(h^3). \quad (1.3.17)$$

This process of generating higher order methods using lower order base methods is known as “*the method of local extrapolation.*” Notice that the local error of Equation (1.3.17) is $\mathcal{O}(h^3)$ with global error on the order of $\mathcal{O}(h^2)$. Furthermore, if we want to achieve another higher order method with global error on the order of $\mathcal{O}(h^3)$, then we simply continue this process by generating another solution at time t_{n+1} using our base method with a step size smaller than $h/2$. For example, let’s next choose a step size of $h/3$.

We define this solution as $q_{n+1}^{h/3}$, such that its approximation and local error are as follows:

$$\begin{aligned} q_{n+\frac{1}{3}} &= q_n + \frac{h}{3}f(q_n), \\ q_{n+\frac{2}{3}} &= q_{n+\frac{1}{3}} + \frac{h}{3}f(q_{n+\frac{1}{3}}), \\ q_{n+1} &= q_{n+\frac{2}{3}} + \frac{h}{3}f(q_{n+\frac{2}{3}}), \\ q_{n+1}^{h/3} - q(t_{n+1}) &= 3C_n(h/3)^{p+1} + \mathcal{O}(h^{p+2}). \end{aligned} \quad (1.3.18)$$

Subtracting Equation (1.3.10b) from Equation (1.3.18) results in the following error estimate for the C_n term:

$$|C_n|h^{p+1} = \frac{|q_{n+1}^{h/3} - q_{n+1}^{h/2}|}{(3^{-p} - 2^{-p})} \approx |d_n|. \quad (1.3.19)$$

Therefore, substituting this error estimate back into Equation (1.3.18) and rearranging terms for the solution to $q(t_{n+1})$ results in

$$q(t_{n+1}) = q_{n+1}^{h/3} + \frac{q_{n+1}^{h/2} - q_{n+1}^{h/3}}{1 - 3^p 2^{-p}} + \mathcal{O}(h^{p+2}). \quad (1.3.20)$$

If we now take the RHS of the above equation to represent a new approximation for the

solution q at time t_{n+1} , where $p = 1$ since our chosen base method is forward Euler, then

$$\begin{aligned}\hat{q}_{n+1}^{h/3} &= q_{n+1}^{h/3} + \frac{q_{n+1}^{h/2} - q_{n+1}^{h/3}}{1 - 3/2} + \mathcal{O}(h^3), \\ &= 3q_{n+1}^{h/3} - 2q_{n+1}^{h/2} + \mathcal{O}(h^3).\end{aligned}\tag{1.3.21}$$

Note that both Equations (1.3.17) and (1.3.21) have local errors on the order of $\mathcal{O}(h^3)$. Therefore, we can now use both of these equations to remove the h^3 error terms, leaving us with a new method with local error $\mathcal{O}(h^4)$. Neglecting all of the intermediate steps, the final numerical approximation to the IBVP at time t_{n+1} is:

$$\hat{\hat{q}}_{n+1}^{h/3} = \frac{9}{2}q_{n+1}^{h/3} - 4q_{n+1}^{h/2} + \frac{1}{2}q_{n+1}^h + \mathcal{O}(h^4),\tag{1.3.22}$$

which is a linear combination of the three solutions q_{n+1}^h , $q_{n+1}^{h/2}$, and $q_{n+1}^{h/3}$, and is dependent upon the sequence of step sizes chosen: h , $h/2$, and $h/3$.

Generalizing the Extrapolation Process:

We have just shown how we can use local error estimation to achieve a higher order numerical approximation using a lower order base method. Therefore, let us now define the form of the asymptotic h -expansion of the global error for a base method of order p , using a step size of h as follows:

$$q_h - q(t) = e_p(t)h^p + \cdots + e_N(t)h^N + E_h(t)h^{N+1},\tag{1.3.23}$$

where $e_i(t)$ are the solutions of inhomogeneous differential equations and do not depend on the step size h , where E_h is bounded for $t_0 \leq t \leq t_{end}$, and $e_i(t_0) = 0$ for all i [9], [10], [11]. In the previous examples we chose step sizes of h , $h/2$, and $h/3$. More generally, if we define a sequence of positive integers

$$\{n_j\}_{1 \leq j \leq M}, \quad \text{with} \quad n_j < n_{j+1},\tag{1.3.24}$$

and a corresponding sequence of step sizes

$$\{h_j\}_{1 \leq j \leq M}, \quad \text{with} \quad h_j = \frac{H}{n_j},\tag{1.3.25}$$

then we can compute the numerical solution to the IBVP in Equation (1.3.1) using a p^{th} -order “base method” that we define as q_{h_j} , to numerically approximate $q(t)$ at time $t = t_0 + H$ using n_j steps of the base method with step size h_j and macro-step of H . More precisely, we define our base method as

$$q_{h_j}(t_0 + H) := T_{j,1} \quad \text{for } 1 \leq j \leq M, \quad (1.3.26)$$

where M represents the total number of numerical approximations using the p^{th} -order base method we have chosen. We then linearly combine the M approximations via the classical Richardson extrapolation process described in Section 1.3.1 to eliminate the $e_i(t)$ error terms within the asymptotic h-expansion of the global error equation shown in Equation (1.3.23).

It is described in [11] and proven in [12] that an efficient iterative process for computing each $T_{j,k}$ approximation is through the use of the Aitken-Neville algorithm:

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{\frac{n_j}{n_{j-k}} - 1}, \quad \text{for } k+1 \leq j \leq M. \quad (1.3.27)$$

Although there have been several sequence choices proposed for $\{n_j\}$, the most computationally efficient sequence to use is the “harmonic sequence” [12]

$$1, 2, 3, 4, 5, \dots \quad (1.3.28)$$

Additionally, it is convenient to represent all of the $T_{j,k}$ results using the following extrapolation tableau with associated p -order as described in [9].

Table 1.3.1. Tableau of $T_{j,k}$ Solutions with Corresponding Base Method Orders

(a) $T_{j,k}$ Solution Tableau					(b) Classical Orders for a p^{th} -order Base Method				
T_{11}					p				
T_{21}	T_{22}				p	$p+1$			
T_{31}	T_{32}	T_{33}			p	$p+1$	$p+2$		
$\vdots$	$\vdots$	$\vdots$	$\ddots$		$\vdots$	$\vdots$	$\vdots$	$\ddots$	
$T_{M,1}$	$T_{M,2}$	$T_{M,3}$	$\cdots$	$T_{M,M}$	p	$p+1$	$p+2$	$\cdots$	$p+M-1$

Note that Table 1.3.1(a) not only depicts the final solution $T_{M,M}$ that can be generated via the extrapolation process by iterating through the entire $\{n_j\}$ sequence, but it also shows in 1.3.1(b) the usefulness of extrapolation in producing a sequence of lower-order embedded methods, which can easily be used for local error estimation, step size control and variable order strategies as shown by Hairer in [11]. Furthermore, the order of each $T_{j,k}$ solution is on the order of $\mathcal{O}(h^{p+k-1})$.

For completeness of our previous extrapolation examples within this section, we have conveniently listed the following numerical solution approximations in their equivalent $T_{j,k}$ annotation with reference to their equation numbers:

$$T_{11} := q_{n+1}^h, \quad (1.3.16a)$$

$$T_{21} := q_{n+1}^{h/2}, \quad (1.3.16c) \quad T_{22} := \hat{q}_{n+1}^{h/2}, \quad (1.3.17)$$

$$T_{31} := q_{n+1}^{h/3}, \quad (1.3.18) \quad T_{32} := \hat{q}_{n+1}^{h/3}, \quad (1.3.21) \quad T_{33} := \hat{\hat{q}}_{n+1}^{h/3}. \quad (1.3.22)$$

Fixed Iteration Extrapolation Methods:

As previously mentioned in the introduction (Section 1.3), when the above extrapolation scheme uses a fixed number of iterations, the entire process can be considered as a one-step explicit Runge-Kutta (ERK) method. In other words, each $T_{j,k}$ solution in Table 1.3.1(a) is an ERK method [10], [11]. We discuss ERK methods in more detail in Chapter 2.

When using forward Euler as the base method, the extrapolation process results in an s -stage ERK method, with $s = \frac{P^2-P+2}{2}$ stages [11] and P represents the desired global accuracy. For example, to compute the T_{33} solution above, which has global error of order $\mathcal{O}(h^3)$, then $p = 1$ and $P = 3$. Therefore, the number of stage evaluations for the new ERK method is $s = \frac{3^2-3+2}{2} = 4$. Simply put, T_{33} represents a 4-stage ERK method that is 3^{rd} order accurate.

Finally, since T_{33} is simply a linear combination of the solutions $T_{j,1}$ for $j = 1, \dots, 3$, as shown in Equation (1.3.22), then we can write this equation as follows:

$$T_{33} = \frac{9}{2}T_{31} - 4T_{21} + \frac{1}{2}T_{11} + \mathcal{O}(h^3). \quad (1.3.29)$$

Since we know that each of the $T_{j,1}$ solutions in Equation (1.3.29) is generated using forward

Euler with steps sizes h , $h/2$, and $h/3$, then neglecting the $\mathcal{O}(h^3)$ term we can show that

$$\begin{aligned}
T_{33} = & \frac{9}{2}q_n + \frac{3h}{2} \left[f(q_n) + f\left(q_n + \frac{h}{3}f(q_n)\right) + f\left(q_n + \frac{h}{3}[f(q_n) + f(q_n + \frac{h}{3}f(q_n))]\right) \right] \\
& - 4q_n - 2h \left[f(q_n) + f\left(q_n + \frac{h}{2}f(q_n)\right) \right] \\
& + \frac{1}{2}q_n + \frac{h}{2}f(q_n).
\end{aligned} \tag{1.3.30}$$

Next, adding like terms results in the final expression:

$$T_{33} = q_n - 2hf(q_n + \frac{h}{2}f(q_n)) + \frac{3h}{2}f(q_n + \frac{h}{3}f(q_n)) + \frac{3h}{2}f\left(q_n + \frac{h}{3}[f(q_n) + f(q_n + \frac{h}{3}f(q_n))]\right). \tag{1.3.31}$$

Additionally, Equation (1.3.31) can be written in Butcher tableau form as the following ERK method:

$$\begin{array}{c|ccc}
0 & & & \\
\frac{1}{2} & \frac{1}{2} & & \\
\frac{1}{3} & \frac{1}{3} & 0 & \\
\frac{2}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\
\hline
& 0 & -2 & \frac{3}{2} & \frac{3}{2}
\end{array}, \tag{1.3.32}$$

where we have color coded the coefficients in both Equations (1.3.31) - (1.3.32) for easier comparison. Furthermore, it may be easier to view Equation (1.3.31) in its ERK stage formula, where we have let $h = \Delta t$:

$$\begin{aligned}
K_1 &= f(t_n, q_n), \\
K_2 &= f\left(t_n + \frac{\Delta t}{2}, q_n + \frac{\Delta t}{2}K_1\right), \\
K_3 &= f\left(t_n + \frac{\Delta t}{3}, q_n + \frac{\Delta t}{3}K_1\right), \\
K_4 &= f\left(t_n + \frac{2\Delta t}{3}, q_n + \frac{\Delta t}{3}[K_1 + K_3]\right), \\
T_{33} &:= q_{n+1} = q_n + \frac{\Delta t}{2}(-4K_2 + 3K_3 + 3K_4),
\end{aligned} \tag{1.3.33}$$

We discuss the above Butcher tableau form in more detail within Chapter 2.

Explicit-Euler Extrapolation Algorithm:

In the above section we have shown how the extrapolation process works using explicit forward Euler as the base method; however, it is useful to view the process outlined in Equations (1.3.24) - (1.3.28) using Algorithm 1.3.1, which can also be found in [13].

Algorithm 1.3.1 Explicit-Euler (Richardson) Extrapolation

for $k = 1 \rightarrow M$ do $Y_{k,0} = q_n$ for $j = 1 \rightarrow k$ do $Y_{k,j} = Y_{k,j-1} + \frac{H}{n_j} f(Y_{k,j-1})$ end for $T_{k,1} = Y_{k,k}$ end for for $k = 2 \rightarrow M$ do for $j = k \rightarrow M$ do $T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{j/(j-k+1) - 1}$ end for end for $q_{n+1} = T_{M,M}$ $\hat{q}_{n+1} = T_{M-1,M-1}$	$\diamond$ Compute first-order approximations using explicit-Euler method with $h_j = \frac{H}{n_j}$ step-sequence $\diamond$ Note: $n_j = k$ for harmonic sequence $\diamond$ Perform extrapolation using Aitken-Neville formula to order k $\diamond$ New higher-order solution value $\diamond$ Embedded method solution value
---	---

Although it is not required, we can also store $\hat{q}_{n+1} = T_{M-1,M-1}$, which is the embedded method solution value and can be used for error estimation and/or step-size selection.

1.3.2 Gragg Extrapolation

As shown in the previous section, extrapolation offers a simple way to obtain higher order solutions using lower order methods. In this section we briefly review a significant contribution to extrapolation due to Gragg [10]. Specifically, he proved that by using the midpoint rule as the “base method” for extrapolation, the global error function contains only even powers of h , such that its asymptotic expansion is in powers of h^2 [10], as opposed to Equation (1.3.23) which is in terms of h when using forward Euler as the base method.

To solve the IBVP in Equation (1.3.1), Gragg's method is defined as:

$$\begin{aligned}
q_0 &= q(t_0) \\
q_1 &= q_0 + hf(t_0, q_0) \\
q_{i+1} &= q_{i-1} + 2hf(t_i, q_i), \quad \text{for } i = 1, 2, \dots, n-1, \\
S_h(t_n) &:= q_{n+1} = \frac{1}{2}(q_n + q_{n-1} + hf(t_0 + H, q_n))
\end{aligned} \tag{1.3.34}$$

where $t_i = t_0 + ih$, and h is determined by the sequence $\{h_j\}$ [10], [11]. Additionally, the final step in the method computes the solution $S_h(t)$, which is used as a smoothing step (weighted average) for improved stability.

Since the midpoint method is a multi-step method which solves for q_{i+1} using information from q_i and q_{i-1} , it requires a starting method. Gragg recommended using the explicit forward Euler scheme for both convenience and stability. More importantly, he proved that the specific choice of using forward Euler as the starting method ensured the error function for the method would have an asymptotic expansion in powers of h^2 [14].

Neglecting the smoothing step in Equation (1.3.34), the error function has the following expansion as shown by Shampine [14], provided that the function, $f(t, q(t))$, from the IBVP in Equation (1.3.1) is sufficiently smooth:

$$q_h - q(t) = \sum_{k=1}^M h^{2k} [e_k(t) + (-1)^{(t-t_0)/h} \tilde{e}_k(t)] + \mathcal{O}(h^{2M+2}). \tag{1.3.35}$$

Notice that the coefficient in front of the $\tilde{e}_k(t)$ term in Equation (1.3.35) has an oscillatory behavior depending on even or odd time intervals. Therefore, Gragg cleverly defined the following function:

$$v_n = \frac{1}{2}[q_n + q_{n-1} + hf(t_n, q_n)], \tag{1.3.36}$$

to remove the oscillations. Notice that the midpoint rule:

$$q_{n+1} = q_{n-1} + 2hf(t_n, q_n), \tag{1.3.37}$$

can be written in terms of q_{n-1} , such that when substituted into Equation (1.3.36), the

function v_n can also be written as:

$$v_n = \frac{1}{2}[q_n + q_{n+1} - hf(t_n, q_n)]. \quad (1.3.38)$$

Next, by adding Equations (1.3.36) and (1.3.38) together, we find:

$$v_n = \frac{1}{4}q_{n-1} + \frac{1}{2}q_n + \frac{1}{4}q_{n+1}, \quad (1.3.39)$$

where Gragg proved that the Taylor series expansion in h for $v(t)$ has the following form:

$$v(t) = q(t) + h^2[e_1(t) + \frac{1}{4}q''(t)] + \sum_{k=2}^M h^{2k}[e_k(t) + (-1)^{(t-t_0)/h}\tilde{e}_k(t)] + \mathcal{O}(h^{2M+2}). \quad (1.3.40)$$

Notice that the oscillatory behavior in the leading term of the asymptotic error function has now been removed. Therefore, as previously mentioned, the last step in Gragg's method (1.3.34) was to compute the final solution using Equation (1.3.36), which he defined as the “smoothing” function $S_h(t)$, where

$$S_h(t_n) := v_n = \frac{1}{2}[q_n + q_{n-1} + hf(t_n, q_n)]. \quad (1.3.41)$$

In [14], Shampine states that the smoothing function is an unnecessary step when performing extrapolation using the midpoint rule, as the process of extrapolation naturally removes the first error terms. He further provides an explanation in [15] for performing extrapolation without the use of the $S_h(t)$ function. Additionally, Shampine's conclusions are supported by Deuflhard and Bornemann in [16] when they state “Formerly, Gragg's smoothing step was thought to promise a better-behaved discretization. But the resulting improvement in the total computational time observed in practice has been found marginal at best.” It should be noted that the smoothing function is recommended when using the midpoint (leapfrog) method in the absence of an extrapolation process as increased stability has been shown. This is known as the Robert-Asselin time filter and new filters can be found in a paper by Williams [17].

This does not discount Gragg's method; it simply implies that the smoothing step is not a necessary requirement when used in conjunction with the extrapolation process. Further-

more, his results still show that the midpoint rule is an excellent choice for extrapolation based on the h^2 error expansion from Equation (1.3.35). Therefore, it can be shown that the extrapolation process eliminates two powers of h at each iteration when using the midpoint rule as the base method for extrapolation.

Aitken-Neville Algorithm for Symmetric Methods:

Finally, it is shown in both [10] and [11] that Gragg's method leads to a slight modification in the Aitken-Neville algorithm in Equation (1.3.27), such that the n_j/n_{j-k} term in the numerator must now be squared when computing the $T_{j,k}$ approximations, such that the modified formula is now:

$$T_{j,k+1} = T_{j,k} + \frac{T_{j,k} - T_{j-1,k}}{\left(\frac{n_j}{n_{j-k}}\right)^2 - 1}, \quad \text{for } k+1 \leq j \leq M. \quad (1.3.42)$$

Step-Sequence Selection:

As with Richardson extrapolation, Gragg's extrapolation method also requires a step-sequence, $\{n_j\}$, to be selected for the iterative process. However, since the midpoint rule is a symmetric method, the sequence of n_j terms is required to be even. Common choices for symmetric methods are:

$$\{n_j\} = 2, 4, 8, 16, 32, 64, \dots, \quad \text{where } n_j = 2n_{j-1}, \quad \text{for } j = 2, 3, 4, \dots \quad (1.3.43a)$$

$$\{n_j\} = 2, 4, 6, 8, 12, 16, \dots, \quad \text{where } n_j = 2n_{j-2}, \quad \text{for } j = 4, 5, 6, \dots \quad (1.3.43b)$$

$$\{n_j\} = 2, 4, 6, 8, 10, 12, \dots, \quad \text{where } n_j = 2j, \quad \text{for } j = 1, 2, 3, \dots \quad (1.3.43c)$$

The sequence in Equation (1.3.43a) was proposed by Romberg, Bulirsch and Stöer [18] proposed the sequence in Equation (1.3.43b), and Deuffhard [12], [19] suggested that the sequence in Equation (1.3.43c) is the most efficient of the three for higher order extrapolation, in terms of minimizing the total number of RHS evaluations computed.

Fixed Iteration Extrapolation:

Just as we saw with Richardson's extrapolation, Gragg also states in [10] that when his method "is coupled with the Neville scheme using a fixed number of extrapolations per step h_0 , then the entire process is a Runge-Kutta (one-step) method."

Furthermore, when using the explicit midpoint rule as the "base method", the extrapolation

process results in an s -stage ERK method, where $s = \frac{P^2+4}{4}$ stages [11] and P represents the desired higher order approximation on the order of $\mathcal{O}(h^P)$. Because the method results in even powers of h , we only seek higher order approximations for $P = 4, 6, 8, \dots$, etc.

Gragg Extrapolation Algorithm for Fixed Order:

We now show Gragg's extrapolation method in Algorithm 1.3.2 for fixed order approximations, where “the number of stages increases quadratically with the order” [13]. The step sequence used within the algorithm is Equation (1.3.43c) due to its proven efficiency [12], [19] over the other two sequences.

Algorithm 1.3.2 Explicit-Midpoint (Gragg) Extrapolation

$r = P/2$ for $k = 1 \rightarrow r$ do $Y_{k,0} = q_n$ $Y_{k,1} = Y_{k,0} + \frac{H}{2k} f(Y_{k,0})$ for $j = 2 \rightarrow 2k$ do $Y_{k,j} = Y_{k,j-2} + \frac{H}{k} f(Y_{k,j-1})$ end for $T_{k,1} = Y_{k,2k}$ end for for $k = 2 \rightarrow r$ do for $j = k \rightarrow r$ do $T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\frac{j^2}{(j-k+1)^2} - 1}$ end for end for $q_{n+1} = T_{r,r}$ $\hat{q}_{n+1} = T_{r-1,r-1}$	$\diamond$ Compute second-order approximations using Deuffhard step-sequence $\diamond$ Initial explicit-Euler step $\diamond$ Midpoint steps $\diamond$ Perform Extrapolation using Aitken-Neville formula for symmetric methods to order $2k$ $\diamond$ New higher-order solution value $\diamond$ Embedded method solution value
---	--

Notice that if we only desire a second order approximation to the IBVP, then we choose

$P = 2$, such that Algorithm 1.3.2 reduces to:

$$\begin{aligned}
P = 2 &\rightarrow r = 1, \\
Y_{10} &= q_n, \\
Y_{11} &= Y_{10} + \frac{H}{2}f(Y_{10}), \\
T_{11} &:= Y_{12} = Y_{10} + Hf(Y_{11}),
\end{aligned} \tag{1.3.44}$$

which is equivalent to the midpoint rule (RK22) found in Equation (2.1.10) of Chapter 2. Since Gragg's method uses a second order base method, there is no extrapolation procedure required unless $P \geq 4$. Note, Gragg's extrapolation method is originally intended for non-fixed order extrapolation where the number of extrapolation steps varied during time stepping to achieve a desired tolerance.

To see the correspondence between Gragg and RK we let $P = 4$ so that $r = 2$, which gives the final solution approximation by Algorithm 1.3.2:

$$T_{22} = \frac{4}{3}T_{21} - \frac{1}{3}T_{11}. \tag{1.3.45}$$

Using the definitions for T_{11} and T_{21} , Equation (1.3.44) is equivalent to the following RK scheme:

$$\begin{aligned}
Y_{20} &= q_n, \\
Y_{21} &= Y_{20} + \frac{H}{4}f(Y_{20}), \\
Y_{22} &= Y_{20} + \frac{H}{2}f(Y_{21}), \\
Y_{23} &= Y_{21} + \frac{H}{2}f(Y_{22}), \\
T_{21} &:= Y_{24} = Y_{22} + \frac{H}{2}f(Y_{23}).
\end{aligned} \tag{1.3.46}$$

1.3.3 Richardson versus Gragg Extrapolation Comparison

In this subsection we compare the efficiency and order of accuracy of the two extrapolation methods in Algorithms (1.3.1) and (1.3.2), using both the linear SWE and non-linear SWE as described by Equations (B.0.1) - (B.0.4) and Equations (1.1.11) - (1.1.13), respectively.

Although we care about the order of accuracy of the extrapolation methods, we must also consider their efficiency, which we measure by computing their operation count.

Operation Count:

In general, the operation count of a method is simply determined by looking at the most time consuming (i.e., expensive) operation within the algorithm. For ODE solvers, the most expensive operation is the RHS function evaluation of $f(Y_{k,j})$.

In comparison to other well-known SR RK methods such as RK2 (midpoint rule) or the classical RK4, which require two and four functional evaluations (per time-step) respectively, the explicit forward Euler extrapolation method in Section 1.3.1 requires three functional evaluations to achieve second order accuracy and 10 functional evaluations per time-step for a fourth order approximation. Likewise, the Gragg extrapolation process requires two and six functional evaluations per time-step for second and fourth order accuracy, respectively. This can be seen in Table 1.3.2, which compares the total number of RHS evaluations per time-step for the IBVP in Equation (1.3.1) using both the Richardson and Gragg extrapolation methods.

Table 1.3.2. $f(Y_{j,k})$ Functional Evaluation Comparison for Richardson vs. Gragg Extrapolation

Order of Accuracy	Number of RHS Functional Evaluations (per time-step)									
	1 st	2 nd	3 rd	4 th	5 th	6 th	7 th	8 th	9 th	10 th
Explicit-Euler (Richardson) Extrap	1	3	6	10	15	21	28	36	45	55
Explicit-Midpoint (Gragg) Extrap	-	2	-	6	-	12	-	20	-	30

The results in Table 1.3.2 can be expressed in closed form using the following equations:

$$\text{RHS Evaluations (Richardson)} = \frac{M^2 + M}{2}, \quad M = 1, 2, 3, 4, \dots, \quad (1.3.47a)$$

$$\text{RHS Evaluations (Gragg)} = \frac{P^2 + 2P}{4}, \quad P = 2, 4, 6, 8, \dots \quad (1.3.47b)$$

When comparing the two extrapolation methods we have discussed within this chapter, Gragg's method is theoretically more efficient than Richardson's since its asymptotic error expansion is in powers of h^2 . However, we find that this efficiency also depends on

the computational difficulty of the RHS evaluations. For example, if the RHS function is a simple calculation, then lower order ($P < 4$) Richardson extrapolations might still be computationally more efficient as the total computation of each method is not solely dependent on the RHS. We now use our IBVP to numerically verify the results from Table 1.3.2.

Work-Precision Plots:

In Figure 1.3.1 we show the results for Richardson and Gragg extrapolation methods, where we solved the linear SWE from Equations (B.0.1) - (B.0.4). For simplicity, we assumed $g = h_B = 1$ and used a wave frequency constant of $c = 8$ within the initial condition from Equation (B.0.3). Furthermore, the spatial discretization method used an inexact DG method with 16^{th} order polynomials within 25 elements. The decision to use a high order spatial method is because we are interested in analyzing the error of the two extrapolation time-integration schemes.

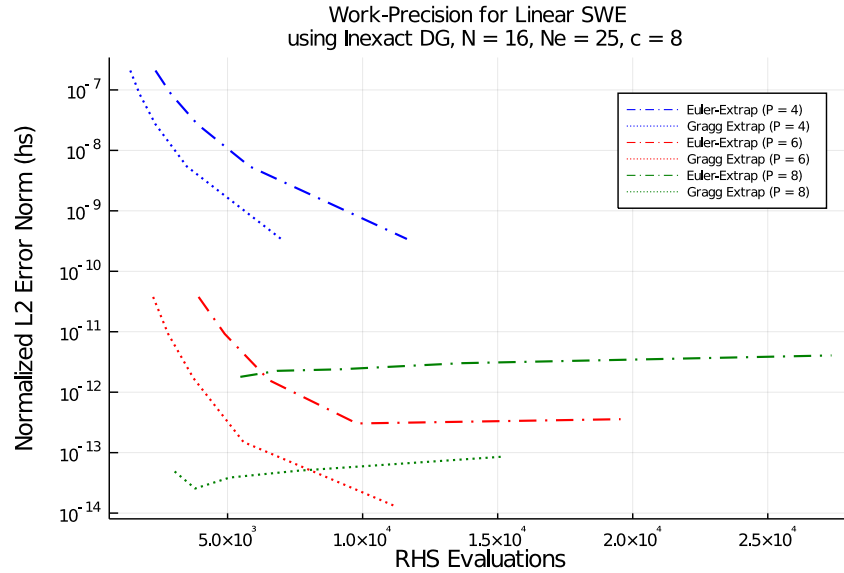


Figure 1.3.1. Work-Precision Plot for Linear SWE Using Explicit-Euler (Richardson) vs. Explicit-Midpoint (Gragg) Extrapolation.

Notice that Figure 1.3.1 only displays the error results for h_s as the results for U are similar. Here we used values of $P = 4, 6$, and 8 , for both methods, where it is easily seen that the Gragg extrapolation method is more efficient than Richardson's method for all values of

P selected. Furthermore, Gragg's method also achieves a higher order of accuracy as the step-size decreases for $P > 4$. Finally, all results are computed using time, $0 \leq t \leq 0.4$ and values of $P \geq 10$ are not used as the time integration schemes already achieve the best accuracy possible given the spatial discretization parameters.

Let's now look at the results for these two methods when solving the more difficult non-linear SWE from Equations (1.1.11) - (1.1.13), where the RHS function is defined as,

$$f(\mathbf{Y}_{k,j}) = -\frac{\partial}{\partial x} \left(\frac{U}{h_s+1} \right) - \frac{\partial}{\partial x} \left(\begin{matrix} 0 \\ h_s + \frac{1}{2}h_s^2 \end{matrix} \right), \quad (1.3.48)$$

for $g = h_B = 1$. Using the same assumptions and parameters as before, the work-precision results for the non-linear SWE can be seen in Figures 1.3.2a and 1.3.2b.

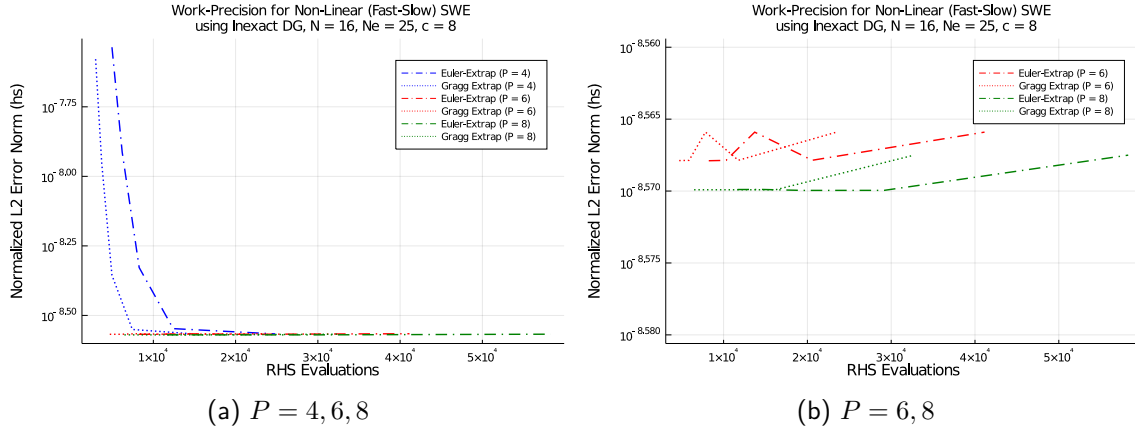


Figure 1.3.2. Work-Precision Plot for Non-Linear SWE Using Richardson vs. Gragg Extrapolation.

Notice that when solving the non-linear shallow water equations with both extrapolation methods, Gragg's method is once again noticeably faster (i.e. uses fewer RHS evaluations) than Richardson's method, as shown in Figure 1.3.2a, which displays the results for $P = 4, 6$, and 8 . We have also including the zoomed in results for $P = 6$ and 8 in Figure 1.3.2b. However, we do notice that the accuracy achieved for all of the non-linear solutions was significantly worse than the solution accuracy observed when solving the linear problem shown in Figure 1.3.1.

For example, using $P = 8$, Gragg's method achieved approximately 10^{-14} accuracy for the linear SWE as opposed to approximately $10^{-8.5}$ when solving the non-linear problem. Therefore, to ensure the decrease in accuracy is not attributed to the respective time-integration schemes, we recomputed each time-integration method using the DG method with 32^{nd} order polynomials in each spatial element.

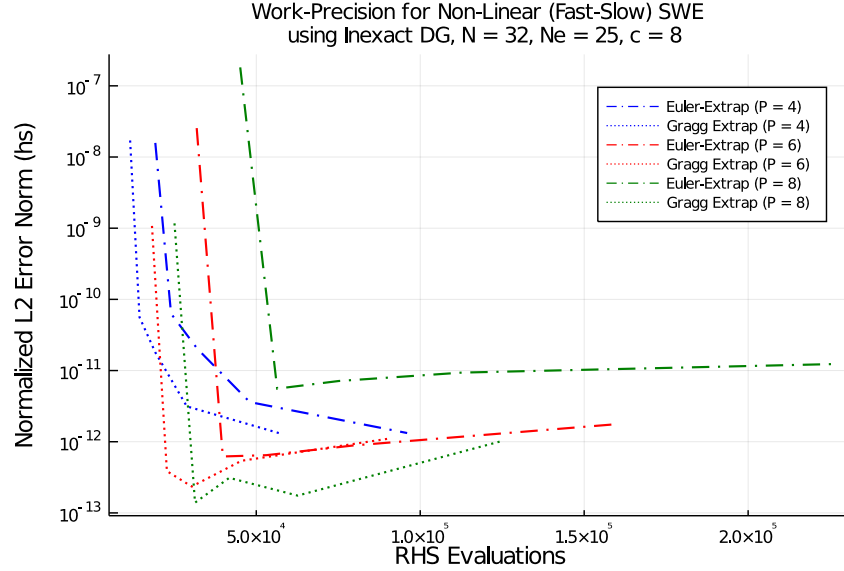


Figure 1.3.3. Extrapolation Work-Precision Plot for Non-Linear SWE using Inexact DG, with 32^{nd} Order Polynomials.

Results are found in Figure 1.3.3, where we now observe that the numerical accuracy of both extrapolation methods is improved when solving the non-linear problem, by simply using a higher order spatial DG method. Therefore, the lower accuracy seen in Figure 1.3.2 was attributed to the spatial error of the method, and not the time-integration schemes. Furthermore, the above figure once again highlights that Gragg's method is more efficient than Richardson's for all choices of P used.

Since all of the work-precision results were computed for time on the interval of $0 \leq t \leq 0.4$, we can infer that Gragg's method would be even more efficient over Richardson's, if we were to run our computations on a much longer time interval. Furthermore, the theoretical results from Table 1.3.2 are supported by the numerical results within the work-precision plots above. For example, the explicit-Euler scheme requires nine more RHS function

evaluations when $P = 6$ and a total of 16 more RHS evaluations when $P = 8$. These additional RHS computations when using Richardson’s method are why we are seeing the improved efficiency of Gragg’s extrapolation method within both Figures 1.3.1 and 1.3.2. Furthermore, the total number of RHS computations for each method becomes even more significant depending on the dimensionality of the problem. For example, our results were only computed in one-dimension (1D), where we can assume that if the problem were increased to 2D or 3D, we would find that Gragg’s method would be even more efficient than Richardson’s method.

Finally, it is important to note that we only showed results for $P \leq 8$ within the work-precision plots, as we found there is no advantage in constructing a higher order extrapolation method (i.e., $P \geq 10$), in terms of the accuracy achieved versus computational efficiency, with regards to the test IBVP we are solving. To further demonstrate this, we will now display a comparison of the order of accuracy and convergence rates between the two methods.

Order of Accuracy and Convergence Rates Plots:

Before we move on, we need to define the maximum stable Courant value. We previously mentioned in Section 1.2.3 our choice of using Lobatto points within each spatial element to construct the N^{th} order interpolating polynomials. These points are based on the roots of the Lobatto polynomial where each spatial element within our inexact DG method consists of $N + 1$ Lobatto points, which are used for both interpolation and quadrature. One benefit in using Lobatto points is that they are not equally spaced throughout the element. Instead, the points become increasingly closer to one another at the elemental boundaries, which is beneficial when constructing high order polynomials through these points. Furthermore, the minimum distance between any two points within an element is determined by computing $\Delta x_{min} = x_2 - x_1$ for $x \in [x_1, x_{N+1}]$.

For hyperbolic problems, the overall numerical approximation for a single-rate method is advanced in time using a specific choice of time-step size, Δt ; however, the maximum stable step-size is limited by the smallest Δx within the spatial discretization of each element (i.e., Δx_{min}). The relationship between Δx_{min} and Δt is known as the Courant value (σ) or the Courant–Friedrichs–Lewy (CFL) condition, such that the maximum Courant value for our 1D IBVP of the SWE is determined by:

$$\sigma_{max} = (|u| + \sqrt{gh}) \frac{\Delta t_{max}}{\Delta x_{min}}, \quad (1.3.49)$$

where $|u|$ represents the slow wave speed and $\sqrt{gh}$ represents the fast wave speed. A brief discussion on stability of these methods is provided in Section 1.3.4.

Continuing on with our results, Figures 1.3.4a and 1.3.4b show the order of accuracy and convergence rates for solving the linear SWE using Richardson and Gragg extrapolation, respectively. Notice that there is no benefit in constructing the explicit-Euler extrapolation for $P \geq 8$ as the 7th order extrapolation already achieves approximately 10^{-12} accuracy for its largest stable Courant value ($\sigma \approx 2.25$). Likewise, Gragg's method achieves its best accuracy at approximately $10^{-13.5}$ for its largest stable Courant value ($\sigma \approx 2.45$) using $P = 8$, with no added benefit in constructing $P \geq 10$ order extrapolations. Additionally, the convergence rates for each method match the theoretical order of accuracy we expect to see.

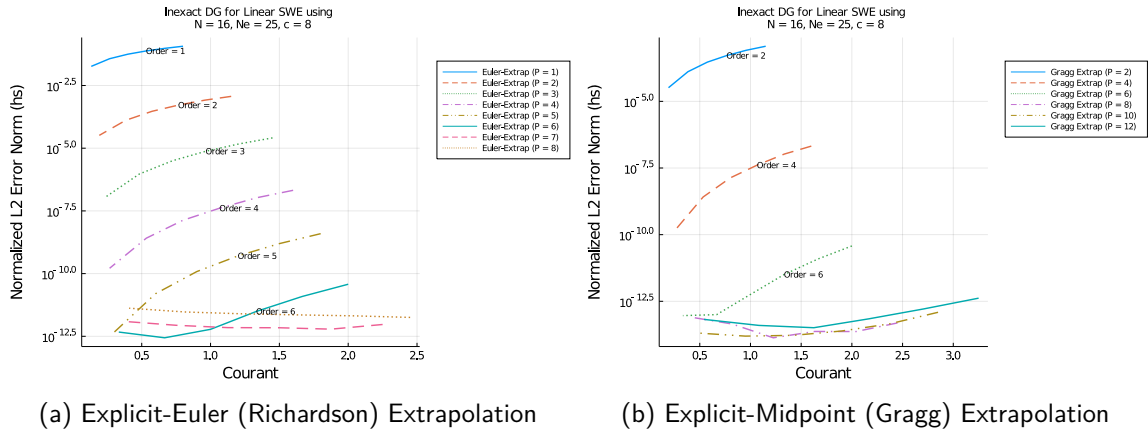


Figure 1.3.4. Order of Accuracy / Convergence Rates Plot for Linear SWE Using Richardson vs. Gragg Extrapolation.

For example, within both Figures 1.3.4 and 1.3.5 we display on each line the method's respective order of accuracy, such that the convergence rates are computed as follows:

$$\text{convergence order} = \frac{\log\left(\frac{norm_1}{norm_2}\right)}{\log\left(\frac{\Delta t_1}{\Delta t_2}\right)}, \quad (1.3.50)$$

where $norm_i$ is the L2-error norm of the method with respect to Δt_i .

Let's now look at the non-linear results, where we will use 32^{nd} order polynomials to construct our DG method. Figures 1.3.5a and 1.3.5b show the order of accuracy and convergence rates for solving the non-linear SWE using Richardson and Gragg extrapolation, respectively. In the case of the non-linear problem, Richardson extrapolation achieves the best accuracy at approximately 10^{-12} for $P = 5$ and $\sigma \approx 0.85$, with no benefit in constructing the explicit-Euler extrapolations for $P \geq 6$. Likewise, Gragg's method also achieves its best accuracy at approximately 10^{-13} for largest stable Courant value of $\sigma \approx 1.125$ using $P = 10$; however, there is no added benefit in constructing Gragg extrapolations for $P > 6$, as the 6^{th} order method achieves roughly the same level of accuracy as the 8^{th} and 10^{th} order extrapolations, while using significantly fewer RHS evaluations.

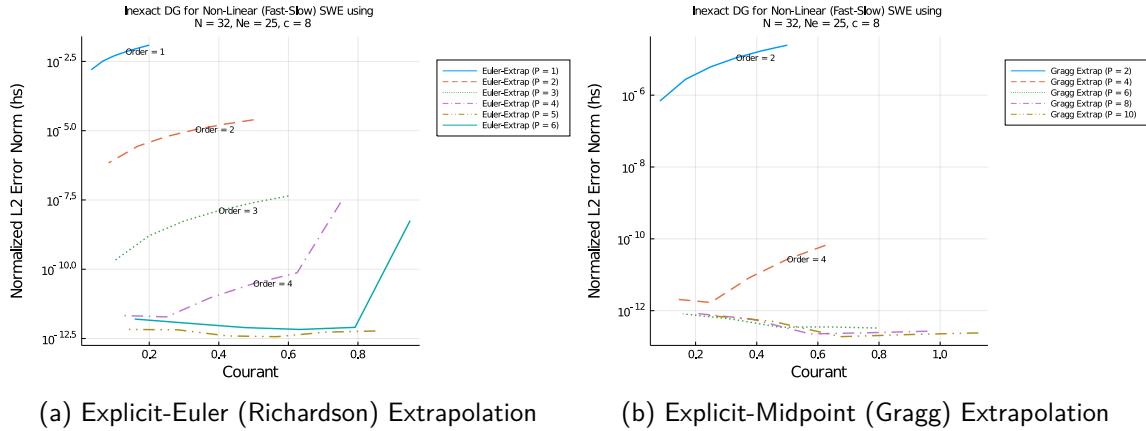


Figure 1.3.5. Order of Accuracy / Convergence Rates Plot for Non-Linear SWE Using Richardson vs. Gragg Extrapolation.

This can be seen in Table 1.3.3, which compares the wall clock times, total RHS evaluations and accuracy for the Gragg extrapolation method for increasing values of P .

In summary, the numerical results we achieved are consistent with the theoretical order of accuracy and convergence rates we expect when solving both the linear and non-linear shallow water equations. Furthermore, our results highlight the benefit in using Gragg's extrapolation method as it performed slightly better than Richardson's when considering both the computational efficiency and accuracy regardless of the choice of P .

Table 1.3.3. RHS Evaluation Comparison for Gragg Extrapolation

Non-Linear SWE, $n = 32, Ne = 25, c = 8, g = 1, h_B = 1$					
P - Order	Max Courant (σ)	Time (S)	Total RHS Evals	Accuracy (h_S)	Accuracy (U)
2	0.417	34.98	6908	1.717e-5	1.674e-4
4	0.625	66.91	13830	6.611e-11	5.844e-10
6	0.792	89.65	21828	2.351e-13	2.852e-12
8	0.959	132.29	30040	1.517e-13	1.626e-12
10	1.125	157.21	38430	2.379e-13	3.006e-12

1.3.4 Internal Stability Analysis of Extrapolation Methods

In this subsection we briefly discuss the internal stability of both Richardson and Gragg's extrapolation methods for fixed iterations. As previously shown, when extrapolation methods are used with a fixed number of iterations, they can be viewed as ERK methods; therefore, the same internal stability and error propagation analysis techniques performed on Runge-Kutta methods can be used to analyze the stability regions of the extrapolation algorithms.

This analysis typically begins with approximating the solution of the scalar IBVP we have seen in Equation (1.3.1):

$$\begin{aligned} q'(t) &= \lambda q(t), \quad \lambda \in \mathbb{C}, \\ q(t_0) &= q_0, \end{aligned} \tag{1.3.51}$$

such that Equation (1.3.51) is modified where $f(q) = \lambda q$, $z = \lambda\tau$, and $\tau := H = t_{n+1} - t_n$.

Internal Stability Polynomial of Explicit-Euler (Richardson) Extrapolation:

If we apply Algorithm 1.3.1 to the scalar IBVP problem in (1.3.51), then:

$$\begin{aligned} \tilde{Y}_{k,j} &= \tilde{Y}_{k,j-1} + \frac{H}{k} f(\tilde{Y}_{k,j-1}) + \tilde{r}_{k,j}, \\ &= \tilde{Y}_{k,j-1} + \frac{\lambda H}{k} \tilde{Y}_{k,j-1} + \tilde{r}_{k,j}, \\ &= \tilde{Y}_{k,j-1} + \frac{z}{k} \tilde{Y}_{k,j-1} + \tilde{r}_{k,j}, \\ &= \left(1 + \frac{z}{k}\right) \tilde{Y}_{k,j-1} + \tilde{r}_{k,j}, \end{aligned} \tag{1.3.52}$$

where we have also added the internal residual error term, $\tilde{r}_{k,j}$, to the method, such that $\tilde{r}_{k,0} = 0$. Note, these errors arise due to roundoff and can affect the accuracy of the solution. Continuing with Algorithm 1.3.1, each $T_{k,1}$ solution is then equivalent to its corresponding $\tilde{Y}_{k,k}$ solution, such that:

$$T_{11} \equiv \tilde{Y}_{11} = \left(1 + z\right) \tilde{Y}_{10} + \tilde{r}_{11}, \quad (1.3.53a)$$

$$\begin{aligned} T_{21} \equiv \tilde{Y}_{22} &= \left(1 + \frac{z}{2} + \frac{z}{2} + \left(\frac{z}{2}\right)^2\right) \tilde{Y}_{20} + \tilde{r}_{22} + \left(1 + \frac{z}{2}\right) \tilde{r}_{21}, \\ &= \left(1 + \frac{z}{2}\right)^2 \tilde{Y}_{20} + \sum_{\ell=1}^2 \left(1 + \frac{z}{2}\right)^{2-\ell} \tilde{r}_{2,\ell}, \end{aligned} \quad (1.3.53b)$$

$$\begin{aligned} T_{31} \equiv \tilde{Y}_{33} &= \left(1 + \frac{z}{3}\right)^3 \tilde{Y}_{30} + \sum_{\ell=1}^3 \left(1 + \frac{z}{3}\right)^{3-\ell} \tilde{r}_{3,\ell}, \\ &\vdots \end{aligned} \quad (1.3.53c)$$

which implies that the general form of each $T_{k,1}$ solution for the explicit-Euler extrapolation method (i.e., $T_{k,1}^{\text{EE}}$) can be found by:

$$T_{k,1}^{\text{EE}} \equiv \tilde{Y}_{k,k} = \left(1 + \frac{z}{k}\right)^k q_n + \sum_{\ell=1}^k \left(1 + \frac{z}{k}\right)^{k-\ell} \tilde{r}_{k,\ell}. \quad (1.3.54)$$

Notice that we have replaced the $T_{k,0}$ terms with q_n based on the starting solution of each step within the algorithm. The result in Equation (1.3.54) can also be found in the work by Ketcheson [20].

The next step in the algorithm is to perform extrapolation using the Aitken-Neville formula, where [20] shows that each $T_{M,M}^{\text{EE}}$ solution for $M \geq 1$ can be written as:

$$T_{M,M}^{\text{EE}} = \sum_{k=1}^M \frac{(-1)^{k+M} k^{M-1}}{(M-k)!(k-1)!} T_{k,1}^{\text{EE}}, \quad (1.3.55)$$

where the $T_{k,1}^{\text{EE}}$ term in Equation (1.3.55) can be replaced by Equation (1.3.54). Therefore, combining Equations (1.3.54) and (1.3.55) together we have:

$$T_{M,M}^{\text{EE}} = \sum_{k=1}^M \frac{(-1)^{k+M} k^{M-1}}{(M-k)!(k-1)!} \left[\left(1 + \frac{z}{k}\right)^k q_n + \sum_{\ell=1}^k \left(1 + \frac{z}{k}\right)^{k-\ell} \tilde{r}_{k,\ell} \right], \quad (1.3.56)$$

Finally, the stability polynomial $P(z)$ of the method is determined by consolidating the q_n coefficient in Equation (1.3.56), where

$$P^{\text{EE}}(z) = \sum_{k=1}^M \frac{(-1)^{k+M} k^{M-1}}{(M-k)!(k-1)!} \left(1 + \frac{z}{k}\right)^k. \quad (1.3.57)$$

Notice that if $M = 1$ in Equation (1.3.60), then

$$P^{\text{EE}}(z) = \sum_{k=1}^1 \frac{(-1)^{k+1}}{(1-k)!(k-1)!} \left(1 + \frac{z}{k}\right)^k = 1 + z, \quad (1.3.58)$$

and if $M = 2$, then

$$\begin{aligned} P^{\text{EE}}(z) &= \sum_{k=1}^2 \frac{(-1)^{k+2} k}{(2-k)!(k-1)!} \left(1 + \frac{z}{k}\right)^k = -1(1+z) + 2\left(1 + \frac{z}{2}\right)^2, \\ &= 1 + z + \frac{z^2}{2}, \end{aligned} \quad (1.3.59)$$

where this pattern continues, such that the stability polynomial for the explicit-Euler extrapolation method is equal to an M^{th} -degree Taylor polynomial of the exponential function:

$$P^{\text{EE}}(z) = \sum_{k=1}^M \frac{(-1)^{k+M} k^{M-1}}{(M-k)!(k-1)!} \left(1 + \frac{z}{k}\right)^k = \sum_{k=0}^M \frac{z^k}{k!}. \quad (1.3.60)$$

Internal Stability Polynomial of the Explicit-Midpoint (Gragg) Extrapolation:

Similar to Richardson extrapolation, the stability polynomial for Gragg's method can also be shown to be equivalent to an M^{th} -degree Taylor polynomial of the exponential function [20], such that:

$$P^{\text{EM}}(z) = \sum_{k=1}^r \frac{2(-1)^r k^{2r}}{(r-k)!(r+k)!} \left(\mathbf{T}_{2k}\left(\frac{iz}{2k}\right) + i\mathbf{U}_{2k-1}\left(\frac{iz}{2k}\right) \right) = \sum_{k=0}^{2r} \frac{z^k}{k!}, \quad (1.3.61)$$

where $\mathbf{T}_n(\cdot)$ and $\mathbf{U}_n(\cdot)$ represent the n^{th} Chebyshev polynomials of the first and second kind, respectively [20]. For example, the first kind can be represented as functions of either

x or θ as follows:

$$\mathbf{T}_n(x) = \cos(n \cos^{-1}(x)), \quad (1.3.62a)$$

$$\mathbf{T}_n(\cos(\theta)) = \cos(n\theta), \quad (1.3.62b)$$

where $x = \cos(\theta)$ and $n \geq 0 \in \mathbb{Z}$. Likewise, the second kind are as follows:

$$\mathbf{U}_n(x) = \frac{\sin((n+1) \cos^{-1}(x))}{\sin(\cos^{-1}(x))}, \quad (1.3.63a)$$

$$\mathbf{U}_n(\cos(\theta)) = \frac{\sin((n+1)\theta)}{\sin(\theta)}. \quad (1.3.63b)$$

Using Equations (1.3.60) and (1.3.61) we can plot the stability regions for both extrapolation methods. However, since they both share the same stability polynomial, the only difference in Figure 1.3.6 is that Gragg's method is only valid for even orders of P . As previously mentioned in Section 1.3.1, both extrapolation methods are equivalent to an explicit Runge-Kutta (ERK) method when the P order is fixed. The stability regions for $1 \leq P \leq 8$ are shown in Figure 1.3.6.

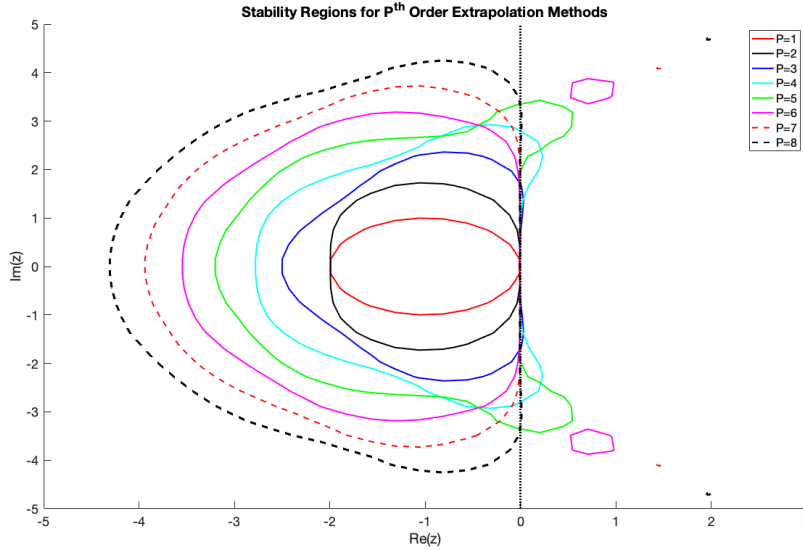


Figure 1.3.6. Stability Regions for P^{th} Order Richardson and Gragg Extrapolation Methods.

1.4 Multirate Extrapolation Literature Review

The study and prior development of multirate time-integration schemes is well studied and spans over 60 years, with the earliest numerical methods occurring from 1960 through the early 1990s, within the works of [21], [22], [23], [24], [25]. Throughout the 2000s, significant contributions to the development of various MR methods are presented in the studies of [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], with some of the most recent results found in [38], [39], [9], [40], [41], [42], [43], [44], [45], [46], [47] from early 2010 to the present. With respect to previous MR time-integration schemes based on Richardson extrapolation using explicit forward Euler, please see Engstler and Lubich [22]. Additionally, Constantinescu and Sandu developed extrapolated implicit-explicit (IMEX) time-stepping methods in 2010 within [9] and extrapolated MR explicit Euler and linearly implicit Euler methods in 2013 within [40].

Multirate Extrapolation Methods for Differential Equations with Different Time Scales:

The first multirate extrapolation scheme was developed by Engstler and Lubich [22], which they called multirate extrapolation (MURX), and is based on Richardson extrapolation using forward Euler as the base method. The MURX time integration scheme is used in conjunction with an adaptive/dynamic partitioning of the slow and fast components of their problems spatial grid, which are integrated simultaneously. At the start of each time step, all fast and slow spatial components are activated (i.e., solution values are computed for active components and not computed for inactive components). Then, at each time interval of the MURX method, active components transition from an active status to an inactive status within the extrapolation process, based on predetermined tolerances for the slow and fast components, respectively.

Specifically, their method inactivates the slow components at earlier extrapolation levels, with higher order extrapolation continuing to be performed on the faster components until all spatial components reach an inactive status. Let's consider some i^{th} component that is currently active. If the following extrapolation solutions for say T_{11}^i , T_{21}^i , and T_{22}^i have been computed for the i^{th} spatial component, then the following error estimation is calculated for component i as follows:

$$|T_{22}^i - T_{21}^i| < \text{tol}. \quad (1.4.1)$$

If the error estimation in Equation (1.4.1) is below the given tolerance, then solution T_{22}^i is

accepted, such that

$$T_{22}^i \approx q^i(t_0 + H), \quad (1.4.2)$$

and component i becomes inactive. However, if the error estimation is not less than the tolerance, then the i^{th} component remains active and the extrapolation process continues until the error estimation for

$$|T_{j,k+1}^i - T_{j,k}^i| < \text{tol}, \quad k + 1 \leq j, \quad (1.4.3)$$

is satisfied.

Once the extrapolation process is complete for all slow and fast components within the respective time step, the method reactivates all components and then restarts the extrapolation process using the next time interval. This process continues until the MURX method has reached final time. Additionally, because their spatial grid is constructed using adaptive mesh refinement (AMR), both the fast and slow components can transition from slow to fast and vice-a-versa within each respective time step. Furthermore, since MURX is used to solve problems with changing spatial elements, it is considered to fall within a subset of multirate methods known as component partitioning or geometric partitioning multirate methods, as opposed to other MR methods that are constructed by splitting the RHS of the problem into fast and slow processes. These RHS splitting schemes are sometimes referred to as split (explicit/IMEX) time-integration schemes, as well as process partition multirate methods; see [3] for a more complete review of geometric and process partitioned methods. As a final note, although the MURX method is considered multirate by the authors, each time step uses the same macro step-size for solving the fast and slow components. Although this time step is fixed, their method is multirate in the sense that the fast components are evaluated at every time step, while the slow components are not.

Extrapolated Implicit-Explicit (IMEX) Time Stepping:

We previously discussed the use of forward Euler and the midpoint rule as potential base methods for extrapolation; however, in [9], Constantinescu and Sandu developed a new family of extrapolated methods, which use three different IMEX base methods for extrapolation. The base methods we have reviewed so far are the forward Euler method:

$$q_{n+1} = q_n + h(S(q_n) + F(q_n)), \quad (1.4.4)$$

and Gragg's Method without smoothing, which uses the following midpoint rule:

$$q_1 = q_0 + h(S(q_0) + F(t_0, q_0)), \quad (1.4.5a)$$

$$q_{i+1} = q_{i-1} + 2h(S(t_i, q_i) + F(t_i, q_i)), \text{ for } i = 1, 2, \dots, n. \quad (1.4.5b)$$

Both Equations (1.4.4) and (1.4.5) are fully explicit base methods for extrapolation, where we have written them in terms of the following initial value problem, which involves both a slow (S) and fast (F) process, respectively:

$$q'(t) = f(t, q(t)), \quad f(t, q(t)) = S(t, q) + F(t, q), \quad q(t_0) = q_0, \quad t \geq t_0. \quad (1.4.6)$$

An issue with using Equations (1.4.4) and (1.4.5) to solve the IBVP in Equation (1.4.6) is that both the fast and slow processes are advanced using the same global step-size, h . Therefore, the overall stability of the method is restricted by the fast process, which dictates the maximum step-size that can be used. In [9], the extrapolation base methods are derived from the linearly implicit Euler method, also known as backward Euler, which is written as follows:

$$q_{n+1} = q_n + h(S(q_{n+1}) + F(q_{n+1})). \quad (1.4.7)$$

Since the slow and fast processes on the RHS of Equation (1.4.7) cannot be directly solved for using q_{n+1} , we instead use a first order Taylor series expansion approximation, such that Equation (1.4.7) becomes:

$$q_{n+1} = q_n + \left[\mathcal{I} - h(S + F)'(q_n) \right]^{-1} \left(hS(q_n) + hF(q_n) \right), \quad (1.4.8)$$

where $[I - h(S + F)'(q_n)]^{-1}$ contains the Jacobian matrix of the slow and fast processes. The specific details for approximating Equation (1.4.7) with Equation (1.4.8) can be found in Appendix C.

Constantinescu and Sandu [9], call their three base methods W-IMEX, pure-IMEX, and

split-IMEX, which are slight variations to Equation (1.4.8) and are defined as follows:

$$q_{n+1} = q_n + \left[\mathcal{I} - hF'(q_n) \right]^{-1} \left(hS(q_n) + hF(q_n) \right), \quad [\text{W-IMEX}] \quad (1.4.9a)$$

$$q_{n+1} = q_n + hS(q_n) + \left[\mathcal{I} - hF'(q_n) \right]^{-1} \left(hF(q_n) \right), \quad [\text{pure-IMEX}] \quad (1.4.9b)$$

$$q_{n+1} = q^* + \left[\mathcal{I} - hF'(q_n) \right]^{-1} \left(hF(q^*) \right); \quad q^* = q_n + hS(q_n). \quad [\text{split-IMEX}] \quad (1.4.9c)$$

In all three IMEX methods above, the fast (stiff) process is solved implicitly, while the slow (non-stiff) process is solved explicitly. They note that the W-IMEX method is essentially the same as the linearly implicit Euler method, except for the Jacobian, which is approximated by the Jacobian of the fast process only. They show that this is sufficient for stability and makes the overall method computationally more efficient. For the pure-IMEX and split-IMEX methods, the same Jacobian approximation is used; however, the slow and fast process are computed separately, “making them truly IMEX schemes” [9].

The advantage with IMEX methods is that the fast processes (i.e., stiff terms, such as acoustic or gravity waves) are solved implicitly, while the slow processes (i.e., non-stiff terms, such as advection waves) are solved explicitly. The combination of implicit-explicit schemes is that there is no restriction on the maximum step-size used for the fast terms; however, this benefit is not free and comes at the cost of solving a system of non-linear equations, which can be computationally expensive. The slow processes are integrated using the explicit method, where the maximum step-size used is no longer restricted by the fast process. Therefore, we can think of IMEX methods as process partition multirate methods.

Extrapolated Multirate Methods for Differential Equations with Multiple Time Scales:

In addition to their IMEX extrapolation methods developed in [9], Constantinescu and Sandu developed three new multirate base methods in [40] for use with Richardson extrapolation. Similar to the works by Engstler and Lubich [22], their three methods “extend [the MURX] strategy to extrapolated explicit and implicit compound multirate steps,” for solving problems where the solution vector can be partitioned into m -components, each with their own respective time-scale.

A simplified two-scale problem for $m = 2$ is:

$$\begin{cases} y'(t) = f(x, y(x), z(x)) \\ z'(t) = g(x, y(x), z(x)) \end{cases}, \quad \begin{bmatrix} y(x_0) & z(x_0) \end{bmatrix}^T = \begin{bmatrix} y_0 & z_0 \end{bmatrix}^T, \quad x > x_0, \quad (1.4.10)$$

where y and z represent the slow and fast components, respectively. The first of their three methods for solving Equation (1.4.10) is the m -rate multirate explicit Euler extrapolation (MREX) method, which is a fully explicit multirate extrapolation scheme that uses forward Euler as its base method. It is defined in [40] as follows:

$$\text{MREX: } \begin{cases} y_{n+1} = y_n + hf(y_n, z_n) \\ z_{n+\frac{i}{m}} = z_{n+\frac{i-1}{m}} + \frac{h}{m}g(Y_{n+\frac{i-1}{m}}, z_{n+\frac{i-1}{m}}) \end{cases}, \quad i = 1, \dots, m, \quad (1.4.11)$$

where $m \geq 1 \in \mathbb{Z}$. Notice that in Equation (1.4.11), that both the fast and slow solutions are advanced using the explicit forward Euler method; however, the slow components use a step-size of h , while the fast components require m -steps of step-size $\frac{h}{m}$.

Notice that the fast component, z , requires knowledge about the slow component, y , at times $t_{n+\frac{i-1}{m}}$ that must be approximated for since y is not computed at these intermediate time values. Therefore, the MREX method considers three possible options for $Y_{n+\frac{i-1}{m}}$, which represent the solution for y at $x_{n+\frac{i}{m}}$. They are the following:

$$Y_{n+\frac{i-1}{m}} = y_n, \quad (1.4.12a)$$

$$Y_{n+\frac{i-1}{m}} = y_{n+1}, \quad (1.4.12b)$$

$$Y_{n+\frac{i-1}{m}} = \left(\frac{m-i+1}{m}\right)y_n + \left(\frac{i-1}{m}\right)y_{n+1}. \quad (1.4.12c)$$

The other two methods found in [40] are both fully implicit multirate base methods for use with the Richardson extrapolation procedure. Similar to the MREX method, these two implicit methods are called MRLIM#1 and MRLIM#2, where MRLIM stands for an m -rate multirate linearly implicit method based off of the backward Euler method. Note, all three methods proposed in their paper fall within the geometric partition multirate methods.

Noticeable differences from the MREX, MRLIM#1 and MRLIM#2 methods from that of the MURX method proposed by Engstler and Lubich [22] is that all three methods are

“multirate” within the base method itself. This means that the base extrapolation method is constructed with both fast and slow components, as opposed to the MURX method, which uses a single-rate forward Euler method as its base method. Furthermore, the MURX method dynamically partitions the spatial domain into fast and slow components via AMR at each time-step, whereas the three methods from [40] perform geometric partitioning only once, at the very beginning.

Summary of Multirate Extrapolation Methods:

When reviewing the methods developed within [22], [9], [40], we notice that all previous multirate extrapolation schemes only consider using either the forward Euler method or the linearly implicit Euler method as base methods for the extrapolation process as defined by Section 1.3.1. We previously demonstrated within this chapter that the midpoint rule used within the Gragg extrapolation method is a more efficient choice for the base method, over that of the forward Euler method when constructing high order single-rate extrapolation. Therefore, we look to explore the potential efficiency and stability gains when using higher order base methods for constructing a multirate extrapolation scheme. To do this, we propose the construction of a multirate extrapolation time-integrator using the multirate infinitesimal step (MIS) method [48], [37], [36], [43] as the base method for extrapolation. Furthermore, we specifically focus our MIS-extrapolation method on process partitioned problems as opposed to the geometric partitioned problem. Additionally, the MIS method can be employed using both fully explicit as well as IMEX splittings of the RHS of our IBVP. We will review the MIS method in Chapter 2 and then show how it can be used as the base method for extrapolation in Chapter 3. As a final note, the MIS method can be efficiently solved using explicit Runge-Kutta (ERK) methods; therefore, we will first conduct a brief review of SR and MR RK methods.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:

Single-Rate and Multirate Runge-Kutta Methods

In Section 2.1 we provide a general overview of SR explicit RK multi-stage time-integration schemes used to approximate the continuous time derivatives of the PDE system from Chapter 1. Then, we introduce the multirate infinitesimal step (MIS) method in Section 2.3, which we will use as the base method for our process partition explicit multirate extrapolation time-integrator (i.e., MIS-extrapolation), shown in Section 3.1. This chapter will also show the numerical results for solving the non-linear shallow water equations using both the single-rate and multirate methods within Sections 2.2 - 2.4. The specific combinations of ERK and additive Runge-Kutta (ARK) butcher tableaux coefficients used within these methods can be found in Appendix A.

2.1 Single-Rate Explicit Runge-Kutta (ERK) and Additive RK (ARK) Methods

The simplest of all SR ERK methods is Euler's method and is the foundation for the development of more complex, higher-order, RK time-integration schemes. Let,

$$\frac{d}{dt}q(t) = R(t, q(t)), \quad (2.1.1)$$

be a method of lines (MOL) semidiscretization of a time dependent PDE, such that forward Euler's method approximates the numerical solution to Equation (2.1.1) from time level t^n to time level t^{n+1} using the following formula:

$$q^{n+1} = q^n + \Delta t R(t^n, q^n), \quad n = 0, 1, 2, \dots, \quad (2.1.2)$$

where

$$\frac{d}{dt}q(t^n) = R(t^n, q^n) \approx \frac{q^{n+1} - q^n}{\Delta t}, \quad \text{and} \quad \Delta t = t^{n+1} - t^n. \quad (2.1.3)$$

To start the sequence of approximations, an initial condition must be known. This condition is usually provided at $t = 0$, such that $q(t^0) = q^0$.

Using Equation (2.1.2), we can rewrite the left hand side (LHS) of Equation (1.2.21) for the 1D SWE as:

$$\frac{\mathbf{q}^{(e),n+1} - \mathbf{q}^{(e),n}}{\Delta t} = \mathbf{R}_S(\mathbf{q}^{(e),n}) + \mathbf{R}_F(\mathbf{q}^{(e),n}), \quad (2.1.4)$$

such that if $\mathbf{R}(\mathbf{q}^{(e),n}) = \mathbf{R}_S + \mathbf{R}_F$, then

$$\mathbf{q}^{(e),n+1} = \mathbf{q}^{(e),n} + \Delta t \mathbf{R}(\mathbf{q}^{(e),n}), \text{ for } \mathbf{q} = \begin{bmatrix} h_s \\ U \end{bmatrix}. \quad (2.1.5)$$

Notice that Equation (2.1.5) explicitly defines the solution for element (e) at time $n + 1$, only in terms of the prior solution at time n ; therefore, this scheme represents the explicit forward Euler method for the IBVP defined in Section 1.1.2 and will be defined as ERK1, since it is a first order, explicit RK method. Additionally, ERK1 is a SR method as Δt is strictly determined by a single scalar value, specifically chosen to ensure that the overall method remains stable throughout the numerical computation.

2.1.1 Generalized Single-Rate ERK Methods

The explicit forward Euler method is only first order accurate, such that improvements to this method are attributed to the works of Carl Runge and Wilhelm Kutta, known as Runge-Kutta (RK) methods. The forward in time generalized s -stage SR explicit RK method, as described in [11], are constructed using the following formula:

$$\begin{aligned} K_1 &= R(t^n, q^n), \\ K_i &= R\left(t^n + c_i \Delta t, q^n + \Delta t \sum_{j=1}^{i-1} a_{i,j} K_j\right), \quad i = 2, \dots, s \\ q^{n+1} &= q^n + \Delta t \sum_{i=1}^s b_i K_i, \end{aligned} \quad (2.1.6)$$

where the coefficients for Equation (2.1.6) can be represented using Butcher's partitioned tableau [49]:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^\top \end{array}, \quad \begin{array}{c|ccc} 0 & & & \\ c_2 & a_{2,1} & & \\ c_3 & a_{3,1} & a_{3,2} & \\ \vdots & \vdots & \vdots & \ddots \\ c_s & a_{s,1} & a_{s,2} & \dots & a_{s,s-1} \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array}. \quad (2.1.7)$$

The K_i terms in Equation (2.1.6) represent a single stage of the method, which is an evaluation of the RHS of the ODE being solved. Each ERK method can be represented using the tableau in Equation (2.1.7), where the $\mathbf{c}$, $\mathbf{A}$, and $\mathbf{b}^\top$ are defined as follows:

- $\mathbf{c} \in \mathbb{R}^s$ - Vector containing the time position for each stage evaluation, where all $c_i = \sum_{j=1}^{i-1} a_{i,j}$ for $i = 2, \dots, s$, and $c_1 = 0$.
- $\mathbf{A} \in \mathbb{R}^{s \times s}$ - Matrix where each row defines one stage of the s -stage method and the $a_{i,j}$ entries indicate the stages dependence on previous stages' derivatives. Note: $a_{1,1} = 0$.
- $\mathbf{b}^\top \in \mathbb{R}^s$ - Vector containing the final solutions dependence (weighted sum) on derivatives evaluated at previous stages.

For example, the forward Euler (ERK1) method in Equation (2.1.2) can be written in Butcher tableau form as

$$q^{n+1} = q^n + \Delta t R(t^n, q^n), \quad \begin{array}{c|c} 0 & \\ \hline & 1 \end{array}. \quad (2.1.8)$$

When using explicit RK methods, higher order accuracy can be achieved if more stage evaluations are computed. Two of the most common 2^{nd} order methods, as shown in [49], are based on the explicit trapezoidal rule (ERK21):

$$\begin{aligned} q^{n+1*} &= q^n + \Delta t R(t^n, q^n), \\ q^{n+1} &= q^n + \frac{\Delta t}{2} [R(t^n, q^n) + R(t^{n+1}, q^{n+1*})] \end{aligned} \quad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}, \quad (2.1.9)$$

and midpoint rule (ERK22):

$$\begin{aligned} q^{n+\frac{1}{2}} &= q^n + \frac{\Delta t}{2} R(t^n, q^n), \\ q^{n+1} &= q^n + \Delta t R(t^{n+\frac{1}{2}}, q^{n+\frac{1}{2}}) \end{aligned} \quad \begin{array}{c|c} 0 & \\ \frac{1}{2} & \frac{1}{2} \\ \hline & 0 \quad 1 \end{array} . \quad (2.1.10)$$

Additionally, the classical 4th order RK method (ERK4), is shown below using the generalized formula from Equation (2.1.6) on the left, and Butcher tableau on the right:

$$\begin{aligned} K_1 &= R(t^n, q^n), \\ K_2 &= R(t^n + \frac{\Delta t}{2}, q^n + \frac{\Delta t}{2} K_1), \\ K_3 &= R(t^n + \frac{\Delta t}{2}, q^n + \frac{\Delta t}{2} K_2), \\ K_4 &= R(t^n + \Delta t, q^n + \Delta t K_3), \\ q^{n+1} &= q^n + \frac{\Delta t}{6} (K_1 + 2K_2 + 2K_3 + K_4) \end{aligned} \quad \begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ \frac{1}{2} & 0 & \frac{1}{2} & \\ 1 & 0 & 0 & 1 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} . \quad (2.1.11)$$

A significant amount of research has been conducted on the construction, stability, accuracy and efficiency of ERK methods over the past few decades, with well-known resources including, but not limited to, [49] and [11]. Additionally, Butcher provides an early history of Runge-Kutta methods in [50], which highlights the works of Runge [51], Heun [52], and Kutta and Nyström [53].

Within this work, the ERK methods under consideration are of orders 1 through 5, where the number of stages of each ERK is equivalent to the order of the method, up to 4th order. The explicit, 5th order, single-rate methods considered are Butcher's ERK5 and Dormand-Prince's ODE45, which have 6 and 7 stages respectively. Additionally, we consider the use of a small subset of generalized additive ERK (ARK) methods developed by Giraldo and Constantinescu [54] and Kennedy and Carpenter [55]. The ARK methods under consideration are of order 1, 2, 3, 4, and 5, which contain 2, 3, 4, 6, and 8 stages,

respectively. For reference, we will use either ARK(order,stages) or ARK-order (# stage) as the notation to indicate the order of accuracy and number of stages for each single-rate ARK method.

For example, ARK(2,3) [54] is a 3-stage RK method that is third order accurate for linear problems, yet second order accurate for non-linear problems. Additionally, ARK(4,6) represents a 6-stage RK method that is 4th-order accurate for both linear and non-linear problems. The respective Butcher tableau coefficients for all s -stage SR ARK methods used to solve the IBVP from Chapter 1 can be found in Appendix A.

Furthermore, all ARK methods we consider are constructed using the time-integration scheme from [56]:

$$\begin{aligned} \mathbf{Q}_i &= \mathbf{q}^n + \Delta t \sum_{j=1}^{i-1} \alpha_{i,j} \mathbf{R}(\mathbf{Q}_j), \\ \mathbf{q}^{n+1} &= \mathbf{q}^n + \Delta t \sum_{i=1}^s \beta_i \mathbf{R}(\mathbf{Q}_i), \end{aligned} \tag{2.1.12}$$

which can be shown to be equivalent to the generalized ERK algorithm in Equation (2.1.6). In addition to the ERK and ARK methods highlighted within this section, other fully explicit time integration schemes include strong stability preserving (SSP) and low-storage Runge-Kutta (LSRK) methods, which exhibit useful properties in maintaining numerical stability and efficiency; however, we will not consider these methods within this work.

2.2 Single-Rate ERK and ARK Results

Let's now look at how the single-rate ERK and ARK methods from Appendix A perform using Equation (2.1.12) to approximate the 1D non-linear shallow water equations IBVP from Chapter 1. Figures 2.2.1a and 2.2.1b display the single-rate results using both the ARK and ERK butcher tableau coefficients, which can be found in Appendix A.

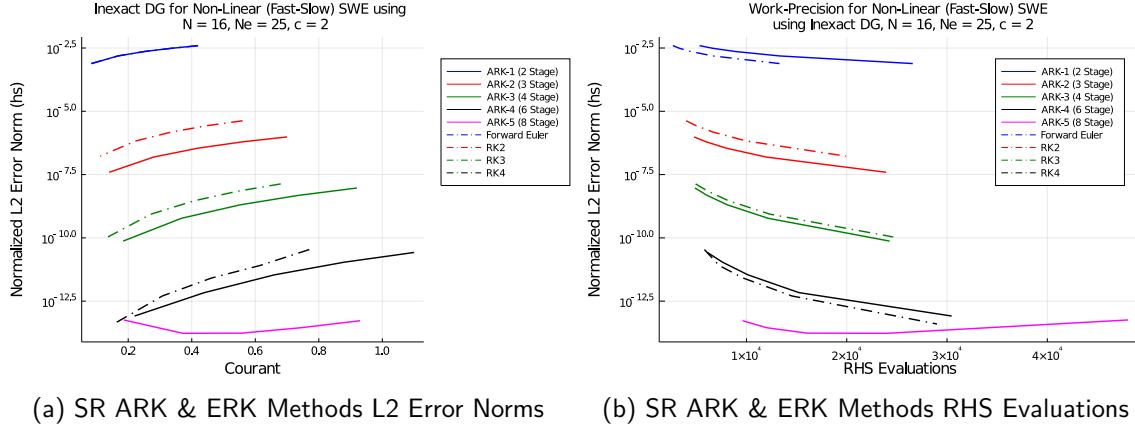


Figure 2.2.1. Error Plots and Total RHS Evaluations for Non-Linear SWE Using Single-Rate ARK & ERK Methods ($g = h_B = 1$).

The two subfigures within Figure 2.2.1 show the results comparing the L2 error norms against Courant value and total RHS evaluations required for each of these methods. The SR ARK methods allow the use of a slightly larger Courant compared to the SR ERK methods; however, because the ARK methods require additional stage evaluations, the use of a larger time-step size is offset by more RHS evaluations (i.e., higher computational cost), which can be seen in Figure 2.2.1b.

Notice that the computational cost difference between the ERK and ARK methods decreases when comparing the higher order methods; therefore, the use of a larger Courant does slightly compensate for the use of more stage evaluations. For example, the total number of RHS evaluations is approximately the same for both the ARK-4 (6 Stage) and RK4 methods, even though the ARK method requires 2 more stage evaluations per time-step. However, these results were computed assuming the gravity and bathymetry are both equal to 1; therefore, the fast process within the problem is not truly fast. Let's now look at these same SR results when increasing gravity to $g = 10$.

Figure 2.2.2 displays the results for all ARK and ERK methods with orders of accuracy less than 4, for $g = 10$, $h_B = 1$. Note, there was no benefit in computing 4th order and above time-integrators for the spatial parameters we used (i.e., Inexact DG, 16th order interpolating polynomial within 25 elements), as the spatial error begins to dominate the error norms.

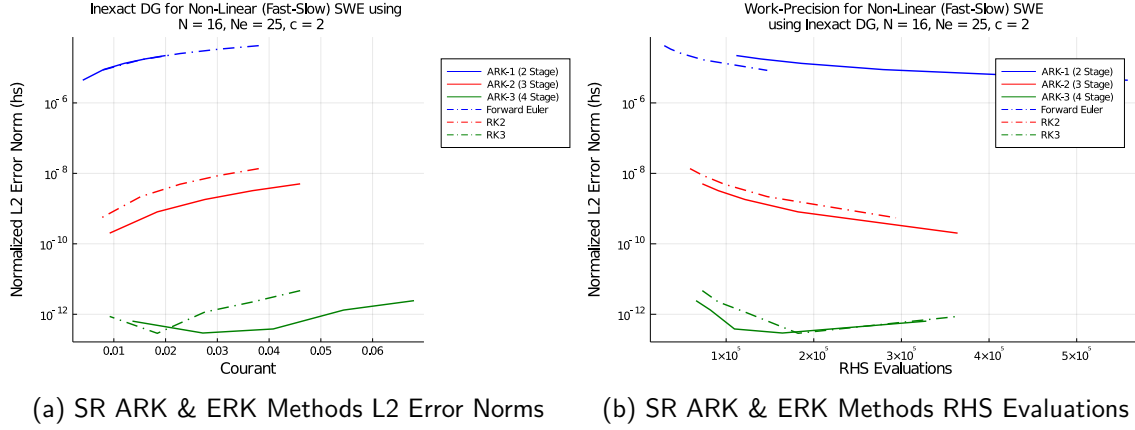


Figure 2.2.2. Error Plots and Total RHS Evaluations for Non-Linear SWE Using Single-Rate ARK & ERK Methods ($g = 10$, $h_B = 1$).

First, we looked at how each ERK method compared to its equivalent ARK method regarding the total number of RHS evaluations at the methods maximum stable Courant value. Since forward Euler requires one less stage evaluation from that of ARK-1 (2 Stage), we would naturally assume that this 2 to 1 ratio of evaluations should hold true, regardless of how much the fast process increases. This assumption was verified, such that regardless of the choice of g and h_B , our numerical results shown in both Figure 2.2.1b and Table 2.2.1 verify that ARK-1 (2 Stage) does require twice as many RHS computations. Furthermore, by increasing gravity to $g = 10$, the fast process requires a much smaller time-step size to be used in order to maintain stability of the overall method. However, this 2:1 ratio remained constant as both methods maximum Courant was also reduced by the same factor.

For a third data point, we held gravity to ten, and increased the bathymetry depth also to ten, such that $g = h_B = 10$, thereby increasing the overall fast process by a factor of one hundred, since $gh_B = 100$. We again verified that the ARK-1 (2 Stage) method still required twice as many RHS evaluations as that of the forward Euler method. Therefore, our numerical results show that regardless of how much the fast process increases, the relationship between the maximum allowable Courant value and total number of RHS evaluations for the two 1st order methods remains constant at a 100% increase for the ARK-1 (2 Stage) method. This analysis was done for all SR ARK and ERK methods up to third order, with results displayed in Table 2.2.1.

The ARK-2 (3 Stage) versus RK2 methods produced similar results, such that the 2nd order ARK method required an average of 21.3% more RHS evaluations over RK2 when increasing the fast process. This was not true of the 3rd order ARK vs. ERK comparison. Instead, we found that the ARK-3 (4 Stage) method was slightly more efficient than RK3. Although ARK-3 (4 Stage) requires more stage evaluations, the method can use a slightly larger time-step value, thereby offsetting the additional stage computations when compared to its RK3 equivalent. These results are shown in the last few columns of Table 2.2.1, where the ARK-3 (4 Stage) method was on average 6.68% more efficient.

Table 2.2.1. Percent Increase/Decrease of Total Number of RHS Evaluations for SR ERK vs. SR ARK Methods up to Order 3.

Single-rate ERK versus Single-rate ARK Methods of Orders 1, 2 and 3									
Gravity & Bathymetry Constants	Euler	ARK (1,2)	% $\uparrow\downarrow$	RK2	ARK (2,3)	% $\uparrow\downarrow$	RK3	ARK (3,4)	% $\uparrow\downarrow$
$g = 1, h_B = 1, gh_B = 1$	2661	5322	100.00	3990	4788	20.00	4932	4860	(1.46)
$g = 10, h_B = 1, gh_B = 10$	55830	111660	100.00	58770	72828	23.92	72828	65688	(9.80)
$g = 10, h_B = 10, gh_B = 100$	214728	429456	100.00	429456	515349	20.00	515349	470148	(8.77)

The results from Table 2.2.1 and Figures 2.2.1 and 2.2.2 show that the first and second order SR ERK methods are more efficient than the equivalent SR ARK methods. This analysis is helpful as we want to construct our multirate method using an efficient base method, where we achieve the higher order accuracy via the extrapolation process. Because we are interested in developing a multirate extrapolation scheme, let's now define our multirate ERK and ARK base methods.

2.3 Multirate ERK and ARK Methods

In this section we will briefly discuss what is meant by multirate and then provide a basic overview of the Multirate Infinitesimal Step (MIS) method we have selected to use as our extrapolated base method. We previously mentioned two different types of multirate methods in Chapter 1, which we referred to as process partition and geometric partition. The development of geometric partition multirate methods came about based on the necessity to use multiple time-scales (time-steps) for advancing the numerical solution when the spatial geometry of the problem consists of varying sizes of elements. This non-constant geometry could be either fixed or adaptive, where the smallest elements within the spatial grid dictate

the maximum step-size (i.e., largest Courant number for hyperbolic problems) that a single-rate method must use in order to maintain the numerical stability of the problem. Therefore, geometric multirate methods are constructed so that different time-steps can be used, such that each time-step is congruent to the size of the various spatial elements. A few examples of different geometric partition multirate methods can be found within the works of [22], [27], [57], [40], [46], [41], and [38]. Note, multirate time-integration methods do not have to be constructed using Runge-Kutta methods.

The second type of MR time-integration schemes are process partition multirate methods, which are specifically developed for solving PDE problems where the right hand side (RHS) of the equation can be split into multiple processes, as was shown in Equations (1.0.1) and (1.0.2). We restate these equations as:

$$\frac{\partial q}{\partial t} = P_1(q) + P_2(q) + \cdots + P_N(q), \text{ and } q(t_0) = q_0, \quad (2.3.1)$$

such that if $N = 2$,

$$\frac{\partial q}{\partial t} = S(q) + F(q), \quad (2.3.2)$$

where $S(q)$ and $F(q)$ represent the slow and fast processes, respectively. We have already shown how the non-linear shallow water equations can represent this type of problem, where the slow and fast processes represent the advection and gravity waves, respectively. Some examples of different process partition multirate methods can be found within the works of [58], [48], [29], [35], [36], [37], [44], and [43]. Let's now look at the multirate infinitesimal step method, which was specifically developed for process partitioned problems.

2.3.1 Multirate Infinitesimal Step (MIS) Method

The basic idea behind the MIS method, is that we can use one ERK scheme to numerically approximate the slow process (referred to as the outer method) and another ERK scheme to approximate the fast process (referred to as the inner method). The slow and fast processes are then advanced in the following manner. The fast process is advanced between each stage of the outer ERK method using the full inner ERK method. During the inner update, the outer stage residual is linearly interpolated and blended over the inner process update

so that by the end of the inner step, the full outer residual has been applied. In other words, within a single stage of the outer ERK method's advance of the slow process, all stages of the inner ERK method are used to advance the fast process. Note, both the outer and inner ERK methods can use the same Butcher tableaux.

The MIS method [48], [36], [37] is shown in its present form within [43] for a two-partition (i.e., two process) problem as follows:

$$Q_1 = q_0, \quad (2.3.3a)$$

$$r_i = \sum_{j=1}^{i-1} \tilde{a}_{i,j}^O S(Q_j), \quad (2.3.3b)$$

$$V_{i,1} = Q_{i-1}, \quad (2.3.3c)$$

$$V_{i,j} = V_{i,j-1} + \Delta t \tilde{c}_i^O \sum_{k=1}^{j-1} \tilde{a}_{j,k}^I \left(\frac{1}{\tilde{c}_i^O} r_i + F(V_{i,k}) \right), \quad \begin{cases} i = 2, \dots, s^O + 1 \\ j = 2, \dots, s^I + 1 \end{cases}, \quad (2.3.3d)$$

$$Q_i = V_{i,s^I+1}, \quad (2.3.3e)$$

where we have written the method to be consistent with solving Equation (2.3.2). Within Equation (2.3.3), the parameters s^O and s^I represent the number of outer and inner stage values, respectively. The tilde parameters (i.e., $\tilde{a}_{ij}, \tilde{c}_i$) denote the incremental RK stage parameters used for both the inner and outer methods, such that:

$$\tilde{a}_{i,j} = \begin{cases} a_{i,j} - a_{i-1,j} & \text{if } i < s + 1 \\ b_j - a_{s,j} & \text{if } i = s + 1 \end{cases}, \quad (2.3.4)$$

and

$$\tilde{c}_i = \begin{cases} c_i - c_{i-1} & \text{if } i < s + 1 \\ 1 - c_s & \text{if } i = s + 1 \end{cases}, \quad (2.3.5)$$

where we use the standard Butcher tableau coefficients ($\mathbf{A} = a_{ij}, \mathbf{b}, \mathbf{c}$) to compute the tilde coefficients. Additionally, the r_i term within the MIS method represents the RHS computation related to the slow process. Notice that Equation (2.3.3) requires two loops (outer and inner) for advancing the solution in time, such that a more precise representation of the algorithm is provided by [3] in Algorithm 2.3.1:

Algorithm 2.3.1 2-Partition General Explicit MIS Method

```

 $Q_1 = q_n$ 
for  $i = 2 : I + 1$  do                                     ◇ Loop through slow process
     $r_i = \sum_{j=1}^{i-1} \tilde{a}_{i,j}^O S(Q_j)$ 
     $V_{i,1} = Q_{i-1}$ 
    for  $j = 2 : J + 1$  do                                     ◇ Loop through fast process
         $V_{i,j} = V_{i,j-1} + \Delta t \sum_{k=1}^{j-1} \tilde{a}_{j,k}^I [r_i + \tilde{c}_i^O F(V_{i,k})]$ 
    end for
     $Q_i = V_{i,J+1}$ 
end for
 $q_{n+1} = Q_{I+1}$ 

```

Although Algorithm 2.3.1 is sufficient for solving autonomous problems, we include Algorithm 2.3.2 when solving non-autonomous problems.

Algorithm 2.3.2 2-Partition General Explicit MIS Method (Non-Autonomous)

```

 $Q_1 = q_n$ 
for  $i = 2 : I + 1$  do                                     ◇ Loop through slow process
     $r_i = 0$ 
    for  $j = 1 : i - 1$  do
         $t_j = t_n + \Delta t c_j^O$                                      ◇ Time update for outer loop
         $r_i = r_i + \tilde{a}_{i,j}^O S(Q_j, t_j)$                              ◇ Build  $r_i$  (slow solution)
    end for
     $V_{i,1} = Q_{i-1}$ 
    for  $j = 2 : J + 1$  do                                     ◇ Loop through fast process
         $R_{sum} = 0$ 
        for  $k = 1 : j - 1$  do
             $t_k = t_n + \Delta t (c_{i-1}^O + \tilde{c}_i^O * c_k^I)$              ◇ Time update for inner loop
             $R_{sum} = R_{sum} + \tilde{a}_{j,k}^I [r_i + \tilde{c}_i^O F(V_{i,k}, t_k)]$    ◇ Build  $R_{sum}$  (slow + fast solution)
        end for
         $V_{i,j} = V_{i,j-1} + \Delta t R_{sum}$ 
    end for
     $Q_i = V_{i,J+1}$ 
end for
 $q_{n+1} = Q_{I+1}$ 
 $t_{n+1} = t_n + \Delta t$ 

```

When using the MIS method with explicit RK schemes, we use the notation, MIS ERK(I, J), where I and J represent the total number of RK stages within each of the ERK methods used within the outer and inner processes, respectively. For example, the MIS ERK(1,2) method uses a single stage, first order ERK method for the outer process and a two stage, second order, ERK method for the inner process. Note, for the ERK methods we compare within Appendix A, the total number of ERK stages is equivalent to the order of accuracy up to 4th order for these methods. This does not hold for higher order ERK and/or ARK methods.

A graphical depiction of the MIS ERK method in Algorithm 2.3.2 is shown in Figure 2.3.1, where we have selected to use forward Euler to solve the slow process, and various ERK methods of orders one through four to solve the fast process. Specifically, Figure 2.3.1 highlights where the various inner and outer ERK stage evaluations are computed along the time axis for both the slow and fast processes, respectively.

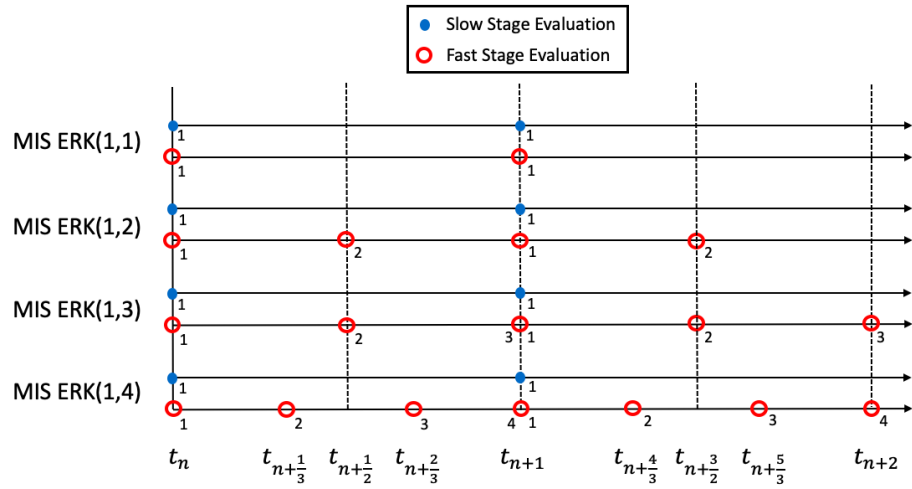


Figure 2.3.1. Multirate Infinitesimal Step Stage Evaluation Diagram for MIS ERK(1, 1) Through MIS ERK(1, 4).

Notice that the fast stage evaluation marker (red open circle) located at t_{n+1} for the MIS ERK(1,3) and (1,4) methods represents two separate fast stage evaluations. For example, the fast stage evaluation for the MIS ERK(1,3) method at t_{n+1} represents the third stage evaluation of the ERK3 method for time interval (t_n, t_{n+1}) and the first stage evaluation for time interval (t_{n+1}, t_{n+2}) . All stage evaluations are numbered underneath the markers.

Before we compare the numerical results using the MIS method, let us also briefly highlight the CFL condition for our two-process PDE system, since we split the RHS of our IBVP into its respective slow and fast processes, as shown by Equation (2.3.2). For the single-rate problem, both processes were advanced using the same rate; therefore, our previous CFL condition in Equation (1.3.49) for the 1D non-linear SWE was:

$$\sigma_{max} = (|u| + \sqrt{gh}) \frac{\Delta t_{max}}{\Delta x_{min}}. \quad (2.3.6)$$

Now, the CFL conditions for our two-process IBVP are:

$$\sigma_{slow} = |u| \frac{\Delta t_s}{\Delta x_{min}}, \quad (2.3.7a)$$

$$\sigma_{fast} = \sqrt{gh} \frac{\Delta t_f}{\Delta x_{min}}, \quad (2.3.7b)$$

where Δt_s and Δt_f represent the slow and fast rates (i.e., time-steps) that our multirate method will use to advance the slow and fast processes, respectively. Note that $\sigma_{slow} = \sigma_{fast} = \sigma_{max}$, such that:

$$\Delta x_{min} \sigma_{max} = |u| \Delta t_s = \sqrt{gh} \Delta t_f. \quad (2.3.8)$$

Therefore, a relationship exists between the fast, $\sqrt{gh}$, and slow, $|u|$, wave speeds, and their respective time-steps, where:

$$\frac{\Delta t_s}{\Delta t_f} = \frac{\sqrt{gh}}{|u|}. \quad (2.3.9)$$

This condition helps ensure the overall stability of the numerical method.

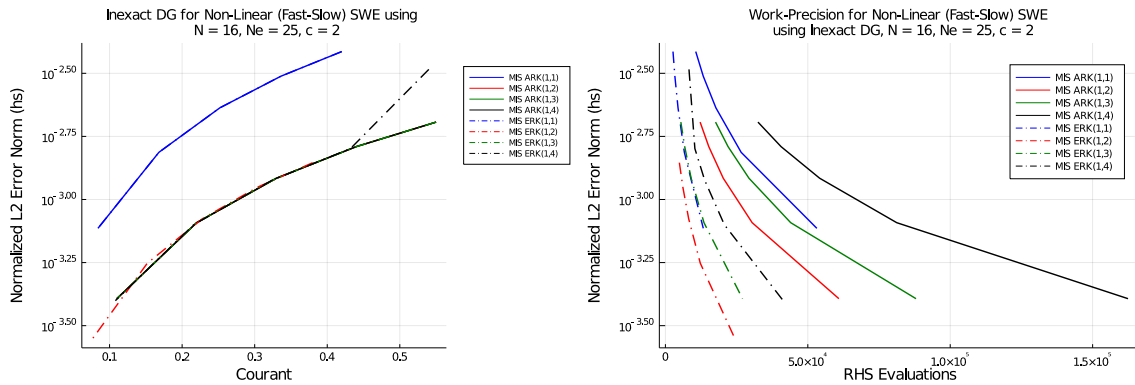
2.4 Multirate ERK and ARK Results

When comparing the MIS ARK(I, J) and MIS ERK(I, J) against each other, we previously defined (I, J) to represent the total number of RK stages within the outer and inner ERK and ARK methods, respectively. The purpose for correctly defining (I, J) is to ensure consistency between Algorithms 2.3.1 and 2.3.2, and the naming convention we use for these methods. Because we only consider ERK methods up to 4th order, this naming convention is not an issue for the MIS ERK(I, J) method; however, the total number of

stages within the ARK methods differs from the order of accuracy of the methods.

For example, MIS ARK(2,6) uses a two stage, first order method to advance the outer process, and a six stage, fourth order method to advance the inner process. Therefore, the MIS ERK(1,4) and MIS ARK(2,6) methods use equivalent order of accuracy RK methods for both the slow and fast processes. However, this can be confusing when comparing both types of methods within the respective plots; therefore, we will use the same (I, J) numbering within the MIS ERK method, to label its equivalent MIS ARK method, such that MIS ARK(I, J) will instead refer to the order of accuracy of the ARK methods in lieu of the total number of stages. For example, MIS ARK(2,6) will be written as MIS ARK(1,4) for easier comparison to the MIS ERK(1,4) method.

Although Algorithm 2.3.1 is sufficient for solving the non-linear shallow water equations, we do not have an exact solution; therefore, we construct all MR methods using Algorithm 2.3.2, since our manufactured solution developed in Appendix B requires a time-dependent source function. Since we now have choices on the combinations of outer and inner methods to use, our first MIS ARK versus MIS ERK comparison will use first order accuracy on the slow process (outer-loop), while varying the orders of accuracy of the fast process (inner-loop). These results are shown in Figures 2.4.1a and 2.4.1b.



(a) MIS ARK & ERK Methods L2 Error Norms (b) MIS ARK & ERK Methods RHS Evaluations

Figure 2.4.1. Error Plots and Total RHS Evaluations for Non-Linear SWE Using 1st Order MIS ARK & ERK Methods ($g = h_B = 1$).

Notice in Figure 2.4.1a that both the MIS ARK(1,1) and MIS ERK(1,1) methods are

equivalent to the first order, single-rate ARK and ERK methods, since both of these multirate methods advance the slow and fast processes using forward Euler. Additionally, Figure 2.4.1b highlights the difference in total RHS computations for all methods, where it is easy to see that all of the MIS ERK methods are more efficient than the MIS ARK methods of equivalent J -order. Furthermore, both figures indicate that regardless of the order of accuracy used to solve the fast process, the overall order of accuracy of the entire method is determined by $\min(I, J)$, such that all methods shown above are 1^{st} order accurate.

Although comparing the total number of RHS evaluations is important, we are primarily interested in the number of slow right hand side computations. For example, the problem we are most interested in solving is:

- An IBVP containing multiple (N) processes. For simplicity, we consider $N = 2$.
- The fast process is much faster than the slow process.
- Slow process is computationally expensive to evaluate.
- Fast process is relatively easy to compute, yet we desire maximum accuracy.

The third and fourth requirements must be balanced, such that we want to compute the slow process as few times as necessary, while maintaining as much accuracy as possible for the fast process. Therefore, instead of displaying the total number of RHS evaluations, Figure 2.4.2 only shows the total number of “slow” RHS evaluations computed for each method, as this is the number we want to minimize.

Once again, all of the MIS ERK methods are more efficient than the MIS ARK methods due to the additional stage evaluations each of the ARK methods require. Note, there was no difference in the plots for MIS ARK(1,2) through MIS ARK(1,4), with all three plots on top of each other. This implies that for the selected parameters (g, h_B, c), etc., if using ARK-1 (2-stage) for the slow process, there is no advantage in using an ARK method of J -order > 2 for the fast process.

When comparing each of the MIS ERK methods with each other, there is a slight efficiency gain for $J = 3$ or $J = 4$ over the single-rate method (i.e., MIS ERK(1,1)); however, it is important to note that these results are for $gh_B = 1$. Since the fast process is not significantly faster than the slow process, let’s now look at these same methods using $g = 10, h_B = 1$, where $gh_B = 10$. Figures 2.4.3a and 2.4.3b display these results, where by simply increasing

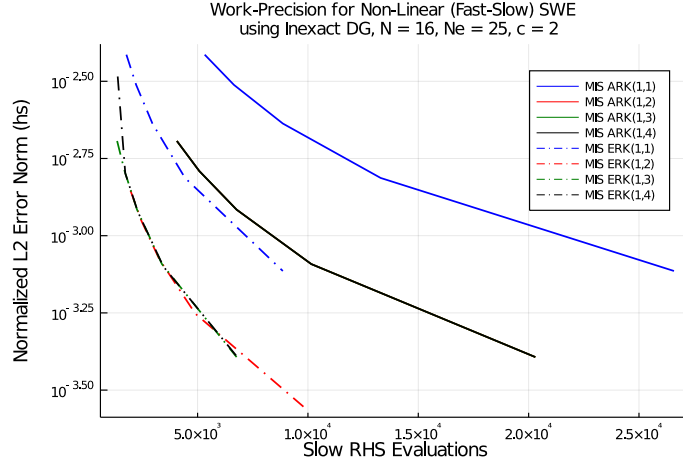
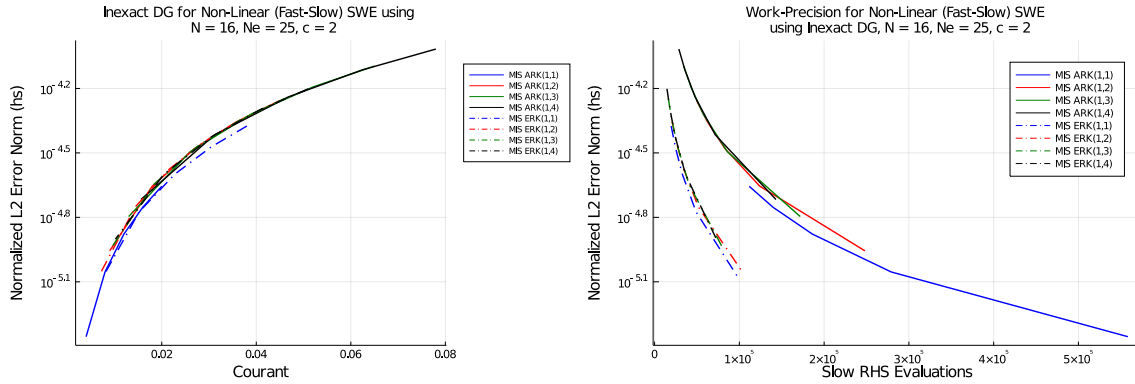


Figure 2.4.2. Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1st Order MIS ARK & ERK Methods ($g = h_B = 1$).

the fast waves slightly, we now notice an even greater efficiency advantage when using ERK within the MIS method versus the MIS ARK methods.



(a) MIS ARK & ERK Methods L2 Error Norms (b) MIS ARK & ERK Methods Slow RHS Evals

Figure 2.4.3. Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1st Order MIS ARK & ERK Methods ($g = 10$, $h_B = 1$).

We should note that if the MIS ARK methods are used, the increased gravity clearly demonstrates that the number of slow RHS evaluations decreases as the J -order increases. This was not previously observed when $g = h_B = 1$ for $J > 2$. Although the MIS ARK methods show improvement with increased gravity, the use of ERK methods within the MIS algorithm remain the most cost efficient in terms of the number of slow RHS evaluations.

If we zoom into the MIS ERK results within Figure 2.4.3b, we get a better view of how they compare against each other for largest Courant values.

Similar to the results in Figure 2.4.2, the MIS ERK results in Figure 2.4.4 also show that for $J > 2$, the MIS method uses significantly fewer slow RHS evaluations compared to the MIS ERK(1,1), which is the same as the single-rate forward Euler method. In addition to the above plots, the percent increase/decrease of the MIS method compared to the single-rate method of equivalent order can be seen in Table 2.4.1.

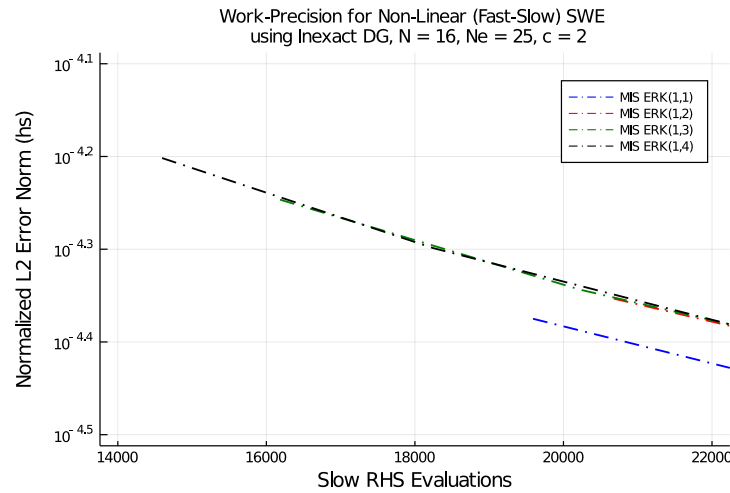


Figure 2.4.4. Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1st Order MIS ERK Methods ($g = 10$, $h_B = 1$).

From Table 2.4.1 we find that the MIS ERK(1,1) method is indeed equivalent to the SR forward Euler method, such that both methods required the exact number of slow RHS computations. Furthermore, it is easy to see that for $J \geq 3$, the MIS ERK(1, J) method is computationally more efficient based on the total number of slow RHS computations required for each method. For example, the MIS ERK(1,3) and MIS ERK(1,4) methods require approximately 20.85% and 25.16% less RHS evaluations, respectively, compared to the SR forward Euler method. These results highlight that the MIS ERK method can outperform a single-rate ERK method of equivalent order, using a sufficiently “high-enough” order of accuracy method to resolve the fast process, while at the same time minimizing the amount of work required to compute the slow process.

Table 2.4.1. Percent Increase/Decrease of Slow RHS Evaluations for MIS ERK Methods of Order 1 vs. SR ERK (Forward Euler) Method.

Forward Euler versus MIS ERK(1, J) Methods for $1 \leq J \leq 4$									
Gravity & Bathymetry Constants	Euler	MIS ERK(1,1)	% $\uparrow\downarrow$	MIS ERK(1,2)	% $\uparrow\downarrow$	MIS ERK(1,3)	% $\uparrow\downarrow$	MIS ERK(1,4)	% $\uparrow\downarrow$
$g = 1, h_B = 1, gh_B = 1$	1774	1774	0.00	1960	10.48	1354	(23.68)	1380	(22.21)
$g = 10, h_B = 1, gh_B = 10$	19590	19590	0.00	20678	5.55	16010	(18.27)	14596	(25.49)
$g = 10, h_B = 10, gh_B = 100$	143152	143152	0.00	143152	0.00	113648	(20.61)	103388	(27.78)

As previously mentioned, all of the MIS ERK methods within Figures 2.4.1 - 2.4.4 and Table 2.4.1 of Section 2.4 have been 1st order methods; therefore, let's look at one final example, such that we will now use a 2nd order ERK method within the outer loop (slow process) of the MIS algorithm. The inner loop (fast process) will be varied for $2 \leq J \leq 4$. Results are shown for $g = h_B = 1$ in Figures 2.4.5a and 2.4.5b, where we notice that both the single-rate ARK-2 (3 stage) and SR RK2 methods perform better than their respective multi-rate methods in terms of computational efficiency.

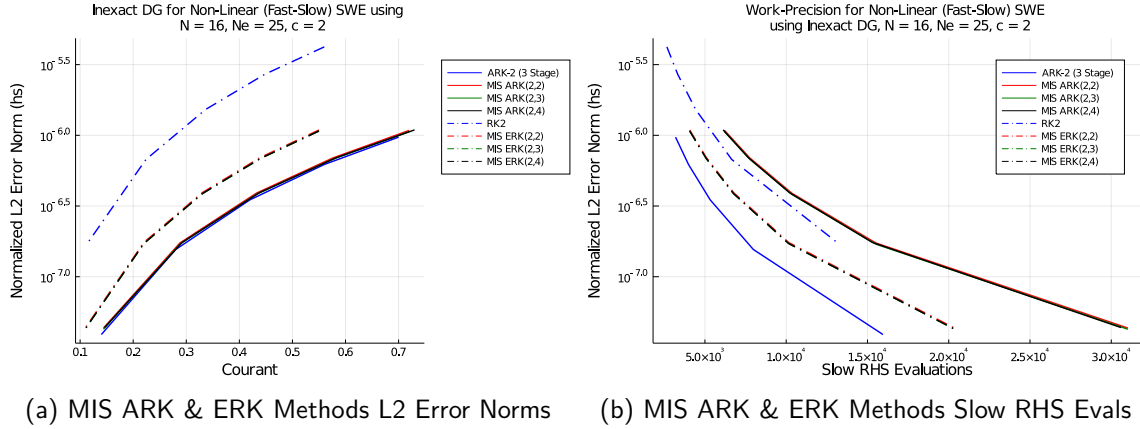


Figure 2.4.5. Error Plots and Slow RHS Evaluations for Non-Linear SWE Using 2nd Order MIS ARK & ERK Methods ($g = h_B = 1$).

Furthermore, we notice that there is no significant difference between using $2 \leq J \leq 4$ for either the ARK or ERK methods; therefore, it is computationally costly to use $J > 2$ for $gh_B = 1$. The MIS ERK(2, J) methods do have a slight advantage in overall accuracy compared to RK2; however, let's see if the multirate methods performance improves by

increasing the fast process. Figures 2.4.6a - 2.4.6b show the results for the same 2^{nd} order methods where we have increased gravity to $g = 10$.

We now see separation between the various MIS ARK(2, J) and MIS ERK(2, J) methods, where using a higher order method to resolve the fast process improves the computational efficiency based on the number of slow RHS evaluations required. We also find that the single-rate ARK-2 (3 stage) method no longer holds an accuracy or computational efficiency advantage over the MIS ERK methods, specifically compared to the MIS ERK(2,2) method. Furthermore, these results highlight the benefit in using the ERK methods over the ARK methods, within the MIS algorithm.

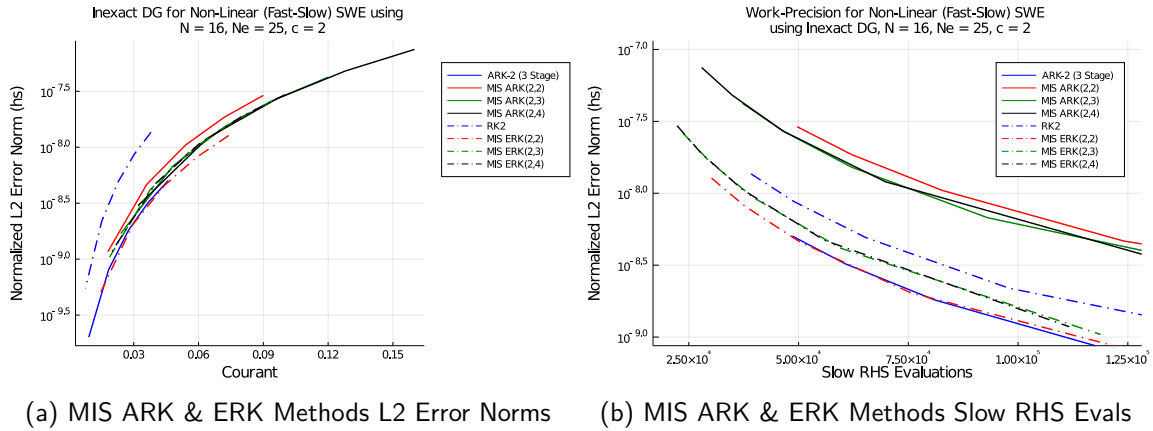


Figure 2.4.6. Error Plots and Slow RHS Evaluations for Non-Linear SWE Using 2^{nd} Order MIS ARK & ERK Methods ($g = 10$, $h_B = 1$).

Finally, Table 2.4.2 shows the percent increase/decrease comparison for both 2^{nd} order single-rate methods (RK2 and ARK-2 (3 stage)) versus MIS ERK(2, J) methods using $gh_B = 1, 10$ and 100 .

Table 2.4.2. Percent Increase/Decrease of Slow RHS Evaluations for MIS ERK Methods of Order 2 vs. SR RK2 and ARK-2 (3 stage) Methods.

RK2 versus MIS ERK(2, J) Methods							
Gravity & Bathymetry Constants	RK2	MIS ERK(2,2)	% $\uparrow\downarrow$	MIS ERK(2,3)	% $\uparrow\downarrow$	MIS ERK(2,4)	% $\uparrow\downarrow$
$g = 1, h_B = 1, gh_B = 1$	2660	4062	52.71	4062	52.71	4062	52.71
$g = 10, h_B = 1, gh_B = 10$	39180	30180	(22.97)	23760	(39.35)	22332	(42.99)
$g = 10, h_B = 10, gh_B = 100$	286304	223320	(21.99)	171786	(39.99)	159516	(44.28)

ARK-2 (3 stage) versus MIS ERK(2, J) Methods							
Gravity & Bathymetry Constants	ARK (2,3)	MIS ERK(2,2)	% $\uparrow\downarrow$	MIS ERK(2,3)	% $\uparrow\downarrow$	MIS ERK(2,4)	% $\uparrow\downarrow$
$g = 1, h_B = 1, gh_B = 1$	3192	4062	27.26	4062	27.26	4062	27.26
$g = 10, h_B = 1, gh_B = 10$	48552	30180	(37.84)	23760	(51.06)	22332	(53.99)
$g = 10, h_B = 10, gh_B = 100$	343566	223320	(34.99)	171786	(49.99)	159516	(53.57)

Both single-rate methods (i.e., ARK-2 and RK2) are clearly a better option when $gh_B = 1$, as shown in Figure 2.4.5; however, there is a drastic improvement for the MIS ERK(2, J) methods when the fast process is increased to $gh_B = 10$ and 100. For example the MIS ERK(2,2) method uses approximately 22.48% and 36.42% fewer slow RHS evaluations compared to the RK2 and ARK-2 (3 stage) methods, respectively. Furthermore, as the J -order increases, the MIS ERK(2,4) method uses approximately 43.64% and 53.78% fewer slow process RHS evaluations, respectively. Figures 2.4.7a and 2.4.7b display the graphical results for $gh_B = 100$ found within Table 2.4.2.

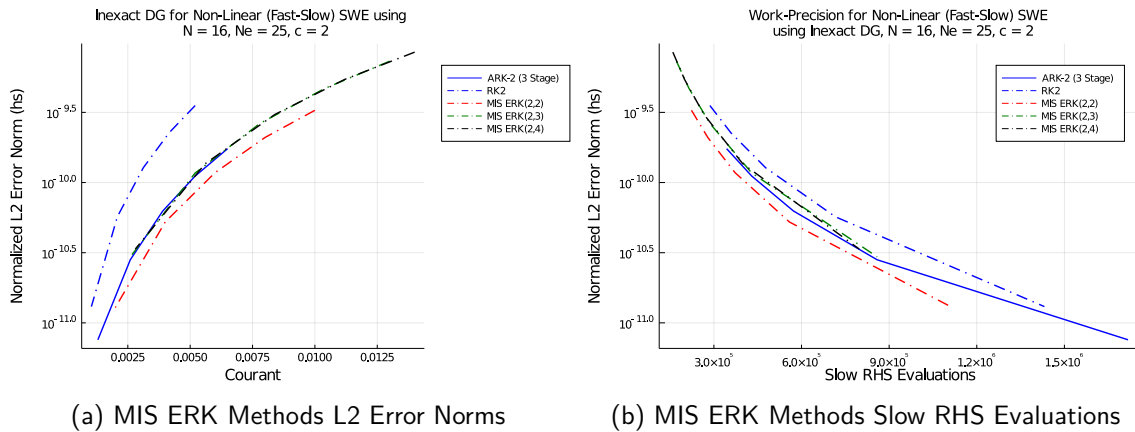


Figure 2.4.7. Error Plots and Slow RHS Evaluations for Non-Linear SWE using 2nd Order MIS ERK Methods ($g = 10, h_B = 10$).

In summary, we first demonstrated that the SR ERK methods are more cost efficient than the SR ARK methods when used either independently of, or in conjunction with, the MIS method. Next we showed that when the ratio between the fast and slow processes is large enough (i.e., 10:1 or 100:1), the multirate infinitesimal step method using explicit Runge-Kutta methods is more efficient than an explicit single-rate RK method of equivalent order of accuracy, in terms of the number of slow RHS evaluations. However, our numerical results showed that the MIS ERK(I, J) method's order of accuracy could only be at best 3^{rd} order, which is supported by [43]. However, to achieve 3^{rd} order accuracy, certain order conditions must be met. We will briefly highlight these conditions in the next chapter. We now look to combine what we have learned about the MIS ERK method with the extrapolation process, in order to achieve a higher order method using a lower order “multirate” base method.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:

Extrapolated Multirate Methods

In this chapter we develop our explicit extrapolation multirate method using the MIS ERK algorithm from Chapter 2, combined with the extrapolation process shown in Chapter 1. We also look at further improvements to the MIS ERK method, such as substepping of the fast process within the inner loop of the MIS ERK algorithm, which has been shown by [3] to improve the overall efficiency of the MIS ERK method. Finally, we compare all results of various explicit single-rate RK methods, to those of the MIS ERK method, when both substepping and extrapolation are applied. To help compare the computational cost between the single-rate and multirate methods, we introduce an efficiency metric for determining what the slow to fast computational ratio should be for a given multirate method to outperform another method of equivalent order of accuracy.

3.1 Multirate Infinitesimal Step (MIS) ERK Method with Extrapolation

In Chapter 1 we discussed the use of extrapolation methods, and how the process can be used to achieve higher order accuracy using a lower order base method. We will now apply the same process of extrapolation using the MIS ERK(I, J) method as the “base” method for extrapolation, thereby constructing a new multirate extrapolation method. Our MIS ERK extrapolation method is found in Algorithm 3.1.1.

Notice that we have simply wrapped the extrapolation process around the MIS ERK method, using the harmonic step-sequence. Therefore, we can now compute higher order approximations for a given MIS ERK(I, J) method, where the variable p is now included within the notation, such that MIS ERK(I, J, p) refers to a p^{th} order solution using the multirate MIS base method with explicit RK schemes in the outer and inner loops. Let’s compare the results for this multirate extrapolation method.

Algorithm 3.1.1 2-Partition General Explicit MIS Method with Extrapolation	
for $e = 1 : p$ do	◊ Loop constructs p approximations using harmonic step-sequence
$t_0 = t_n$	
$Y_{e,1} = q_n$	
for $d = 2 : e + 1$ do	◊ Loop builds temporary solution using $\frac{\Delta t}{e}$
$q_1 = Y_{e,d-1}$	
$Q_1 = q_1$	
for $i = 2 : I + 1$ do	◊ Loop through slow process
$r_i = \sum_{j=1}^{i-1} \tilde{a}_{i,j}^O S(Q_j)$	
$V_{i,1} = Q_{i-1}$	
for $j = 2 : J + 1$ do	◊ Loop through fast process
$V_{i,j} = V_{i,j-1} + \frac{\Delta t}{e} \sum_{k=1}^{j-1} \tilde{a}_{j,k}^I [r_i + \tilde{c}_i^O F(V_{i,k})]$	
end for	
$Q_i = V_{i,J+1}$	
end for	
$Y_{e,d} = Q_{I+1}$	◊ Store temporary solution of step-sequence
$t_0 = t_0 + \frac{\Delta t}{e}$	
end for	
$T_{e,1} = Y_{e,e+1}$	◊ Store final solution of step-sequence for $\frac{\Delta t}{e}$
end for	
 for $k = 2 : p$ do	
for $j = k : p$ do	
$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\frac{j}{j-k+1} - 1}$	◊ Perform extrapolation using Aitken-Neville formula to order k
end for	
end for	
$q_{n+1} = T_{p,p}$	
$t_{n+1} = t_n + \Delta t$	◊ Update final solution using p^{th} order approximation

**Note, for non-autonomous problems, refer to Algorithm 2.3.2 for incorporating the slow and fast time computations when constructing the r_i and $V_{i,j}$ solutions, such that $\frac{\Delta t}{e}$ is used in place of Δt .

3.2 MIS ERK Method with Extrapolation Results

Using Algorithm 3.1.1, we only consider 1^{st} or 2^{nd} order multirate base methods, as the extrapolation process is then used to generate the higher order approximation, in lieu of requiring a higher order base method. Therefore, the slow process within our IBVP will be advanced using either forward Euler or an explicit 2^{nd} order method. Choices for various 2^{nd} order methods may be explored; however, we will initially compute results using the ERK2 method from Table A.0.2 in Appendix A as our explicit 2^{nd} order method of choice.

The first set of multirate extrapolation results use the forward Euler method in both the outer and inner loops with a fixed order of extrapolation up to order five (i.e., MIS ERK(1,1, p) for $1 \leq p \leq 5$). When using forward Euler in both the outer and inner loops, the multirate algorithm is equivalent to a single-rate forward Euler extrapolation method. We then compare the MIS ERK(1,1, p) methods to the lowest order multirate extrapolation base method, which uses forward Euler to solve the slow process and ERK2 to solve the fast process (i.e., MIS ERK(1,2, p) for $1 \leq p \leq 5$). Results are displayed in Figure 3.2.1.

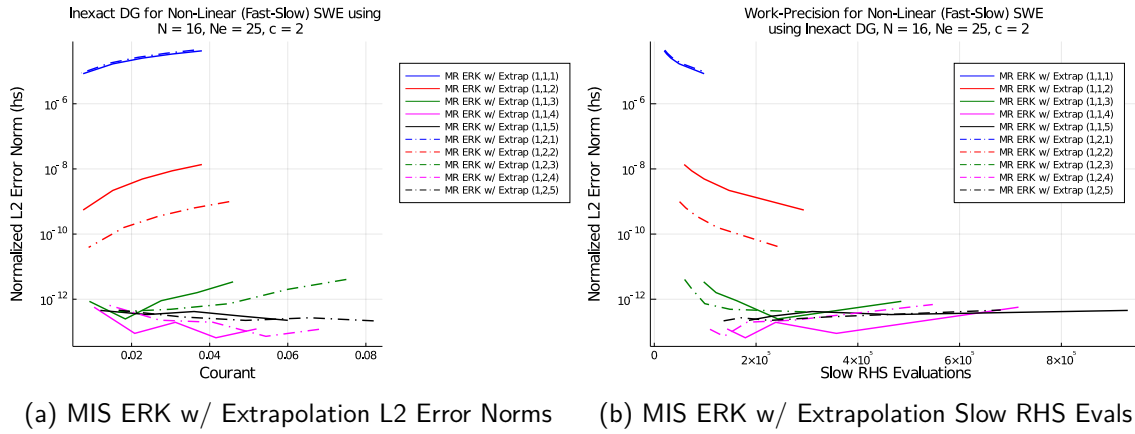


Figure 3.2.1. Error Plots and Slow RHS Evaluations for Non-Linear SWE using 1st Order MIS ERK(1, J , p) Method ($J = 1, 2$; $g = 10$, $h_B = 1$).

From Figure 3.2.1a we notice that when $gh_B = 10$, all MIS ERK(1,2, p) methods of order $p > 1$ are able to use a larger Courant value compared to their equivalent MIS ERK(1,1, p) method. Therefore the MIS ERK(1,2, p) methods for $2 \leq p \leq 5$ require fewer slow RHS evaluations, which demonstrates that our multirate extrapolation method performs better than single-rate extrapolation in terms of the number of slow RHS evaluations. These results are shown in Figure 3.2.1b. Additionally, we notice that for our given choice of spatial parameters, there is no added benefit in computing higher than a 4th order extrapolation as the spatial error begins to dominate the solution.

Notice that without extrapolation, both choices of (I, J) used within the multirate base method would only result in a first order accurate approximation; however, our extrapolated multirate method using Algorithm 3.1.1 indeed produces p^{th} order accuracy, which was confirmed using Equation (1.3.50) to compute the order of convergence for each method. It

is also important to note that the MIS ERK(I, J) method from Algorithms 2.3.1 or 2.3.2 is only at best 2^{nd} order accurate in time if both the I and J methods used are at least second order accurate [43]. However, the multirate method can achieve 3^{rd} order accuracy if both the inner and outer methods are at least third order accurate, and the outer method satisfies the additional order condition:

$$\sum_{i=1}^{s^O} (c_{i+1}^O - c_i^O) \sum_{j=1}^{i-1} (a_{i+1,j}^O + a_{i,j}^O) c_j^O = \frac{1}{3}. \quad (3.2.1)$$

Thus far, our numerical results for solving the non-linear shallow water equations using the MIS ERK(I, J) method without extrapolation have all been at best second order accurate. Although no 3^{rd} order MIS ERK results have been shown, we did achieve third order accuracy using the MIS ERK(4, J), for $J \geq 3$, where the 4^{th} order method used was the ERK4 ($\frac{3}{8}$ Rule) method found in Table A.0.5. Note, 3^{rd} order accuracy was not achieved when using the classical ERK4 from Table A.0.4. Furthermore, fourth order accuracy was not achieved for the MIS method when using at least 4^{th} order accurate ERK methods in both the inner and outer loops. This result is also confirmed by [43].

Therefore, the benefit in using Algorithm 3.1.1, is we can now generate solutions of higher order accuracy for the multirate infinitesimal step method, which could only previously achieve at best third order accuracy without extrapolation. The question we must now ask is, for what choices of (I, J, p) , if any, is the MIS ERK extrapolation method more efficient than simply using a single-rate method of equivalent order of accuracy? Although we have focused on the slow RHS evaluations, the number of fast RHS evaluations are not computationally free. Therefore, we define a computational ratio between the fast and slow processes of our MR method compared to a SR method of equivalent order as follows:

Definition: The slow to fast computational ratio, $\frac{S_{comp}}{F_{comp}}$, required for the MIS method to outperform a SR method of equivalent order is:

$$E_{SR}T_{comp} = E_{SR}(S_{comp} + \alpha F_{comp}) > SE_{MR}S_{comp} + FE_{MR}F_{comp}, \quad (3.2.2)$$

where the terms in Equation (3.2.2) are defined by:

- E_{SR} - is the total number of RHS evaluations for the single-rate method.

- SE_{MR} - is the total number of slow RHS evaluations for the multirate method.
- FE_{MR} - is the total number of fast RHS evaluations for the multirate method.
- S_{comp} - is the computational time to apply the slow RHS operator.
- F_{comp} - is the computational time to apply the fast RHS operator.
- T_{comp} - is the total computational time to apply the RHS operator.

If we assume that:

- $S_{comp} + \alpha F_{comp} = T_{comp}$, for some $0 \leq \alpha \leq 1$,
- $SE_{MR} < E_{SR} \leq FE_{MR}$,
- $E_{SR} = SE_{SR} = FE_{SR}$,

then a given multirate method will theoretically outperform a single-rate method of equivalent order if the slow to fast computational ratio, $\frac{S_{comp}}{F_{comp}}$, is:

$$\frac{S_{comp}}{F_{comp}} > \frac{FE_{MR} - \alpha E_{SR}}{E_{SR} - SE_{MR}}, \text{ for some } 0 \leq \alpha \leq 1. \quad (3.2.3)$$

Notice, that T_{comp} represents the total cost of applying the RHS operator. If the fast computation is free when computed along with the slow computation then $\alpha = 0$. If the fast and slow computations are inherently decoupled then $\alpha = 1$. In general, the more decoupled the processes are, the easier it will be for the multirate method to be computationally beneficial, such that Equation (3.2.3) provides us with an efficiency metric for determining what the slow to fast computational ratio bound should be for a given multirate method to be computationally more efficient than a single-rate method of equivalent order. For our respective IBVP, we have separated the fast and slow RHS operators within our code, such that $\alpha = 1$.

Since there is no known MIS ERK(I, J) method, without using extrapolation, that can achieve 4^{th} order accuracy, let's look at what the $\frac{S_{comp}}{F_{comp}}$ ratio bound should be when comparing our fourth order accurate MIS ERK(I, J, p) method for $p = 4$, against the single-rate ERK4 method when $gh_B = 10$ and 100. Comparison results are displayed in Table 3.2.1, where we notice that the number of slow RHS evaluations significantly decreases as J increases from 1 to 4 within the MIS ERK(1, J , 4) method. Furthermore, the MIS ERK(1,4,4) method's maximum stable Courant value is approximately double that of the

MIS ERK(1,1,4) method. However, using higher order methods within the inner loop of the MIS method, combined with the extrapolation process, increases the total number of fast RHS evaluations for the extrapolated multirate method. Therefore, from a quick comparison of the total number of slow and fast RHS evaluations, the SR ERK4 method was still more efficient than the MIS ERK(1, J , 4).

Table 3.2.1. Comparison of SR ERK4 Versus MIS ERK(1, J , 4) Methods for $1 \leq J \leq 4$ and $gh_B = 10, 100$ Using Algorithm 3.1.1.

SR ERK4 versus MIS ERK(1, J , 4) Method for $1 \leq J \leq 4$					
Gravity & Bathymetry Constants $g = 10, h_B = 1, gh_B = 10$	SR ERK4	MIS ERK(1,1,4)	MIS ERK(1,2,4)	MIS ERK(1,3,4)	MIS ERK(1,4,4)
Max Courant	0.052	0.052	0.068	0.068	0.1
Slow RHS Evaluations	57264	143160	109480	109480	74440
Fast RHS Evaluations	28632	71580	164220	328440	372200

SR ERK4 versus MIS ERK(1, J , 4) Method for $1 \leq J \leq 4$					
Gravity & Bathymetry Constants $g = 10, h_B = 10, gh_B = 100$	SR ERK4	MIS ERK(1,1,4)	MIS ERK(1,2,4)	MIS ERK(1,3,4)	MIS ERK(1,4,4)
Max Courant	0.0072	0.0072	0.0097	0.0097	0.0151
Slow RHS Evaluations	413552	1033880	767420	767420	492980
Fast RHS Evaluations	206776	516940	1151130	2302260	2464900

This should be no surprise, since the process of building the extrapolation tableau is costly. Therefore, for all choices of 4th order multirate methods above, our assumption that $SE_{MR} < E_{SR}$ is not satisfied and we cannot use Equation (3.2.3) to compute the slow to fast computational efficiency. In summary, the extrapolated multirate methods in Table 3.2.1 will not be more efficient than the single-rate ERK4 method when using Algorithm 3.1.1, due to a significant inefficiency within the extrapolation algorithm itself. For example, within both the inner and outer loops of Algorithm 3.1.1, more slow and fast RHS evaluations are computed than are computationally required.

From Table 1.3.1 in Chapter 1, we notice that our algorithm's inefficiency occurs when constructing each row of the extrapolation tableau. Notice that each $T_{e,1}$ solution from Algorithm 3.1.1 is built via a $Y_{e,e+1}$ tableau, where the first slow and fast RHS evaluations computed when solving the $Y_{e,1}$ solutions, use the same starting value at time t_n . This means that the first slow RHS computation for $S(Q_j)$ used to construct r_i within the slow process loop is the same computation for each row of the extrapolation process. Similarly, the first fast RHS computation of $F(V_{i,k})$ used to construct the $V_{i,j}$ solution within the

fast process loop is also the same for each row of the extrapolation process. Therefore, instead of repeating these two computations for both the fast and slow RHS processes, we can simply store the solution values at the start of each time-step. At the start of a new time-step, we then zero out the storage locations for both the fast and slow RHS solutions and start over. The modification to Algorithm 3.1.1 is shown in red text within Algorithm 3.2.1, and annotated as Version 2. This change can also be applied to Algorithms 1.3.1 and 1.3.2.

Algorithm 3.2.1	2-Partition General Explicit MIS Method w/ Extrapolation (Ver. 2)
$S_{temp} = S(q_n)$	◇ Compute slow and fast RHS evaluations at time t_n
$F_{temp} = F(q_n)$	
for $e = 1 : p$ do	◇ Loop constructs p approximations using harmonic step-sequence
$t_0 = t_n$	
$Y_{e,1} = q_n$	
for $d = 2 : e + 1$ do	◇ Loop builds temporary solution using $\frac{\Delta t}{e}$
$q_1 = Y_{e,d-1}$	
$Q_1 = q_1$	
for $i = 2 : I + 1$ do	◇ Loop through slow process
if $d = 2$ & $i = 2$ then	
$r_i = \sum_{j=1}^{i-1} \tilde{a}_{i,j}^O S_{temp}$	
else	
$r_i = \sum_{j=1}^{i-1} \tilde{a}_{i,j}^O S(Q_j)$	
end if	
$V_{i,1} = Q_{i-1}$	
for $j = 2 : J + 1$ do	◇ Loop through fast process
if $d = 2$ & $i = 2$ & $j = 2$ then	
$V_{i,j} = V_{i,j-1} + \frac{\Delta t}{e} \sum_{k=1}^{j-1} \tilde{a}_{j,k}^I [r_i +$	
$\tilde{c}_i^O F_{temp}]$	
else	
$V_{i,j} = V_{i,j-1} + \frac{\Delta t}{e} \sum_{k=1}^{j-1} \tilde{a}_{j,k}^I [r_i +$	
$\tilde{c}_i^O F(V_{i,k})]$	
end if	
end for	
$Q_i = V_{i,J+1}$	
end for	
$Y_{e,d} = Q_{I+1}$	
$t_0 = t_0 + \frac{\Delta t}{e}$	◇ Store temporary solution of step-sequence
end for	
$T_{e,1} = Y_{e,e+1}$	◇ Store final solution of step-sequence for $\frac{\Delta t}{e}$
end for	
for $k = 2 : p$ do	
for $j = k : p$ do	
$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\frac{j}{j-k+1} - 1}$	◇ Perform extrapolation using Aitken-Neville formula to order k
end for	
end for	
$q_{n+1} = T_{p,p}$	◇ Update final solution using p^{th} order approximation
$t_{n+1} = t_n + \Delta t$	

Using Algorithm 3.2.1, we recomputed the results from Table 3.2.1, which are displayed in Table 3.2.2.

Table 3.2.2. Comparison of SR ERK4 Versus MIS ERK(1, J , 4) Methods for $1 \leq J \leq 4$ and $gh_B = 10, 100$ Using Algorithm 3.2.1.

SR ERK4 versus MIS ERK(1, J , 4) Method for $1 \leq J \leq 4$					
Gravity & Bathymetry Constants $g = 10, h_B = 1, gh_B = 10$	SR ERK4	MIS ERK(1,1,4)	MIS ERK(1,2,4)	MIS ERK(1,3,4)	MIS ERK(1,4,4)
Max Courant	0.052	0.052	0.068	0.068	0.1
Slow RHS Evaluations	57264	100212	76636	76636	52108
Fast RHS Evaluations	28632	50106	147798	312018	361034
S_{comp}/F_{comp}	—	-0.50	-6.15	-14.63	64.47

SR ERK4 versus MIS ERK(1, J , 4) Method for $1 \leq J \leq 4$					
Gravity & Bathymetry Constants $g = 10, h_B = 10, gh_B = 100$	SR ERK4	MIS ERK(1,1,4)	MIS ERK(1,2,4)	MIS ERK(1,3,4)	MIS ERK(1,4,4)
Max Courant	0.0072	0.0072	0.0097	0.0097	0.0151
Slow RHS Evaluations	413552	723716	537194	537194	345086
Fast RHS Evaluations	206776	361858	1036017	2187147	2390953
S_{comp}/F_{comp}	—	-0.50	-6.71	-16.02	31.90

From these updated results we see a noticeable improvement in the overall efficiency of the extrapolated MIS ERK(I , J , p) method when using Algorithm 3.2.1 compared to Algorithm 3.1.1. For example, when comparing the MIS ERK(1,4,4) method in Table 3.2.2 to the single-rate ERK4 method, we notice that the total number of slow RHS evaluations for the MR method are now less than the SR method. Therefore, Equation (3.2.3) can now be used to determine what the slow to fast computational ratio bound should be for the extrapolated MIS ERK(1,4,4) method to perform better than the SR ERK4 method. However, for $1 \leq J \leq 3$, the SR ERK4 method still performs better as indicated by the negative $\frac{S_{comp}}{F_{comp}}$ ratio values. Note, a negative ratio value indicates the MR method cannot beat the SR method it is being compared to. Equation (3.2.4) provides the positive ratio results for various values of the bathymetry constant, h_B , when gravity is set to equal to 10, for the MIS ERK(1, J , 4) method compared to the SR ERK4 method.

$$\frac{S_{comp}}{F_{comp}} > \begin{cases} 64.47 & \text{for } g = 10, h_B = 1, gh_B = 10, \\ 31.90 & \text{for } g = 10, h_B = 10, gh_B = 100, \\ 31.04 & \text{for } g = 10, h_B = 100, gh_B = 1000. \end{cases} \quad (3.2.4)$$

Our results indicate that when $gh_B = 10$, our extrapolated MIS ERK(1,4,4) method should theoretically perform better than the single-rate ERK method if the cost of applying the slow RHS operator is approximately 65 times the cost of the fast RHS operator. This ratio was approximately halved when the gravity waves, gh_B , were greater than 100. Although the ratio bounds for $gh_B = 10$ and 100 are approximately equal, we assume the bathymetry increase from 10 to 100 is not enough to help determine whether the $\frac{S_{comp}}{F_{comp}}$ ratio would improve if gh_B were significantly increased to say 10,000 or higher. For example, the bathymetry within the Pacific Ocean near the Mariana Trench reaches a maximum depth close to 11,000 meters. Simulations for increased bathymetry were not computed for our SWE model due to the computational time required to run on our computer.

Next, we computed the results for the MIS ERK(2, J , 4) method, which uses a 2^{nd} order ERK method in the outer loop. Results are shown in Table 3.2.3.

Table 3.2.3. Comparison of SR ERK4 Versus MIS ERK(2, J , 4) Methods for $1 \leq J \leq 4$ and $gh_B = 10$ Using Algorithm 3.2.1.

SR ERK4 versus MIS ERK(2, J , 4) Method for $1 \leq J \leq 4$					
Gravity & Bathymetry Constants $g = 10, h_B = 1, gh_B = 10$	SR ERK4	MIS ERK(2,1,4)	MIS ERK(2,2,4)	MIS ERK(2,3,4)	MIS ERK(2,4,4)
Max Courant	0.052	0.144	0.157	0.125	0.14
Slow RHS Evaluations	57264	139590	128034	160812	143586
Fast RHS Evaluations	28632	129708	135147	348426	523823
S_{comp}/F_{comp}	—	-0.19	-1.51	-3.09	-5.74

From these results, we notice that there is no improvement in the extrapolated MIS ERK(2, J , 4) method compared to the MIS ERK(1, J , 4) methods, since both the total number of slow and fast RHS evaluations significantly increased within the MR method when $I = 2$. Specifically, using a 2^{nd} order ERK method in the outer loop did not increase the maximum stable Courant value enough to compensate for the additional outer loop stage evaluations of the slow process. Therefore, the MIS ERK(2, J , 4) methods required more slow RHS evaluations compared to the SR ERK4 method, such that all $\frac{S_{comp}}{F_{comp}}$ ratios were negative. We do not show results for increasing the fast wave speeds of gh_B for the MIS ERK(2, J , 4) methods as it is unnecessary.

Although our MIS ERK(I , J , p) method does show improved results when using Algorithm 3.2.1 compared to Algorithm 3.1.1, the $\frac{S_{comp}}{F_{comp}}$ ratios in Equation (3.2.4) can be further

reduced by increasing the number of fast steps the inner method uses to resolve the fast process. We refer to the additional fast steps within each time-step as “substepping”, where the substeps are applied to the MIS ERK(I, J) base method prior to the extrapolation process. We will use the notation, MIS ERK(I, J, M, p), to reference our p^{th} order, extrapolated multirate method using M substeps.

3.3 Extrapolated MIS ERK Method with Substepping

The concept of substepping is discussed in [3], where the method of substepping can be applied within each stage of the outer RK method used to solve the slow process. Options for choosing how many substeps to use can vary, such that a naïve choice would be to keep M constant within each outer RK stage, where M is determined by the ratio (R_s) of slow to fast time-steps:

$$M := R_s = \frac{\Delta t_{slow}}{\Delta t_{fast}}. \quad (3.3.1)$$

However, not all RK methods compute their respective stage evaluations at uniform time intervals. Therefore, another option is to vary the choice of M within each stage of the outer RK method based on its respective c_i values, which are found by looking at its Butcher tableau, or via the following RK condition:

$$c_i = \sum_{j=1}^{i-1} a_{i,j} \text{ for } i = 2, \dots, s, \text{ and } c_1 = 0. \quad (3.3.2)$$

Therefore, an optimal number of substeps to be used within the i^{th} outer RK stage of the MIS ERK method is:

$$M_i \approx \tilde{c}_i^O R_s, \quad (3.3.3)$$

where the $\tilde{c}_i^O$ in Equation (3.3.3) was defined in Chapter 2 using Equation (2.3.4). Algorithm 3.3.1 includes the substepping loop into the fast solution process.

To demonstrate our extrapolated multirate method with substepping algorithm, we have provided two graphical depictions of the MIS ERK(1,1,2,3) and MIS ERK(1,2,2,4) methods, which are located in Figures 3.3.1 and 3.3.2, respectively.

Algorithm 3.3.1 2-Partition General Explicit Extrapolated MIS Method w/ Substepping

$S_{temp} = S(q_n)$	◊ Compute slow and fast RHS evaluations at time t_n
$F_{temp} = F(q_n)$	
for $e = 1 : p$ do	◊ Loop constructs p approximations using harmonic step-sequence
$t_0 = t_n$	
$Y_{e,1} = q_n$	
for $d = 2 : e + 1$ do	◊ Loop builds temporary solution using $\frac{\Delta t}{e}$
$q_1 = Y_{e,d-1}$	
$Q_1 = q_1$	
for $i = 2 : I + 1$ do	◊ Loop through slow process
if $d = 2$ & $i = 2$ then	
$r_i = \sum_{j=1}^{i-1} \tilde{a}_{i,j}^O S_{temp}$	
else	
$r_i = \sum_{j=1}^{i-1} \tilde{a}_{i,j}^O S(Q_j)$	
end if	
$q_m = Q_{i-1}$	
for $m = 1 : M$ do	◊ Substepping loop
$V_{i,1} = q_m$	
for $j = 2 : J + 1$ do	◊ Loop through fast process
if $d = 2$ & $i = 2$ & $m = 1$ & $j = 2$ then	
$V_{i,j} = V_{i,j-1} + \frac{\Delta t}{Me} \sum_{k=1}^{j-1} \tilde{a}_{j,k}^I [r_i + \tilde{c}_i^O F_{temp}]$	
else	
$V_{i,j} = V_{i,j-1} + \frac{\Delta t}{Me} \sum_{k=1}^{j-1} \tilde{a}_{j,k}^I [r_i + \tilde{c}_i^O F(V_{i,k})]$	
end if	
end for	
$q_m = V_{i,J+1}$	
end for	◊ Substep solution update
$Q_i = q_m$	
end for	
$Y_{e,d} = Q_{I+1}$	
$t_0 = t_0 + \frac{\Delta t}{e}$	◊ Store temporary solution of step-sequence
end for	
$T_{e,1} = Y_{e,e+1}$	◊ Store final solution of step-sequence for $\frac{\Delta t}{e}$
end for	
for $k = 2 : p$ do	
for $j = k : p$ do	
$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\frac{j}{j-k+1} - 1}$	◊ Perform extrapolation using Aitken-Neville formula to order k
end for	
end for	
$q_{n+1} = T_{p,p}$	◊ Update final solution using p^{th} order approximation
$t_{n+1} = t_n + \Delta t$	

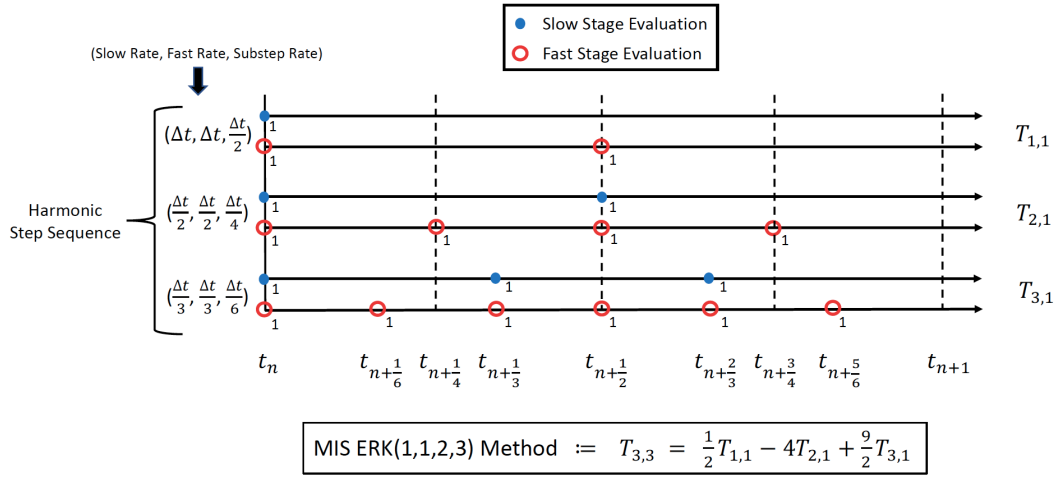


Figure 3.3.1. Example Diagram for the 3rd Order MIS ERK(1,1,2,3) Method using Harmonic Sequence.

To help explain Figure 3.3.1, each $T_{e,1}$ solution represents the numerical approximation using the MIS ERK(1,1,2) base method to advance the slow and fast processes from time t_n to time t_{n+1} using its respective $h_j = \frac{\Delta t}{n_j}$ step-size from the harmonic step sequence $n_j = \{1, 2, 3\}$. Because the overall method is the MIS ERK(1, 1, 2, 3), both the slow and fast processes are advanced using forward Euler; however, since $M = 2$, the fast process within each $T_{e,1}$ approximation uses two substeps of forward Euler within each of the outer forward Euler stage evaluations. For example, the $T_{3,1}$ approximation advances the slow process using 3 slow steps of forward Euler with a step-size of $\frac{\Delta t}{3}$. The fast process is advanced using 2 fast forward Euler steps inside each of the slow process stage evaluations with a step-size of $\frac{\Delta t}{6}$, for a total of 6 fast stage evaluations. On a single core, the $T_{1,1}$ approximation is computed first, followed by the $T_{2,1}$ and $T_{3,1}$ approximations. Once all $T_{e,1}$ approximations are computed, the MIS ERK(1,1,2,3) method constructs the extrapolation tableau by linearly combining the respective $T_{j,k}$ approximations via the Aitken-Neville algorithm. Finally, since $p = 3$, the final approximation of the IBVP solution at time t_{n+1} uses the $T_{p,p}$ approximation which is a linear combination of all $T_{e,1}$ approximations. The next figure demonstrates the MIS ERK(1,2,2,4) method.

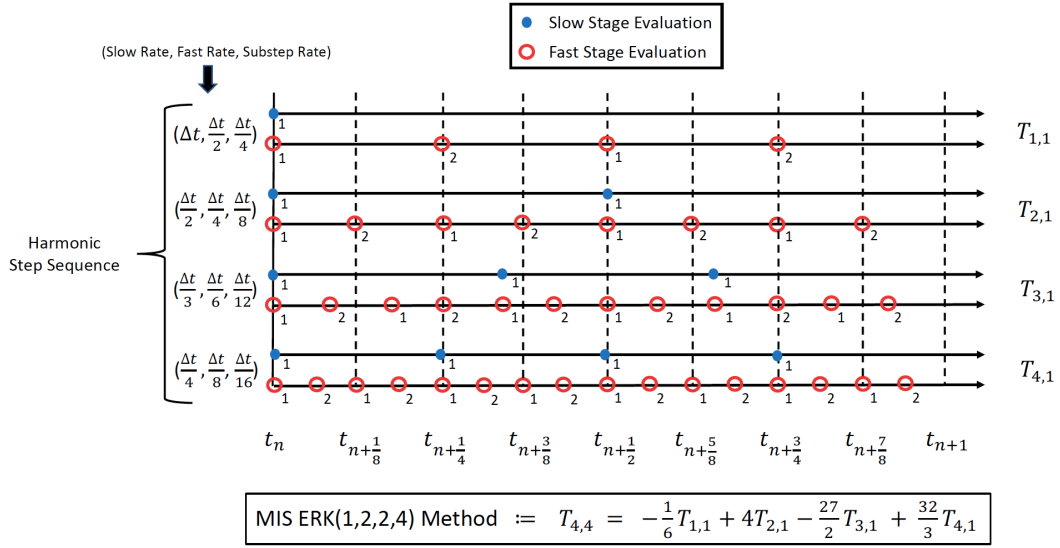


Figure 3.3.2. Example Diagram for the 4th Order MIS ERK(1,2,2,4) Method using Harmonic Sequence.

Notice that within Figures 3.3.1 and 3.3.2, we have provided the MIS ERK(I, J, m, p)'s equivalent $T_{j,k}$ representation using the p^{th} order extrapolation formula from Equation (1.3.27) within Chapter 1. It is important to emphasize that each $T_{j,k}$ solution represents a numerical approximation using the MIS ERK(I, J, m) method as the base method for extrapolation. In summary, Algorithm 3.3.1 uses Richardson extrapolation with the harmonic step sequence, $n_j = \{1, 2, 3, 4, \dots\}$, to extrapolate the MIS ERK(I, J, m) base method to a fixed order RK method on the order of $\mathcal{O}(\Delta t_{\text{slow}}^p)$. However, the overall numerical approximation will be on the order of $\mathcal{O}(\Delta x^{N+1}, \Delta t_{\text{slow}}^p)$, where N represents the polynomial order used within our DG spatial discretization for inexact integration. Let's now look at the numerical results for approximating our initial boundary value problem using Algorithm 3.3.1, when both substepping and extrapolation are applied to the MIS ERK base method.

3.4 Extrapolated MIS ERK with Substepping Results

In Section 3.2 we found that our extrapolated MIS ERK(I, J, p) method can be used to construct higher order multirate approximations; however, the computational cost of using extrapolation requires a relatively large slow to fast computational ratio for the multirate

method to beat out an explicit single-rate RK method of equivalent order. Therefore, we now look at the numerical results when substepping of the fast process is included.

We concluded Section 3.2 by comparing various 4^{th} order approximations; therefore, let's continue this approach using Algorithm 3.3.1. The simplest base method will use forward Euler for both the slow and fast process (i.e., $I = J = 1$), such that the inner forward Euler method is substepped for choices of $M = 1, \dots, 6$. We then extrapolate the respective substepped multirate base method for $p = 4$, and compare these results to the explicit single-rate RK4 method found in Equation (A.0.5). To compare all 4^{th} order extrapolated MR methods, we use both the total number of slow and fast RHS evaluations required for each method, as well as the $\frac{S_{comp}}{F_{comp}}$ ratio from Equation (3.2.3). The results are shown in Table 3.4.1.

Table 3.4.1. Comparison of SR ERK4 Versus MIS ERK(1, 1, m , 4) Methods for $1 \leq m \leq 6$ Using Algorithm 3.3.1.

SR ERK4 versus MIS ERK(1, 1, m , 4) Method for $1 \leq m \leq 6$ and $gh_B = 10$							
Gravity & Bathymetry Constants $g = 10, h_B = 1$	SR ERK4	MIS ERK (1,1,1,4)	MIS ERK (1,1,2,4)	MIS ERK (1,1,3,4)	MIS ERK (1,1,4,4)	MIS ERK (1,1,5,4)	MIS ERK (1,1,6,4)
Max Courant	0.052	0.052	0.136	0.158	0.192	0.226	0.262
Slow RHS Evaluations	57264	100212	38318	32984	27146	23058	19894
Fast RHS Evaluations	28632	50106	46529	63612	71743	77409	80997
Total Evaluations	85896	150318	84847	96596	98889	100467	100891
Approximate Error	10^{-13}	10^{-13}	10^{-14}	10^{-13}	10^{-13}	10^{-13}	10^{-13}
S_{comp}/F_{comp}	—	-0.50	0.94	1.44	1.43	1.43	1.40

SR ERK4 versus MIS ERK(1, 1, m , 4) Method for $1 \leq m \leq 6$ and $gh_B = 100$							
Gravity & Bathymetry Constants $g = 10, h_B = 10$	SR ERK4	MIS ERK (1,1,1,4)	MIS ERK (1,1,2,4)	MIS ERK (1,1,3,4)	MIS ERK (1,1,4,4)	MIS ERK (1,1,5,4)	MIS ERK (1,1,6,4)
Max Courant	0.0072	0.0072	0.0194	0.022	0.0268	0.0317	0.0368
Slow RHS Evaluations	413552	723716	268604	236852	194432	164388	141596
Fast RHS Evaluations	206776	361858	326162	456786	513856	551874	576498
Total Evaluations	620328	1085574	594766	693638	708288	716262	718094
Approximate Error	10^{-13}	10^{-13}	10^{-13}	10^{-13}	10^{-13}	10^{-13}	10^{-13}
S_{comp}/F_{comp}	—	-0.50	0.82	1.41	1.40	1.39	1.36

From Table 3.4.1 we make the following observations. First, as the number of substeps increases, the maximum stable Courant used also increased. Therefore, for $m \geq 2$, the MIS ERK(1, 1, m , 4) methods all used significantly fewer slow RHS evaluations compared to the single-rate ERK4 method. However, additional substepping of the fast process increased

the total number of fast RHS evaluations within the MR methods. Second, the $\frac{S_{comp}}{F_{comp}}$ ratios computed indicate that our 4th order extrapolated MR methods with substepping can easily outperform the explicit 4th order single-rate method provided the cost of applying the slow RHS operator is approximately 1.4 times that of applying the fast RHS operator, assuming both the fast and slow operators are independent of each other. Third, the MIS ERK(1, 1, 1, 4) method is equivalent to performing single-rate forward Euler extrapolation. The negative ratio indicates this method cannot beat out the single-rate ERK4 method as both the total number of fast and slow RHS evaluations within the MIS ERK(1, 1, 1, 4) method are greater than the single-rate methods. Let's look at one more example, where we instead compare an explicit 5th order single-rate ERK method to our MIS ERK(I, J, m, p) method. Once again, we will use the simplest MR base method choice, such that $I = J = 1$.

Table 3.4.2. Comparison of SR ERK5 Versus MIS ERK(1, 1, m , 5) Methods for $1 \leq m \leq 6$ Using Algorithm 3.3.1.

SR ERK5 versus MIS ERK(1, 1, m , 5) Method for $1 \leq m \leq 6$ and $gh_B = 10$							
Gravity & Bathymetry Constants $g = 10, h_B = 1$	SR ERK5	MIS ERK (1,1,1,5)	MIS ERK (1,1,2,5)	MIS ERK (1,1,3,5)	MIS ERK (1,1,4,5)	MIS ERK (1,1,5,5)	MIS ERK (1,1,6,5)
Max Courant	0.063	0.06	0.135	0.194	0.24	0.27	0.302
Slow RHS Evaluations	70896	136488	60654	42218	34122	30338	27126
Fast RHS Evaluations	35448	68244	71682	78679	86856	97909	106038
Total Evaluations	106344	204732	132336	120897	120978	128247	133164
Approximate Error	10^{-14}	10^{-13}	10^{-13}	10^{-13}	10^{-13}	10^{-13}	10^{-13}
S_{comp}/F_{comp}	—	-0.50	3.54	1.51	1.40	1.54	1.61

SR ERK5 versus MIS ERK(1, 1, m , 5) Method for $1 \leq m \leq 6$ and $gh_B = 100$							
Gravity & Bathymetry Constants $g = 10, h_B = 10$	SR ERK5	MIS ERK (1,1,1,5)	MIS ERK (1,1,2,5)	MIS ERK (1,1,3,5)	MIS ERK (1,1,4,5)	MIS ERK (1,1,5,5)	MIS ERK (1,1,6,5)
Max Courant	0.0088	0.0084	0.019	0.027	0.034	0.0377	0.041
Slow RHS Evaluations	507540	974798	430980	303270	240834	217206	199716
Fast RHS Evaluations	253770	487399	509340	565185	613032	700983	780708
Total Evaluations	761310	1462197	940320	868455	853866	918189	980424
Approximate Error	10^{-14}	10^{-13}	10^{-13}	10^{-13}	10^{-13}	10^{-12}	10^{-12}
S_{comp}/F_{comp}	—	-0.50	3.34	1.52	1.35	1.54	1.71

Comparing the results within Table 3.4.2, we notice that for $m \geq 2$, our 5th order MIS ERK(1, 1, m , 5) method will easily outperform the single-rate ERK5 method from Table A.0.6. Specifically, if $m \geq 3$, the $\frac{S_{comp}}{F_{comp}}$ ratio is approximately 1.5, with the best result using $m = 4$ substeps. We also compared our 5th order MR method against the Dormand-Prince ODE45 method in Table A.0.7, achieving almost identical results to the ERK5 comparison.

In the first two examples, we chose to use forward Euler to approximate both the fast and slow processes, with the fast process being substepped. However, our generalized multirate extrapolation algorithm allows for higher order RK methods to be used within both the fast and slow processes. Within Appendix D we have placed the numerical results for all MIS ERK(I, J, m, p) simulations of the non-linear shallow water equation IBVP using $I = 1, 1 \leq J \leq 3, 1 \leq m \leq 6$ and $1 \leq p \leq 5$.

3.4.1 MIS ERK Method with Substepping (No Extrapolation)

If no extrapolation is desired (i.e., $p = 1$), Algorithm 3.3.1 simplifies to Algorithm 3.4.1, where we have also included the time updates for non-autonomous problems.

Algorithm 3.4.1 2-Partition General Explicit MIS Method with Substepping	
$Q_1 = q_n$	
for $i = 2 : I + 1$ do	◊ Loop through slow process
$r_i = 0$	
for $j = 1 : i - 1$ do	
$t_j = t_n + \Delta t c_j^O$	◊ Time update for outer loop
$r_i = r_i + \tilde{a}_{i,j}^O S(Q_j, t_j)$	◊ Build r_i (slow solution)
end for	
$q_m = Q_{i-1}$	
for $m = 1 : M$ do	◊ Substepping loop
$V_{i,1} = q_m$	
for $j = 2 : J + 1$ do	◊ Loop through fast process
$R_{sum} = 0$	
for $k = 1 : j - 1$ do	
$t_k = t_n + \Delta t (c_{i-1}^O + (\tilde{c}_i^O * c_k^I) \frac{m-1}{M})$	◊ Time update for inner loop
$R_{sum} = R_{sum} + \tilde{a}_{j,k}^I [r_i + \tilde{c}_i^O F(V_{i,k}, t_k)]$	◊ Build R_{sum} (slow + fast solution)
end for	
$V_{i,j} = V_{i,j-1} + \frac{\Delta t}{M} R_{sum}$	
end for	
$q_m = V_{i,J+1}$	◊ Substep solution update
end for	
$Q_i = q_m$	
end for	
$q_{n+1} = Q_{I+1}$	
$t_{n+1} = t_n + \Delta t$	

Results for substepping without extrapolation can also be found in the first column of each table within Appendix D; however, all MIS ERK($1, J, m, 1$) results for $1 \leq J \leq 3$ and $1 \leq m \leq 6$ are only first order accurate, regardless of the order of method used to

approximate the fast process. Since $I = 1$, the overall time-integration scheme can only be at best 1^{st} order.

At the start of this chapter, we mentioned additional improvements that could be made to the MIS ERK method. The first improvement was to add substepping to the fast process within the inner loop of the MIS ERK algorithm, which was shown in Sections 3.1 - 3.4. The next improvement that can be made is not to the MIS ERK base method itself; instead, it is an improvement to the overall efficiency of the extrapolation process. Specifically, within Algorithms 3.1.1 - 3.3.1, the outer most loop (i.e., e -loop) of the extrapolation process can be performed in parallel. Therefore, the next section will briefly describe a way in which parallelization of the extrapolation process can be applied.

3.5 Parallelizing the Extrapolated MIS ERK Methods

In this section we develop the speedup one can theoretically achieve by parallelizing the extrapolation process. Specifically, we will use the 4^{th} order extrapolated MIS ERK(1, 2, 2, 4) method from Figure 3.3.2 as the reference method for parallelization. In Chapter 1 we showed the following extrapolation tableau for a p^{th} order method using Richardson extrapolation.

Table 3.5.1. Tableau of $T_{j,k}$ Solutions with Corresponding Base Method Orders

(a) $T_{j,k}$ Solution Tableau					(b) Classical Orders for a p^{th} -order Base Method				
T_{11}					p				
T_{21}	T_{22}				p	$p + 1$			
T_{31}	T_{32}	T_{33}			p	$p + 1$	$p + 2$		
$\vdots$	$\vdots$	$\vdots$	$\ddots$		$\vdots$	$\vdots$	$\vdots$	$\ddots$	
$T_{M,1}$	$T_{M,2}$	$T_{M,3}$	$\cdots$	$T_{M,M}$	p	$p + 1$	$p + 2$	$\cdots$	$p + M - 1$

From this table it is easily seen that each row of the tableau is inherently parallelizable, as opposed to an ERK method whose stage evaluations must be computed sequentially, as seen in Equations (2.1.6) and (2.1.12); therefore, the stage evaluations of an ERK method cannot be independently evaluated. Parallelization of the extrapolation process isn't a new idea. For prior examples, see the works of [13], [59], [60].

We begin by looking back at Figure 3.3.2, where we specifically focus our attention to the total number of RHS evaluations one core must evaluate within a single time-step versus two cores in parallel. For the MIS ERK(1,2,2,4) method, the total number of evaluations are calculated in Table 3.5.2, and can be graphically seen within Figures 3.5.1 and 3.5.2.

Table 3.5.2. MIS ERK(1, 2, 2, 4) Method: Parallel Comparison for Total Number of RHS Evaluations Per Time-Step

Num. Evals. (per time-step)	Single Core (total)	Two Core (maximum)
Slow RHS Operator	7	4
Fast RHS Operator	37	19

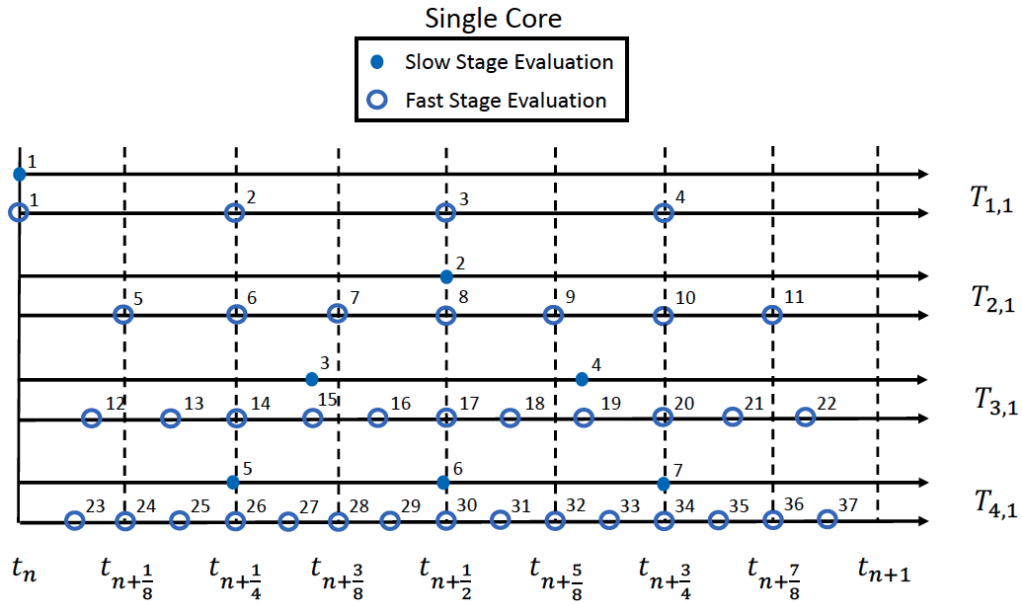


Figure 3.5.1. Single Core Implementation for the 4th Order MIS ERK(1,2,2,4) Method Using Harmonic Sequence.

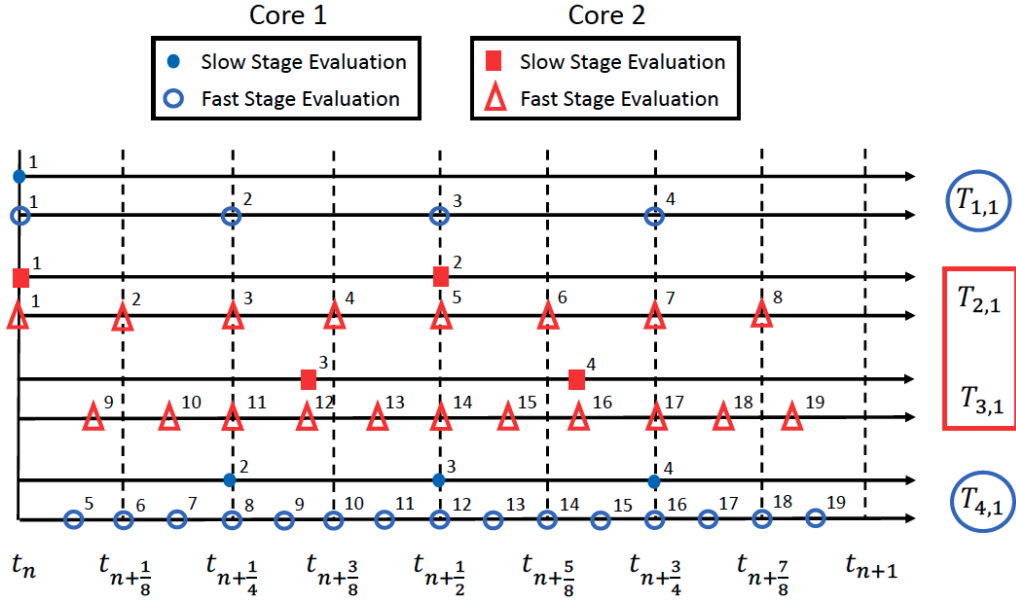


Figure 3.5.2. Two Core Parallel Implementation for the 4th Order MIS ERK(1,2,2,4) Method Using Harmonic Sequence.

Reviewing Figures 3.5.1 and 3.5.2, the single core implementation in Figure 3.5.1 requires a total of 7 slow and 37 fast RHS evaluations per time-step, compared to a maximum of 4 slow and 19 fast RHS evaluations when using two cores in Figure 3.5.2 to construct the 4th order approximation. Notice that in both the single core and two core implementations, each core only needs to compute the slow and fast RHS evaluations at time, t_n , for the first $T_{i,1}$ solution they approximate. This idea was briefly discussed when we made efficiency modifications to the extrapolated MIS ERK method in Algorithm 3.2.1.

The theoretical speedup, $\hat{S}_p$, for the parallel implementation of the MIS ERK(I, J, m, p) method is approximately:

$$\hat{S}_p \approx \hat{S}_{p_{slow}} + \hat{S}_{p_{fast}}, \quad (3.5.1)$$

where, $\hat{S}_{p_{slow}}$ and $\hat{S}_{p_{fast}}$, represent the theoretical speedup per time-step of the slow and fast processes, respectively. The hat notation ($\hat{\cdot}$) above the S indicates the variable is a “speedup” term, as opposed to the standard use of S throughout this dissertation, which has

represented a “slow” term. The speedup terms are calculated as follows:

$$\hat{S}_{p_{slow}} = \frac{S_{stages} - (p - 1)}{\max\left(S_{p_{stages}} - \left(\frac{p}{C} - 1\right)\right)} = \frac{\text{slow serial stages}}{\text{maximum number of slow parallel stages}}, \quad (3.5.2a)$$

$$\hat{S}_{p_{fast}} = \frac{F_{stages} - (p - 1)}{\max\left(F_{p_{stages}} - \left(\frac{p}{C} - 1\right)\right)} = \frac{\text{fast serial stages}}{\text{maximum number of fast parallel stages}}, \quad (3.5.2b)$$

where the $p - 1$ and $\frac{p}{C} - 1$ terms represent the total number of serial and parallel RHS stage evaluations at time t_n , respectively, that Algorithm 3.3.1 does not compute.

For example, within Figure 3.5.1 we notice that a total of three slow and three fast stage evaluations are not required when constructing a 4th order approximation. Therefore, since $p = 4$, we subtract a total of $4 - 1 = 3$ stages from both the total number of slow and fast serial stage evaluations within the numerators of Equations (3.5.2a) - (3.5.2b). Likewise, within Figure 3.5.2, we notice that a total of $\frac{p}{C} - 1$ stages can be subtracted from the parallel implementation, assuming we choose the total number of cores, such that $\frac{p}{C} = 2n$ for an $n \in \mathbb{N}$ (i.e., must be an even integer) and $C \leq p$.

Note, we say approximate because there are other factors a parallel implementation must consider, such as memory allocation, communication between cores, and load balancing. Finally, the theoretical efficiency, E , that can be achieved in parallel is determined by dividing the theoretical speedup, $\hat{S}_p$, by the total number of cores, C , used [13], such that:

$$E = \frac{\hat{S}_p}{C}. \quad (3.5.3)$$

3.5.1 Determining the Total Number of RHS Evaluations

When using explicit RK schemes within the MIS ERK(I, J, m, p) method, the total number of stage evaluations per time interval, T_{stages} , for the serial implementation is determined as follows:

$$T_{s_{stages}} = -2(p-1) + \sum_{i=1}^p \left(iI(1+Jm) \right), \quad (3.5.4)$$

where the summation term in Equation (3.5.4) contains both the slow and fast RHS stage evaluations per time-step. They can be individually computed as follows:

$$S_{s_{stages}} = \sum_{i=1}^p iI, \quad (3.5.5a)$$

$$F_{s_{stages}} = \sum_{i=1}^p iIJm, \quad (3.5.5b)$$

such that $T_{s_{stages}} = -2(p-1) + S_{s_{stages}} + F_{s_{stages}}$. The $-2(p-1)$ represents the computational savings by storing the first fast and slow RHS evaluation within each time interval.

Note, Equations (3.5.4) - (3.5.5b) assume the following:

- The extrapolation process uses the harmonic sequence (i.e., $n_i = \{1, 2, 3, 4, 5, \dots\}$), such that the step-size sequence is found by $h_i = \{\frac{\Delta t}{n_i}\}$, for $i = 1, 2, \dots, p$.
- The number of stage evaluations for I and J are equal to the minimum number of stages within the standard SR ERK method. Table 3.5.3 provides both the minimum number of stage evaluations and order conditions required for these methods [11].

Table 3.5.3. Minimum Number of Stage Evaluations and Order Conditions Required for SR ERK Methods up to Order 8.

ERK Order (p)	1	2	3	4	5	6	7	8
Minimum No. of ERK Stages	1	2	3	4	6	7	9	11
Minimum No. of Order Conditions	1	2	4	8	17	37	85	200

The total number of stage evaluations within a parallel implementation is not as straight forward for several reasons. First, a choice must be made on how to effectively divide up the work each core is responsible for evaluating, such that each core is kept as busy as possible (i.e., proper load balancing). Additionally, communication between all cores must be

established, with a single core responsible for evaluating the final numerical approximation at the end of each time interval via the Aitken-Neville formula.

To parallelize the MIS ERK(1,2,2,4) method, we used two cores (i.e., $C = 2$), such that optimal load balancing was achieved by dividing the work between each core. Specifically, core 1 is responsible for computing the numerical approximations for $T_{1,1}$ and $T_{4,1}$ using step sequence values $\{h_1, h_4\}$ and core 2 is responsible for approximating $T_{2,1}$ and $T_{3,1}$ using step sequence values $\{h_2, h_3\}$, respectively. This is displayed in Figure 3.5.2 above.

Similar to Equations (3.5.4) - (3.5.5b), the total number of parallel stage evaluations for core 1 and core 2 can be computed using the following formulas:

$$\text{Core 1: } T_{p1_{stages}} = -2\left(\frac{p}{C} - 1\right) + (n_1 + n_4)(I + IJm), \quad (3.5.6a)$$

$$\text{Core 2: } T_{p2_{stages}} = -2\left(\frac{p}{C} - 1\right) + (n_2 + n_3)(I + IJm). \quad (3.5.6b)$$

The slow and fast parallel stage evaluations for each core can be computed individually, such that:

$$S_{p1_{stages}} = (n_1 + n_4)I, \quad \text{and} \quad F_{p1_{stages}} = Jm(S_{p1_{stages}}). \quad (3.5.7)$$

Additionally,

$$S_{p2_{stages}} = (n_2 + n_3)I, \quad \text{and} \quad F_{p2_{stages}} = Jm(S_{p2_{stages}}). \quad (3.5.8)$$

We can now compute the theoretical parallel speedup per time-step with two cores for the MIS ERK(1,2,2,4) method using Equation (3.5.1), such that:

$$\hat{S}_p \approx \frac{7}{4} + \frac{37}{19} = 3.697. \quad (3.5.9)$$

Furthermore, the theoretical parallel efficiency for the 4th order extrapolated multirate method is:

$$E = \frac{\hat{S}_p}{C} \approx \frac{3.697}{2} \approx 1.85, \quad (3.5.10)$$

such that Equation (3.5.10) implies that a two-core parallel implementation of the MIS ERK(1,2,2,4) method would be approximately 1.85 times faster than the current serial implementation.

In summary, parallelization in time of the MIS ERK(I, J, m, p) method should be considered if additional resources (i.e., cores) are available for computation. However, the computational efficiency of the overall numerical method may be better achieved through parallelization of the spatial discretization method. Although we have not parallelized our code in either space or time, our choice for using element based Galerkin methods lends itself easily to a parallel in space implementation. Theoretically speaking, if N cores are available, such that the upper limit of speedup via a parallel in space scheme is achieved using M cores, for $M < N$, then further parallelization in time could potentially be achieved if $N/M > 2$, since each parallel RHS evaluation requires M cores.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:

Nonhydrostatic Unified Model of the Atmosphere (NUMA)

In this chapter we use our extrapolated multirate time-integration schemes developed in Chapter 3 to solve both a two-dimensional non-linear shallow water equation model with artificial viscosity and both a two and three-dimensional compressible Euler equations model. Both models are currently used within the Nonhydrostatic Unified Model of the Atmosphere (NUMA), which is used for simulating atmospheric processes (e.g., weather prediction). For consistency with Chapter 3 results, we will compare results using 4th order accurate time-integration schemes.

4.1 Governing Equations for the 2D Shallow Water Model

In Chapter 1 we showed the two-dimensional shallow water equation model in Equations (1.1.6a) - (1.1.6b), such that:

$$\frac{\partial h}{\partial t} + \nabla \cdot \mathbf{U} = 0 \quad (4.1.1a)$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \left[\frac{\mathbf{U} \otimes \mathbf{U}}{h} + \frac{1}{2}gh_s^2\mathbf{I}_2 + gh_Bh_s\mathbf{I}_2 \right] = gh_s\nabla(h_B) + \nabla \cdot (\nu\nabla(\mathbf{U})), \quad (4.1.1b)$$

where $\mathbf{I}_2$ is the rank-2 identity matrix, $\mathbf{U} = h\mathbf{u}$, where $\mathbf{u} = (u, v)^\top$ and $h = h_s + h_B$. We also assumed a non-viscous fluid and constant bathymetry in Chapter 1, so that both $gh_s\nabla(h_B) = \mathbf{0}$ and $\nabla \cdot (\nu\nabla(\mathbf{U})) = \mathbf{0}$ on the RHS of Equation (4.1.1b).

Now, if we let $\phi = gh$ and multiply both equations in (4.1.1) by g , then we can write:

$$\frac{\partial \phi_s}{\partial t} + \nabla \cdot \mathbf{U} = 0 \quad (4.1.2a)$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \left[\frac{\mathbf{U} \otimes \mathbf{U}}{\phi} + \frac{1}{2}\phi_s^2\mathbf{I}_2 + \phi_B\phi_s\mathbf{I}_2 \right] = \phi_s\nabla(\phi_B) + \nabla \cdot (\nu\nabla(\mathbf{U})), \quad (4.1.2b)$$

where $\mathbf{U} = \phi\mathbf{u}$ and $\phi = \phi_s + \phi_B$.

If we assume a constant bathymetry then $\phi_s \nabla(\phi_B) = \mathbf{0}$ and $\frac{\partial \phi}{\partial t} = \frac{\partial \phi_s}{\partial t}$; however, in our NUMA SWE model (4.1.2), we will keep the diffusion operator, which contains the viscosity constant ν . As shown in [3], diffusion operators can be introduced into a model for increased stability, where

$$\nabla \cdot (\nu \nabla(\mathbf{U})) = \nu \nabla^2(\mathbf{U}), \quad (4.1.3)$$

such that the Laplacian, ∇^2 represents a 2^{nd} order diffusion operator. In general, we can increase the order of the diffusion process by constructing a $2K$ order Laplacian (i.e., ∇^{2K}). For our purposes, the additional diffusion operator is added into the 2D SWE model to represent a third (slow) process that is computationally expensive as the order of K increases; therefore, we will assume a very small viscosity coefficient, such that the artificial viscosity is negligible and ensures the diffusion operator does not become a dominant process within the PDE system. For example, $K = 2$ represents a 4^{th} order Laplacian.

Using Galerkin methods, we can construct the Laplacian operator by discretizing the diffusion operator using inexact integration within each element. First, we multiply the diffusion term in Equation (4.1.2b) by a test function, ψ_i , and then apply integration by parts as follows:

$$\int_{\Omega_e} \psi_i \nabla^2 \mathbf{U}_N^{(e)} d\Omega_e = \int_{\Gamma_e} \psi_i \hat{\mathbf{n}} \cdot \nabla \mathbf{U}_N^{(e)} d\Gamma_e - \int_{\Omega_e} \nabla \psi_i \cdot \nabla \mathbf{U}_N^{(e)} d\Omega_e. \quad (4.1.4)$$

In general, a $2K$ order Laplacian operator is computed as follows [3]:

$$\nabla^{2K} \mathbf{U}_I = \left[\prod_{k=1}^K \left(-M_I^{-1} L_{I,J} \mathbf{U}_J \right) \right], \quad (4.1.5)$$

where we have ignored the boundary integral for simplicity and M_I and $L_{I,J}$ represent the global mass vector and global Laplacian matrix, respectively. Now if we let $\mathbf{q} = (\phi_s, \mathbf{U})^T$, then we can rewrite Equations (4.1.2a) - (4.1.2b) in compact form as either a two or three process PDE.

4.1.1 Splitting Option One: Two Processes

The two process PDE system we consider is:

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{S}(\mathbf{q}) + \mathbf{F}(\mathbf{q}), \quad (4.1.6)$$

where the slow (diffusion) process is:

$$\mathbf{S}(\mathbf{q}) = \begin{pmatrix} 0 \\ \nu \nabla^{2K}(\mathbf{U}) \end{pmatrix}, \quad (4.1.7)$$

and the fast (advection plus gravity) is:

$$\mathbf{F}(\mathbf{q}) = - \begin{pmatrix} \nabla \cdot \mathbf{U} \\ \nabla \cdot (\frac{\mathbf{U} \otimes \mathbf{U}}{\phi}) + \nabla \cdot (\phi_B \phi_s \mathbf{I}_2 + \frac{1}{2} \phi_s^2 \mathbf{I}_2) \end{pmatrix}. \quad (4.1.8)$$

4.1.2 Splitting Option Two: Three Processes

The three process PDE system we consider is:

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{S}_1(\mathbf{q}) + \mathbf{S}_2(\mathbf{q}) + \mathbf{F}(\mathbf{q}), \quad (4.1.9)$$

where the slowest (diffusion) process is:

$$\mathbf{S}_1(\mathbf{q}) = \begin{pmatrix} 0 \\ \nu \nabla^{2K}(\mathbf{U}) \end{pmatrix}, \quad (4.1.10)$$

the medium (advection) process is:

$$\mathbf{S}_2(\mathbf{q}) = - \begin{pmatrix} \nabla \cdot \mathbf{U} \\ \nabla \cdot (\frac{\mathbf{U} \otimes \mathbf{U}}{\phi}) \end{pmatrix}, \quad (4.1.11)$$

and the fast (gravity) process is:

$$\mathbf{F}(\mathbf{q}) = - \begin{pmatrix} 0 \\ \nabla \cdot (\phi_B \phi_s \mathbf{I}_2 + \frac{1}{2} \phi_s^2 \mathbf{I}_2) \end{pmatrix}. \quad (4.1.12)$$

4.2 NUMA 2D Shallow Water Results

NUMA uses a unified continuous/discontinuous Galerkin (CG/DG) method for spatial discretization. Although we used an inexact DG method within our 1D SWE model case, our extrapolated multirate time-integration method was initially implemented into NUMA's CG framework for testing. Therefore, all NUMA results presented within this chapter use an inexact CG approach to approximate the spatial derivatives.

The first set of 2D SWE simulations computed within NUMA used 4th order polynomial interpolation within 40 elements in both the x and y directions, for $-100 \leq x \leq 100$ and $-100 \leq y \leq 100$. We varied the order of the Laplacian operator, as well as the bathymetry constant, and all simulations were computed for time (seconds), $0 \leq t \leq 2$, unless otherwise stated. Furthermore, the initial condition for the 2D SWE model is defined within the spatial domain as follows:

$$r = \sqrt{x^2 + y^2}, \quad (4.2.1a)$$

$$r_c = 2.5, \quad (4.2.1b)$$

$$h_s = \begin{cases} \frac{\alpha}{2} \left(1 + \cos\left(\frac{\pi r}{r_c}\right) \right), & \text{if } r \leq r_c \\ 0, & \text{otherwise} \end{cases} \quad (4.2.1c)$$

$$\mathbf{q}(x, y, 0) = \left(g(h_s + h_B), 0, 0 \right)^\top, \quad (4.2.1d)$$

where h_s is the surface height, h_B is the bathymetry, the gravity constant $g = 10$, and amplification constant $\alpha = 2$. Additionally, we use no-flux boundary conditions such that $\hat{\mathbf{n}} \cdot \mathbf{u} = 0$ along all physical boundaries so that fluid flow is constrained within the spatial domain.

One of the most efficient single-rate methods used in NUMA is the 4th order, low-storage Runge-Kutta (LSRK) method (LSRK4-5), which is a 5-stage method specifically optimized for approximating hyperbolic problems (see [61], [62]). Table 4.2.1 shows the results comparing our extrapolated MR method with substepping, against the LSRK4-5 method when splitting option one from Section 4.1.1 is used. The Butcher tableau for LSRK4-5 is shown in Appendix A.

Table 4.2.1. 4th Order Comparison Using the MIS ERK(1, 1, m , 4) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 1$ and $\nu = 0.001$ Using Splitting Option One.

SR LSRK4-5 and LSRK4-14 Methods versus the MIS ERK(1, 1, m , 4) for $m \in \{2, 10, 20, 50, 100\}$							
Time-Integration Method	LSRK4-5	MIS ERK (1,1,2,4)	MIS ERK (1,1,10,4)	MIS ERK (1,1,20,4)	MIS ERK (1,1,50,4)	MIS ERK (1,1,100,4)	MIS ERK (1,LSRK4-5,1,1)
Max Δt	0.0453	0.056	0.097	0.147	0.258	0.385	0.0453
Diffusion Count	450	720	420	280	160	120	90
Advection Count	225	720	2100	2800	4000	6000	225
Gravity Count	225	720	2100	2800	4000	6000	225
Total RHS Evaluations	900	2160	4620	5880	8160	12120	540
Max Courant	1.66	2.05	3.56	5.39	9.46	14.14	1.66
Wall Clock Time (sec)	1.66	2.21	3.00	3.71	4.31	5.26	1.10
RHS Comp Time (sec)	0.36	1.00	1.64	2.00	2.82	3.87	0.19
Loss of Mass	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
S_{comp}/F_{comp}	—	-3.67	125.00	30.29	26.03	35.00	0.00

*Note, all NUMA simulations for the 2D SWE model were computed on the Naval Postgraduate School's Hamming supercomputer using 20 cores on AMD Epyc processors.

Remember that the MIS ERK(I, J, m, p) method using Algorithm 3.3.1 allows us the flexibility to choose two different methods to advance both the slow and fast processes respectively, in addition to the number of inner substeps, m , and fixed order of extrapolation, p . However, for simplicity we have chosen to use forward Euler to advance both the slow (diffusion) process and the combined fast (advection plus gravity) processes. We varied the substepping using values of $m \in \{2, 10, 20, 50, 100\}$ and fixed the extrapolation order using $p = 4$. Additionally, our initial simulation used a 2nd order Laplacian within the diffusion process, such that $K = 1$ with an artificial viscosity constant of $\nu = 0.001$. Our choice for using a small value for the viscosity was to ensure the diffusion operator was not the dominant process within the model, yet added a third process to the model with added computational cost.

It is easy to observe from Table 4.2.1 that all of the MIS ERK(1, 1, m , 4) methods were able to use a significantly larger maximum step-size compared to that of the LSRK4-5 method, such that Δt increased as the number of substeps increased. For the multirate methods, the larger step-size reduced the total number of RHS computations of the slower diffusion process; albeit at the cost of increasing the number of fast RHS computations of the advection and gravity processes. Although a method's efficiency is related to the total

number of RHS evaluations it must compute throughout a simulation, this does not tell the entire story. For example, the computational time to apply the diffusion process will vary depending on the order $2K$ of the Laplacian operator. Therefore, we can control how computationally expensive the diffusion operator is by simply increasing the value of K .

The Table 4.2.1 results clearly show that LSRK4-5 outperformed all of the multirate methods when $K = 1$; however, with only using two substeps between the slow and fast processes within the extrapolated MR method, the MIS ERK(1, 1, 2, 4) time-integration scheme still performed relatively well. The last column within the table includes the MIS ERK(1,LSRK4-5,1,1) method which uses Algorithm 2.3.2 to advance the slow and fast processes with forward Euler and LSRK4-5, respectively, with no substepping or extrapolation. This MR scheme performed better compared to both the extrapolated MIS method and the SR LSRK4-5; however, it should be noted that the MR extrapolation methods maintain 4^{th} order accuracy for all three processes, whereas the MIS ERK(1,LSRK4-5,1,1) method only maintains 4^{th} order accuracy in time for the fast process. We use this same approach within our extrapolated MR method when considering splitting option two from Section 4.1.2.

In terms of the shallow water equations, it may not be realistic to use higher than a 4^{th} or 6^{th} order Laplacian; however, if we consider NUMA's 2D SWE model as a three process PDE test for determining how well our MR method performs compared to an efficient single-rate method, then we can increase the order of the Laplacian operator arbitrarily large in order for it to represent a third process (i.e., diffusion) within the model that must be resolved, yet is computationally expensive to compute. Table 4.2.2 displays the results when K is increased to 50, which is equivalent to a 100^{th} order Laplacian operator. Comparing the results from both tables, we now notice a significant improvement within the extrapolated multirate methods, such that the MIS ERK(1, 1, m , 4) method for $m \in \{20, 50\}$ outperformed the LSRK4-5 method; however the MIS ERK(1,LSRK4-5,1,1) method remained the most efficient. As previously mentioned, this is not a fair comparison as all methods within the table maintain 4^{th} order accuracy for all processes within the 2D SWE model except for the MIS ERK(1,LSRK4-5,1,1) method. Although we only demonstrate our method for a few values of m , it is easily seen within both Tables 4.2.1 and 4.2.2 that as m increases, the total number of slow RHS evaluations (i.e., diffusion count) decreases.

Table 4.2.2. 4th Order Comparison Using the MIS ERK(1, 1, m , 4) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 50$ and $\nu = 0.001$ Using Splitting Option One.

SR LSRK4-5 versus MIS ERK(1, 1, m , 4) Method for $m \in \{2, 10, 20, 50, 100\}$ and $h_B = 100$, $K = 50$, $\nu = 0.001$							
Time-Integration Method	LSRK4-5	MIS ERK (1,1,2,4)	MIS ERK (1,1,10,4)	MIS ERK (1,1,20,4)	MIS ERK (1,1,50,4)	MIS ERK (1,1,100,4)	MIS ERK (1,LSRK4-5,1,1)
Max Δt	0.0453	0.056	0.097	0.147	0.258	0.385	0.0453
Diffusion Count	11250	18000	10500	7000	4000	3000	2250
Advection Count	225	720	2100	2800	4000	6000	225
Gravity Count	225	720	2100	2800	4000	6000	225
Total RHS Evaluations	11700	19440	14700	12600	12000	2700	13674
Max Courant	1.66	2.05	3.56	5.39	9.46	14.14	1.66
Wall Clock Time (sec)	6.39	9.80	7.94	6.11	5.89	6.70	2.06
RHS Comp Time (sec)	5.03	8.44	6.30	4.89	4.41	5.30	1.14
Loss of Mass	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.86E-16
S_{comp}/F_{comp}	—	-0.15	5.00	1.21	1.04	1.40	0.00

However, the reduction in slow (diffusion) RHS evaluations is directly related to the increase in the total number of fast (advection plus gravity) RHS evaluations. Therefore, simply reducing the number of slow evaluations is only beneficial if the cost of applying the slow process is more costly than each time the method must apply the fast process; hence our justification for increasing the order of the Laplacian operator for testing purposes, and why we compute the S_{comp}/F_{comp} ratio. Notice that for positive computational ratios close to one, the MR methods either outperform the SR method, or come very close.

It should be mentioned that although not theoretically proven here, we could numerically generate a sequence of extrapolated MR methods using increasing values of substepping that could be used to determine the appropriate number of substeps to use that would minimize both the total number of slow and fast RHS computations, and be computationally more efficient than the SR method. For example, based on the wall clock times shown in Table 4.2.2, both the MIS ERK(1, 1, 20, 4) and MIS ERK(1, 1, 50, 4) methods are more efficient than LSRK4-5 for the given simulation parameters: $N = 4$, $N_{e_x} = 40$, $N_{e_y} = 40$, $\nu = 0.001$, $K = 50$, and $h_B = 100$. Therefore, we can assume for this simulation that there exists an $\{m \in \mathbb{R}, 20 \leq m \leq 100\}$, such that an optimal MIS ERK(1, 1, m , 4) method can be found that would minimize the total number of RHS evaluations and “theoretically” use the most efficient value of m for substepping the fast process. Let’s now look at the convergence rates for the methods used in Table 4.2.2.

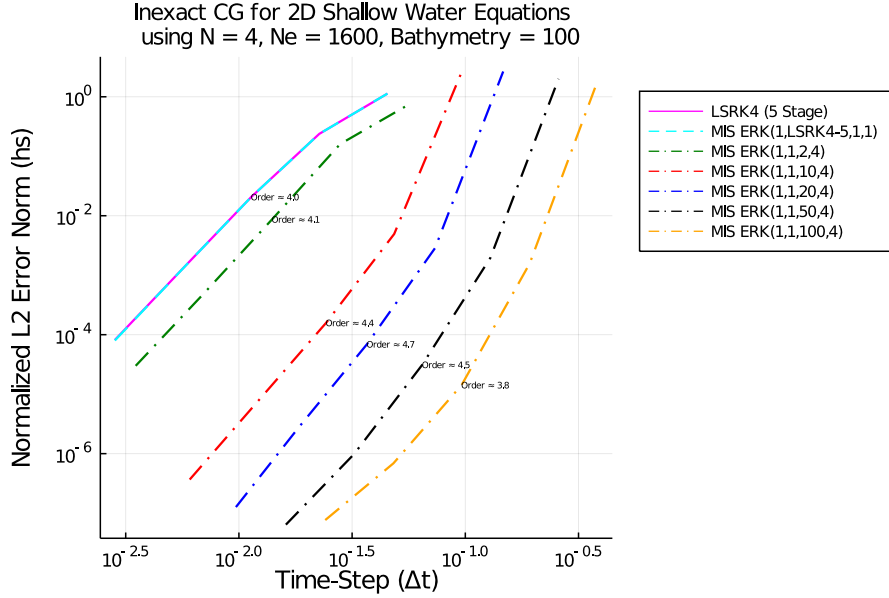


Figure 4.2.1. Convergence Rates for the NUMA 2D Shallow Water Model for RHS Splitting Option 1.

Instead of selecting a test case with a known analytic solution, we use a self-convergence test for the 2D non-linear shallow water equations in NUMA. Therefore, we selected the LSRK4-5 method with $\Delta t = 0.0005$, as our approximate “truth” solution (i.e., $\mathbf{q}_i^{(exact)}$). To generate the convergence rates for h_s displayed in Figure 4.2.1, we compared all MR methods, including LSRK4-5, against the truth solution using the following ℓ^2 norm:

$$\ell^2 = \frac{\sqrt{\sum_{i=1}^{N_p} (h_{s_i}^{(num)} - h_{s_i}^{(exact)})^2}}{\sqrt{\sum_{i=1}^{N_p} (h_{s_i}^{(exact)})^2}}. \quad (4.2.2)$$

Notice that all of the methods approximately achieved 4th order accuracy, which is what we expected. Furthermore, we notice that for higher values of m , the MIS ERK(1, 1, m , 4) method was able to achieve greater numerical accuracy for a given time-step. Note, the Δt we plotted the norms against in Figure 4.2.1 is Δt_s for the slow process.

Let’s now look at a side-by-side comparison of the results for the LSRK4-5 method versus the MIS ERK(1,1,50,4) method for a few snapshots in time. For faster computational time, we computed the previous results using 4th order accuracy in both space and time;

however, the quality of the final plots were not smooth since we only used a total of $(N \cdot N_e + 1)^2 = 25,921$ degrees of freedom (DoF) within a 2D square of dimension $(x, y) \in [-100, 100]^2$. Therefore, for a smoother approximated solution, we increased the order of the interpolating polynomial to 8th order using a total of 80 elements in a $[-200, 200]^2$ square for a total of 410,881 degrees of freedom. We also increased the order of the Laplacian operator to 150th order, the bathymetry to $h_B = 1000$ and ran the simulation for time $0 \leq t \leq 3$, in seconds. The results are displayed in Figures 4.2.2 - 4.2.4.

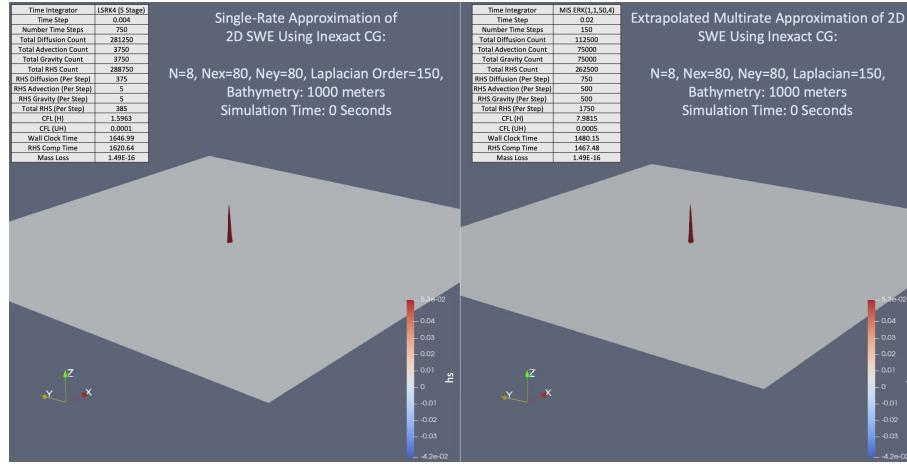


Figure 4.2.2. LSRK4-5 Versus MIS ERK(1,1,50,4) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 0$ Seconds Using Splitting Option One.

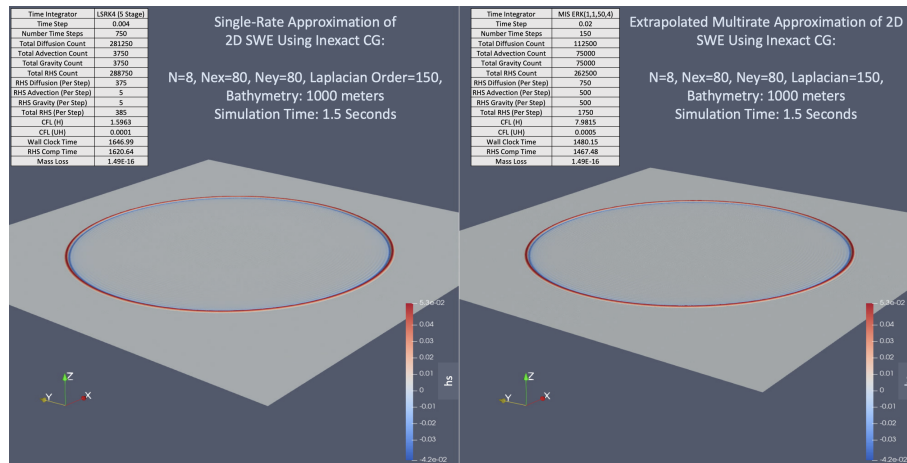


Figure 4.2.3. LSRK4-5 Versus MIS ERK(1,1,50,4) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 1.5$ Seconds Using Splitting Option One.

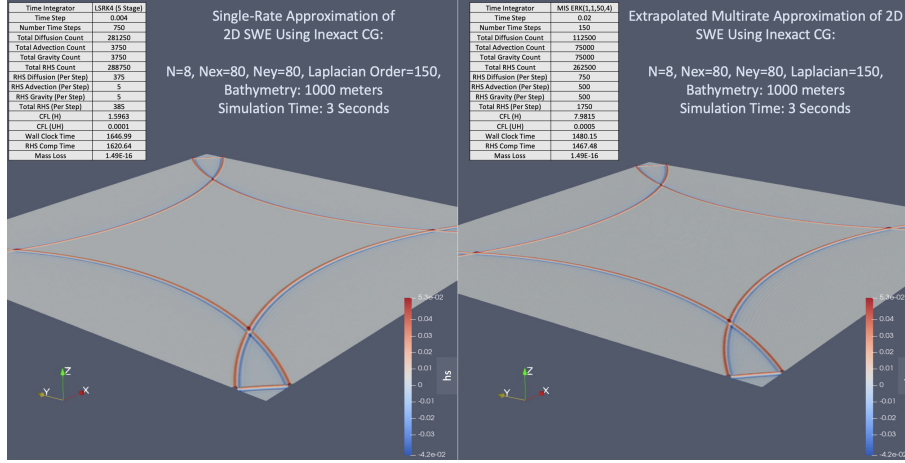


Figure 4.2.4. LSRK4-5 Versus MIS ERK(1,1,50,4) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 3$ Seconds Using Splitting Option One.

Comparing the two methods we find that the MIS ERK(1,1,50,4) method required approximately 153 seconds (2.55 min.) less time to approximate the solution to the same level of accuracy as the SR LSRK4-5 method over the three second simulation timeline. The length of simulation time chosen ensured the wave generated by the initial water column was able to reach the spatial boundaries and reflect back into the spatial domain.

Let's now turn to splitting option two in Section 4.1.2, such that all three processes are advanced in time using three different time-steps, such that $\Delta t_{slow} = m_1 \Delta t_{medium} = m_1 m_2 \Delta t_{fast}$, where m_1 represents the substepping rate between the slow (diffusion) process and the medium (advection) process. Similarly, m_2 is the substepping rate between the medium process and the fast (gravity) process. Note, since our model is now split into three processes, Algorithm 3.3.1 must be slightly modified to include an additional substepping routine for the third process, which we do not show here, but make mention of for reference.

We now use the following notation, MIS ERK(I, J, K, m_1, m_2, p), to represent our extrapolated MIS method, such that the slow, medium and fast processes are advanced in time using three different choices for I, J , and K . For simplicity, we will assume that all three processes only use forward Euler; however, we will use various combinations of the m_1 substepping between the diffusion and advection processes; however, we have fixed $m_2 = 2$ between advection and gravity. Finally, the entire method is extrapolated to order $p = 4$.

Table 4.2.3. 4^{th} Order Comparison Using the MIS ERK(1, 1, 1, m_1 , 2, 4) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 75$ and $\nu = 0.001$ Using Splitting Option Two.

LSRK4-5 vs. MIS ERK(1, 1, 1, m_1 , 2, 4) for $m_1 \in \{2, 10, 20, 25, 50, 100\}$, $h_B = 100$, $K = 75$, and $\nu = 0.001$							
Time-Integration Method	LSRK4-5	MIS ERK (1,1,1,2,2,4)	MIS ERK (1,1,1,10,2,4)	MIS ERK (1,1,1,20,2,4)	MIS ERK (1,1,1,25,2,4)	MIS ERK (1,1,1,50,2,4)	MIS ERK (1,1,1,100,2,4)
Max Δt	0.0453	0.065	0.115	0.173	0.198	0.305	0.452
Diffusion Count	16875	23250	13500	9000	8250	5250	3750
Advection Count	225	620	1800	2400	2750	3500	5000
Gravity Count	225	1240	3600	4800	5500	7000	10000
Total RHS Evaluations	17325	25110	18900	16200	16500	15750	18750
Max Courant	1.66	2.39	4.22	6.35	7.26	11.21	16.59
Wall Clock Time (sec)	8.62	12.31	10.48	8.92	9.20	9.24	10.98
RHS Comp Time (sec)	7.35	11.00	8.84	7.71	7.94	7.80	9.56
Loss of Mass	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.86E-16	0.00E+00
S_{comp}/F_{comp}	—	-0.22	1.47	0.86	0.90	0.86	1.11

Table 4.2.3 shows the 2D SWE results, where 4^{th} order interpolation is used within 40 elements per direction and the numerical solutions are advanced in time for $0 \leq t \leq 2$. From these results we notice that the MIS ERK(1, 1, 1, m_1 , 2, 4) method is comparable to LSRK4-5 for a few of the m_1 choices used. It is likely that there is a more optimal choice of m_1 not tried; however, it should be noted that the current implementation of our extrapolated multirate method into NUMA does not take advantage of the most efficient extrapolation algorithm as discussed in Chapter 3. Therefore, we feel positive with the results shown in Table 4.2.3.

We previously mentioned how we used the MIS ERK(1,LSRK4-5,1,1) scheme to advance both advection and gravity with 4^{th} order accuracy, while only maintaining 1^{st} order accuracy for diffusion. This is not an unreasonable thing to do, as the diffusion process is of little interest to us and our multirate method is not limited to using a specific method or order of accuracy per process. However, single-rate methods only have the option to advance all processes at the same rate and order of accuracy. Therefore, we will now look at one final example for solving the 2D SWE model with our extrapolated MIS method, such that we will use an iterative MR approach.

For example, we will use two extrapolated MIS ERK(I, J, m, p) methods as follows:

$$\text{Outer MIS Method: MIS ERK}(I_1, J_1, m_1, p_1), \quad (4.2.3a)$$

$$\text{Inner MIS Method: MIS ERK}(I_2, J_2, m_2, p_2) = J_1, \quad (4.2.3b)$$

such that the overall method is:

$$\text{MIS ERK}(I_1, \text{MIS ERK}(I_2, J_2, m_2, p_2), m_1, p_1). \quad (4.2.4)$$

Notice that we now have seven different parameters to choose where I_1 , I_2 and J_2 represent the ERK methods used to advance the slow, medium and fast processes respectively. For simplicity, we will set $I_1 = I_2 = J_2$, such that all three processes will be advanced using forward Euler. Additionally, we will set $p_1 = 1$ so that the extrapolation process is only performed within the inner MIS method. Finally, we fix the inner substepping ratio between advection and gravity using $m_2 = 2$, and generate a 4th order extrapolated approximation for the medium and fast processes using $p_2 = 4$; therefore, our only free parameter is m_1 .

Table 4.2.4. Single-rate Versus Multirate Comparison Using the MIS ERK(1,MIS ERK(1,1,2,4), m_1 ,1) Method to Solve NUMA's 2D SWE Model for $h_B = 100$, $K = 20$ and $\nu = 0.001$ Using Splitting Option Two.

LSRK4-5 vs. MIS ERK(1, J_1 , m_1 ,1) for $m_1 \in \{2, 6, 12\}$, $J_1 = \text{MIS ERK}(1,1,2,4)$, $h_B = 100$, $K = 5$, and $\nu = 0.001$							
Time-Integration Method	LSRK4-5	MIS (1, J_1 ,2,1)	MIS ERK (1,LSRK4-5,2,1)	MIS (1, J_1 ,6,1)	MIS ERK (1,LSRK4-5,6,1)	MIS (1, J_1 ,12,1)	MIS ERK (1,LSRK4-5,12,1)
Max Δt	0.0453	0.089	0.09	0.267	0.27	0.54	0.54
Diff. Count	1125	115	115	40	40	20	20
Advec. Count	225	460	230	480	240	480	240
Gravity Count	225	920	230	960	240	960	240
Total RHS Evaluations	1575	1495	575	1480	520	1460	500
Max Courant	1.66	3.26	3.29	9.79	9.90	19.80	19.80
Wall Clock Time (sec)	2.23	1.91	1.10	1.86	1.26	1.84	1.07
RHS Comp Time (sec)	0.65	0.95	0.19	0.97	0.16	0.95	0.15
Loss of Mass	0.00E+00	0.00E+00	1.86E-16	0.00E+00	1.86E-16	0.00E+00	1.86E-16
S_{comp}/F_{comp}	—	0.92	0.01	0.91	0.03	0.90	0.03

Table 4.2.4 shows the results comparing the iterative extrapolated MIS method in (4.2.4) against both the LSRK4-5 and MIS ERK(1,LSRK4-5, m ,1) methods, where we set $m = m_1$ to ensure a fair comparison between the two multirate methods. In review of the results, we find that all of the multirate methods performed very well; however, as previously mentioned, the extrapolation scheme we use in NUMA is not the most efficient implementation. Therefore, although the results we achieved for the MIS ERK(1,MIS ERK(1,1,2,4), m_1 ,1) methods are good, we could further improve this method, whereas there is no further improvements to make for either the SR LSRK4-5 or MIS(1,LSRK4-5, m ,1) methods. It should also be made clear that the primary difference between the MIS ERK(1,LSRK4-5, m ,1) and MIS ERK(1,MIS ERK(1,1,2,4), m_1 ,1) methods is that the former method advances both the advection and gravity using the SR LSRK4-5 method, whereas the later approximates the slow and fast processes using our 4th order MIS extrapolation scheme. Otherwise, they are equivalent regarding their treatment of the slow process and amount of substepping between diffusion and the other two processes within the model.

We complete the study of our extrapolated MR method for the 2D SWE model by comparing the convergence rates, found in Figure 4.2.5, and then display the simulation results comparing the LSRK4-5 method against one of the iterative extrapolated MIS methods.

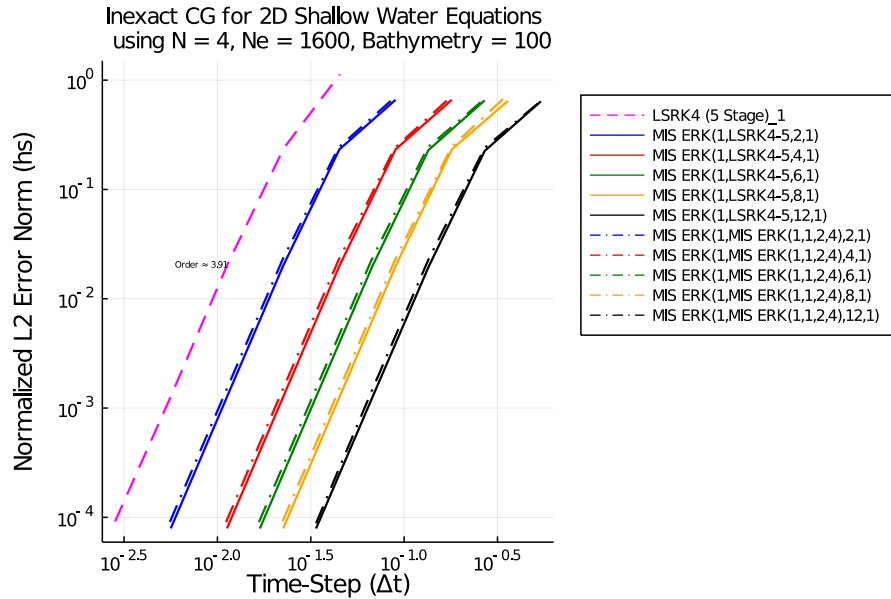


Figure 4.2.5. Convergence Rates for the NUMA 2D Shallow Water Model for RHS Splitting Option Two.

Although not shown in the table, we included the convergence results for outer substepping values of $m = m_1 = 4$ and 8, where Figure 4.2.5 shows that we achieve 4^{th} order accuracy for all methods used in Table 4.2.4 to approximate the 2D SWE model using splitting option two. It is important to mention that although we decided to not maintain 4^{th} order accuracy on the slow diffusion process, we still achieved 4^{th} order accuracy for each method because the artificial viscosity coefficient selected is very small.

We will see in the next section how increasing the viscosity parameter will influence the order of accuracy of these methods when solving both the 2D and 3D Euler (rising thermal bubble) model. Finally, Figures 4.2.6 - 4.2.8 compare the LSRK4-5 method versus the MIS ERK(1,MIS ERK(1,1,2,4),2,1) method using 8^{th} order interpolation polynomials within 40 elements in each direction. The simulation used a 40^{th} order Laplacian operator, bathymetry constant of $h_B = 100$ and time of $0 \leq t \leq 4$, in seconds.

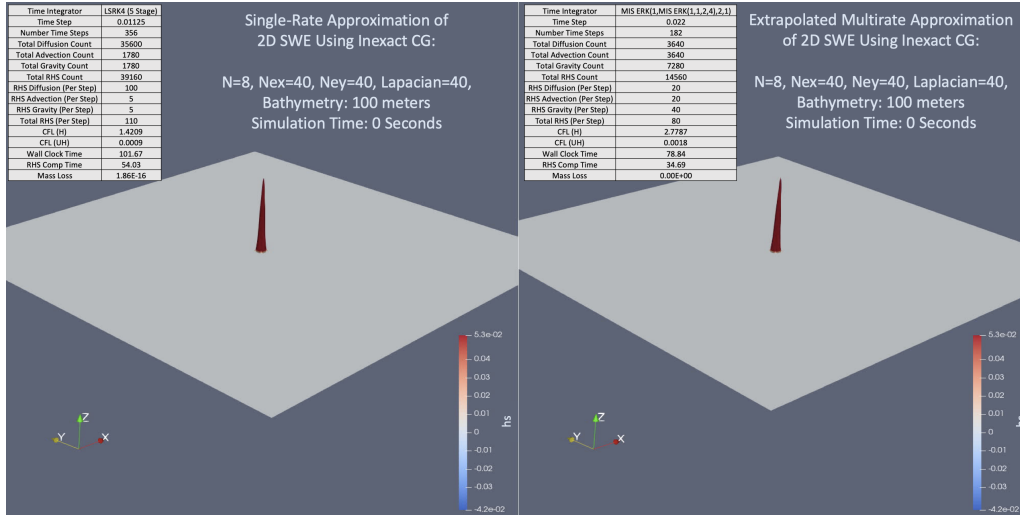


Figure 4.2.6. LSRK4-5 Versus MIS ERK(1,MIS ERK(1,1,2,4),2,1) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 0$ Seconds Using Splitting Option Two.

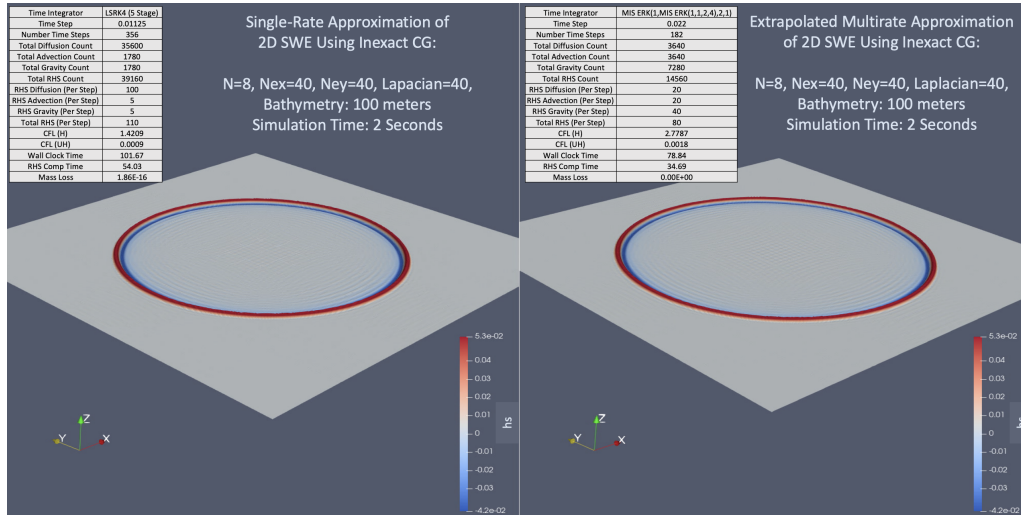


Figure 4.2.7. LSRK4-5 Versus MIS ERK(1,MIS ERK(1,1,2,4),2,1) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 2$ Seconds Using Splitting Option Two.

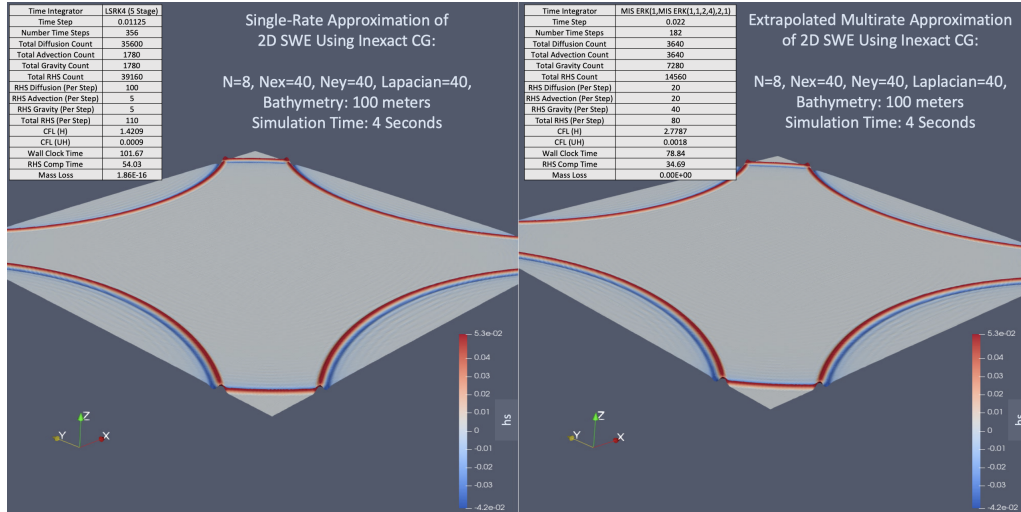


Figure 4.2.8. LSRK4-5 Versus MIS ERK(1,MIS ERK(1,1,2,4),2,1) Surface Height (h_s) Field for NUMA 2D SWE Model at $t = 4$ Seconds Using Splitting Option Two.

Comparing the two methods we find that the MIS ERK(1,MIS ERK(1,1,2,4),2,1) method required approximately 23 seconds less time to approximate the solution to the same level of accuracy as the SR LSRK4-5 method over the four second simulation timeline. Once

again, the length of simulation time chosen ensured the wave generated by the initial water column was able to reach the spatial boundaries and reflect back into the spatial domain. We now consider how these methods perform when approximating the solutions to the nonhydrostatic Euler equations.

4.3 Governing Equations for the Nonhydrostatic Navier-Stokes (Euler) Equations

The Euler equations can describe both incompressible and compressible flow, such as air flow over a wing or the rise of a thermal air bubble within the Earth's atmosphere. Although these equations may take various forms, we will consider the compressible Euler equations, which we have chosen to write as follows (see [54]):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \quad (4.3.1a)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla P + g \hat{\mathbf{r}} + f \hat{\mathbf{r}} \times \mathbf{u} = \nabla \cdot (\nu \nabla(\mathbf{u})), \quad (4.3.1b)$$

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \nabla \cdot (\nu \nabla(\theta)). \quad (4.3.1c)$$

The estimated variables of interest are $(\rho, \mathbf{u}^\top, \theta)^\top$, such that ρ is the density, $\mathbf{u} = (u, v, w)^\top$ is the Cartesian velocity field, θ is the potential temperature, $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})^\top$ is the three-dimensional gradient operator, $\hat{\mathbf{r}} = (r_x, r_y, r_z)^\top$ is the unit vector pointing in the radial direction, f is the Coriolis parameter, g is the gravity constant, and $\mathbf{0} \in \mathbb{R}^d$ is the zero-vector where $d = \dim(\mathbf{u})$. Similar to the 2D SWE problem from Section 4.1, we have added an additional diffusion process to both the momentum (4.3.1b) and potential temperature (4.3.1c) equations, where ν is the artificial viscosity coefficient. The superscript $\top$ indicates the transpose operator.

Similar to [54], we solve the above model in mesoscale mode (i.e., flow in a box) such that $\hat{\mathbf{r}} = \hat{\mathbf{k}}$, which is the unit vector in the z -direction. The momentum equation in (4.3.1b) also contains the pressure P , which is defined as follows:

$$P = P_A \left(\frac{\rho R \theta}{P_A} \right)^\gamma, \quad (4.3.2)$$

where P_A represents the atmospheric pressure at the ground and R is the ideal gas constant. Since we are solving a hyperbolic PDE system within a 2D or 3D box, mass is not allowed to leave the computational domain (i.e., conservation of mass). Therefore, in addition to calculating the computational efficiency and accuracy of our methods, we also compute the total mass lost for each numerical simulation in order to measure if our overall spatial and temporal discretization schemes are conservative.

We use the same boundary conditions within our NUMA model as we did within our 1D and 2D SWE models, such that no-flux boundaries are enforced. This condition ensures fluid flow is constrained within the spatial domain and that no resistance due to viscosity is allowed along the surface boundaries. Specifically, the normal component of the velocity field is set to zero along all physical boundaries, such that $\hat{\mathbf{n}} \cdot \mathbf{u} = 0$. We will also assume the system is non-rotational, such that we can disregard Coriolis effects.

Similar to the 2D SWE model, we will consider two different RHS splitting options for Equations (4.3.1); however, both splitting options will separate all three processes, such that the primary difference between the two options is how we choose to split the advective and acoustic terms. We will represent all of the state variables compactly, such that $\mathbf{q} = (\rho, \mathbf{u}, \theta)^\top$, where the three process PDE system in splitting option one is:

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{S}(\mathbf{q}) + \mathbf{R}(\mathbf{q}) + \mathbf{L}(\mathbf{q}). \quad (4.3.3)$$

4.3.1 Splitting Option One:

This first option uses a hydrostatically balanced reference state to linearize the PDE system such that we subtract and then add the linearized terms to all three equations within (4.3.1). The slowest (diffusion) process is:

$$\mathbf{S}(\mathbf{q}) = \begin{pmatrix} 0 \\ \nu \nabla^{2K}(\mathbf{u}) \\ \nu \nabla^{2K}(\theta) \end{pmatrix}. \quad (4.3.4)$$

Because viscosity is not considered part of the physical system, but rather added as a stabilization mechanism, we expect the viscosity coefficient to be small.

Although we split the diffusion process into its own RHS function, splitting option one does not purely split the medium (advection) process from the fast (acoustic) process. Instead, both the $\mathbf{R}(\mathbf{q})$ and $\mathbf{L}(\mathbf{q})$ terms contain advective and acoustic processes. We will refer to the $\mathbf{R}(\mathbf{q})$ term as the medium (residual) process as it contains all of the non-linear terms minus the linearized terms, such that $\mathbf{R}(\mathbf{q}) = \mathbf{N}(\mathbf{q}) - \mathbf{L}(\mathbf{q})$, where:

$$\mathbf{R}(\mathbf{q}) = - \begin{pmatrix} \nabla \cdot (\rho \mathbf{u}) - L_\rho \\ \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla P + g \hat{\mathbf{k}} - L_{\mathbf{u}} \\ \mathbf{u} \cdot \nabla \theta - L_\theta \end{pmatrix}, \quad (4.3.5)$$

and L_ρ , $L_{\mathbf{u}}$, and L_θ represent the linearized density, momentum and potential temperature, respectively. Now, if we separate the solution into its reference and perturbation states, then we define the scalar variables as follows:

$$\mathbf{q}(\mathbf{x}, t) = \mathbf{q}_0(\mathbf{x}) + \mathbf{q}'(\mathbf{x}, t), \quad (4.3.6)$$

where $\mathbf{q}_0$ represents the hydrostatically balanced fields and $\mathbf{q}'$ is the perturbation from this state. Therefore, the fast (linear) process simply contains the linearized terms just mentioned, such that:

$$\mathbf{L}(\mathbf{q}) = \begin{pmatrix} L_\rho \\ L_{\mathbf{u}} \\ L_\theta \end{pmatrix} = \begin{pmatrix} \nabla \cdot (\rho_0 \mathbf{u}) \\ \frac{1}{\rho_0} \nabla \left[\gamma P_0 \left(\frac{\rho'}{\rho_0} + \frac{\theta'}{\theta_0} \right) \right] \\ \mathbf{u} \cdot \nabla \theta_0 \end{pmatrix}, \quad (4.3.7)$$

where ρ_0 , θ_0 and P_0 are the reference density, potential temperature, and pressure, respectively. Additionally, the perturbed density and potential temperature are ρ' and θ' , respectively.

4.3.2 Splitting Option Two:

For splitting option two we consider a similar three process PDE system:

$$\frac{\partial \mathbf{q}}{\partial t} = \mathbf{S}(\mathbf{q}) + \mathbf{N}_1(\mathbf{q}) + \mathbf{N}_2(\mathbf{q}), \quad (4.3.8)$$

where there is no change to the slowest (diffusion) process $\mathcal{S}(\mathbf{q})$; however, the other two RHS terms within this option are split more accurately in terms of the advective and acoustic processes, respectively. For example, the medium (advection) process is:

$$\mathbf{N}_1(\mathbf{q}) = - \begin{pmatrix} \mathbf{u} \cdot \nabla \rho \\ \mathbf{u} \cdot \nabla \mathbf{u} \\ \mathbf{u} \cdot \nabla \theta \end{pmatrix}, \quad (4.3.9)$$

and the fast (acoustic) process is:

$$\mathbf{N}_2(\mathbf{q}) = - \begin{pmatrix} \rho \nabla \cdot \mathbf{u} \\ \frac{1}{\rho} \nabla P + g \hat{\mathbf{k}} \\ 0 \end{pmatrix}. \quad (4.3.10)$$

Note, we chose to represent both of these RHS functions using $\mathbf{N}_i(\mathbf{q})$ since each of them contain non-linear terms, respectively.

4.4 NUMA 2D Euler Results

The 2D Euler (rising thermal bubble) simulation, as described by [63], “shows the evolution of a warm bubble in a constant potential temperature environment. Because the bubble is warmer than the ambient air it rises while deforming as a consequence of the shearing motion caused by the velocity field gradients until it forms a mushroom cloud.” We use the same initial conditions as [63], where the air mass is initially at rest in hydrostatic balance and the motion of air flow is initiated by adding the following potential temperature perturbation:

$$\theta' = \begin{cases} 0 & \text{for } r > r_c, \\ \frac{\theta_c}{2} \left[1 + \cos \left(\frac{\pi_c r}{r_c} \right) \right] & \text{for } r \leq r_c, \end{cases} \quad (4.4.1)$$

where $\theta_c = 0.5^\circ\text{C}$, π_c is the trigonometric constant, and $r = \sqrt{(x - x_c)^2 + (z - z_c)^2}$ for the following constants: $\theta_0 = 300$ Kelvin (K), $r_c = 250$ meters (m), $(x_c, z_c) = (500, 350)$ m, and $(x, z) \in [0, 1000]^2$ m, and time of solution $t \in [0, 700]$ seconds. Note, from Equation

(4.3.6) we split the density, potential temperature and pressure terms as follows:

$$\rho(\mathbf{x}, t) = \rho_0(z) + \rho'(\mathbf{x}, t), \quad (4.4.2a)$$

$$\theta(\mathbf{x}, t) = \theta_0(z) + \theta'(\mathbf{x}, t), \quad (4.4.2b)$$

$$P(\mathbf{x}, t) = P_0(z) + P'(\mathbf{x}, t), \quad (4.4.2c)$$

such that hydrostatic balance is defined as:

$$\frac{d\rho_0}{dz} = -\frac{g}{c_p\theta_0}, \quad (4.4.3)$$

and c_p is the specific heat for constant pressure. Additionally, we set the artificial viscosity constant, $\nu = 1$, use a 2^{nd} order Laplacian (i.e., $K = 1$) for all simulations. Finally, the spatial derivatives are discretized using an inexact CG method with $N = 4$ in 20 elements per direction.

Since both splitting options represent a three process PDE system, we will approximate the 2D Euler equations using the following time integration schemes for both splitting options one and two:

1. 4^{th} Order Single-rate Method (All Processes):
 - (a) LSRK4-5
2. 1^{st} Order (Diffusion), 4^{th} Order (Advection and Acoustics) Multirate Methods:
 - (a) MIS ERK(1,MIS ERK(1,LSRK4-5,6,1),5,1)
 - (b) MIS ERK(1,MIS ERK(1,1,2,4),10,1)
 - (c) MIS ERK(1,LSRK4-5,10,1)
3. 4^{th} Order Extrapolated Multirate Methods (All Processes) :
 - (a) MIS ERK(1,1,10,4)
 - (b) MIS ERK(1,1,1,10,2,4)
 - (c) MIS ERK(1,1,1,1,1,4)

Table 4.4.1 displays the results for approximating the 2D Euler equations using the seven different time-integration methods just listed when the RHS splitting option one from Section 4.3.1 is used.

Table 4.4.1. Single-rate Versus Multirate Comparison for NUMA's 2D Euler Model for $K = 1$ and $\nu = 1$ Using Linearized Splitting Option One.

$J_1 = \text{MIS ERK}(1, \text{LSRK4-5}, 6, 1)$, $J_2 = \text{MIS ERK}(1, 1, 2, 4)$, $J_3 = \text{LSRK4-5}$, $K = 1$, and $\nu = 1$							
Time-Integration Method	LSRK4-5	MIS ERK (1, J_1 , 5, 1)	MIS ERK (1, J_2 , 10, 1)	MIS ERK (1, J_3 , 10, 1)	MIS ERK (1, 1, 10, 4)	MIS ERK (1, 1, 1, 10, 2, 4)	MIS ERK (1, 1, 1, 1, 1, 4)
Max Δt	0.04	1.22	0.5	0.4	0.045	0.069	0.034
Diff. Count	87500	574	1400	1750	155560	101450	205890
Advec. Count	87500	2870	140000	87500	1555600	1014500	205890
Gravity Count	87500	86100	280000	87500	1555600	2029000	205890
Total RHS Evaluations	262500	89544	421400	176750	3266760	3144950	617670
Max Courant	1.61	49.09	20.12	16.10	1.81	2.78	1.37
Wall Clock Time (sec)	31.50	14.36	79.07	22.78	403.66	479.03	117.22
RHS Comp Time (sec)	28.93	11.85	76.58	20.36	401.17	476.43	114.56
Loss of Mass	2.09E-16	2.09E-16	4.18E-16	2.09E-16	1.88E-15	3.34E-15	6.27E-16
$S_{\text{comp}}/F_{\text{comp}}$	—	0.99	2.85	0.00	-43.14	-205.63	-2.00

*Note, all NUMA simulations for the 2D Euler model were computed on the Naval Postgraduate School's Hamming supercomputer using 20 cores on AMD Epyc processors.

From the results in Table 4.4.1 we clearly see that the multirate methods that do not extrapolate all three processes performed better (in terms of wall clock time) than those that do maintain 4^{th} order accuracy for all processes. In the 2D SWE model the MR methods that did not keep 4^{th} order accuracy for all processes still maintained an overall accuracy of 4^{th} order. However, because the artificial viscosity constant is larger in the 2D Euler model, the three MR methods within the enumerated list 2.(a) - 2.(c) no longer maintain 4^{th} order accuracy. Instead, the diffusion process significantly influences their accuracy and rate of convergence. These results can be found in Figure 4.4.1, where Sp.1 and Sp.2 after the time-integration name indicates the splitting option used.

Notice that only the LSRK4-5 and the 4^{th} order extrapolated MIS methods achieved 4^{th} order convergence. Additionally, their overall accuracy was between 10^{-6} and 10^{-10} , whereas the methods that did not maintain 4^{th} order accuracy for all process achieved at best 10^{-4} accuracy and converged on the order $\mathcal{O}(\Delta t)$. It is unclear why the extrapolated MIS method for splitting option one was non-convergent, and we believe that this is not a limitation of the time-integration but a bug or other problem in the NUMA physical splitting. This warrants further evaluation because splitting option one conserves mass with MR methods, whereas splitting option two does not. All methods used in Table 4.4.1 were

also used for splitting option two. Results are displayed in Table 4.4.2.

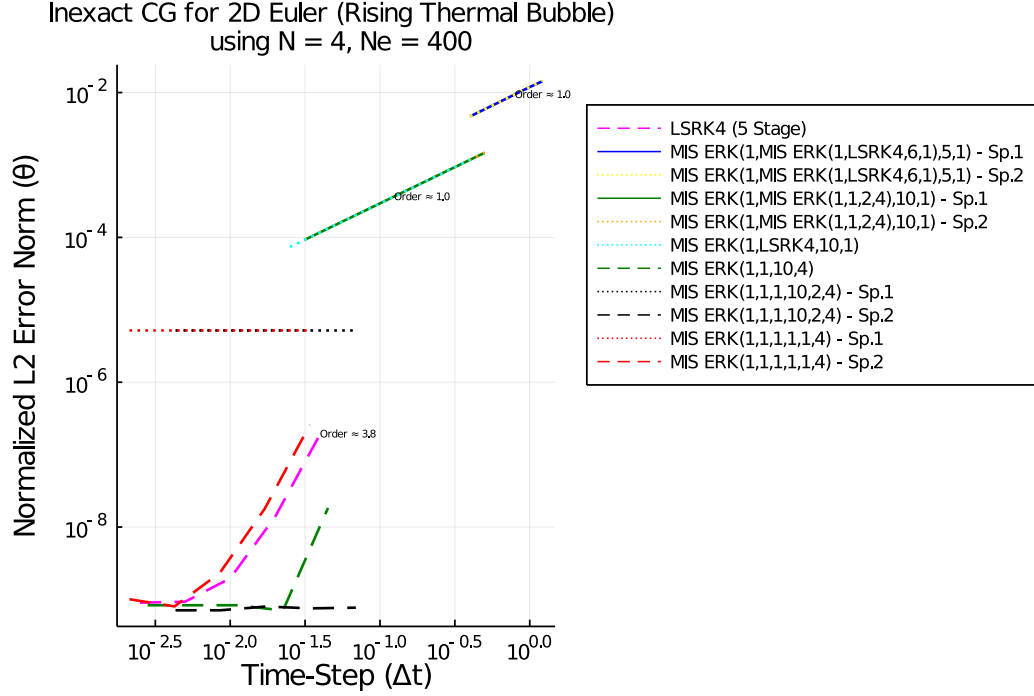


Figure 4.4.1. Convergence Rates for the NUMA 2D Euler (Rising Thermal Bubble) Model for RHS Splitting Option One.

Table 4.4.2. Single-rate Versus Multirate Comparison for NUMA's 2D Euler Model for $K = 1$ and $\nu = 1$ Using Splitting Option Two.

$J_1 = \text{MIS ERK}(1, \text{LSRK4-5}, 6, 1)$, $J_2 = \text{MIS ERK}(1, 1, 2, 4)$, $J_3 = \text{LSRK4-5}$, $K = 1$, and $\nu = 1$							
Time-Integration Method	LSRK4-5	MIS ERK (1, J_1 , 5, 1)	MIS ERK (1, J_2 , 10, 1)	MIS ERK (1, J_3 , 10, 1)	MIS ERK (1, 1, 10, 4)	MIS ERK (1, 1, 1, 10, 2, 4)	MIS ERK (1, 1, 1, 1, 1, 4)
Max Δt	0.04	1.22	0.5	0.4	0.045	0.069	0.034
Diff. Count	87500	574	1400	1750	155560	101450	205890
Advec. Count	87500	2870	140000	87500	1555600	1014500	205890
Gravity Count	87500	86100	280000	87500	1555600	2029000	205890
Total RHS Evaluations	262500	89544	421400	176750	3266760	3144950	617670
Max Courant	1.61	49.09	20.12	16.10	1.81	2.78	1.37
Wall Clock Time (sec)	31.50	18.97	85.97	22.78	403.66	529.59	116.84
RHS Comp Time (sec)	28.93	16.50	83.39	20.36	401.17	526.99	114.23
Loss of Mass	2.09E-16	2.49E-09	7.75E-14	2.09E-16	1.88E-16	3.55E-15	8.36E-16
S_{comp}/F_{comp}	—	0.99	2.85	0.00	-43.14	-205.63	-2.00

We have highlighted in red the multirate methods which lost mass due to splitting option two. Note, the MIS ERK methods in columns four and five do not use splitting option one or two as they only split two ways. The MIS ERK(1,1,1,1,1,4) method in the final column is the same as using SR forward Euler as base method for 4th order extrapolation. Potential temperature simulation results for 2D rising thermal bubble at times $t = 0, 350$ and 700 are displayed in Figures 4.4.2 - 4.4.3 for all methods except the MIS ERK(1,1,1,1,1,4).

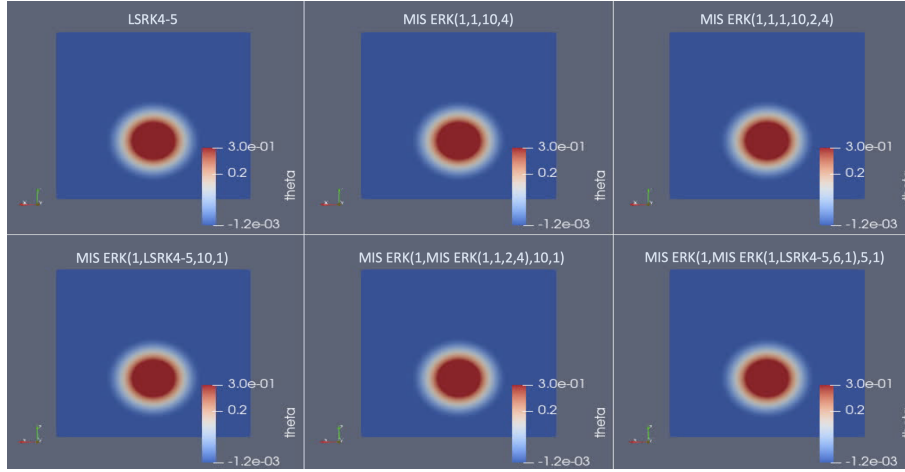


Figure 4.4.2. Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 2D Euler Model at $t = 0$ Seconds Using Splitting Option One.

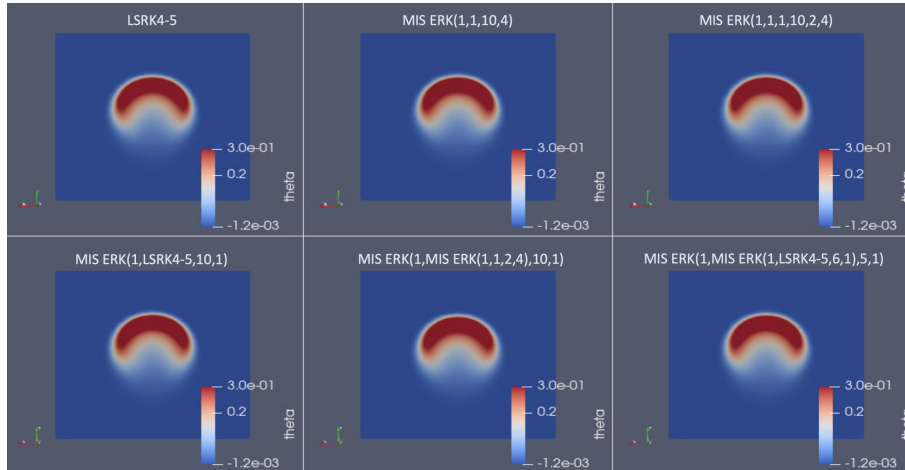


Figure 4.4.3. Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 2D Euler Model at $t = 350$ Seconds Using Splitting Option One.

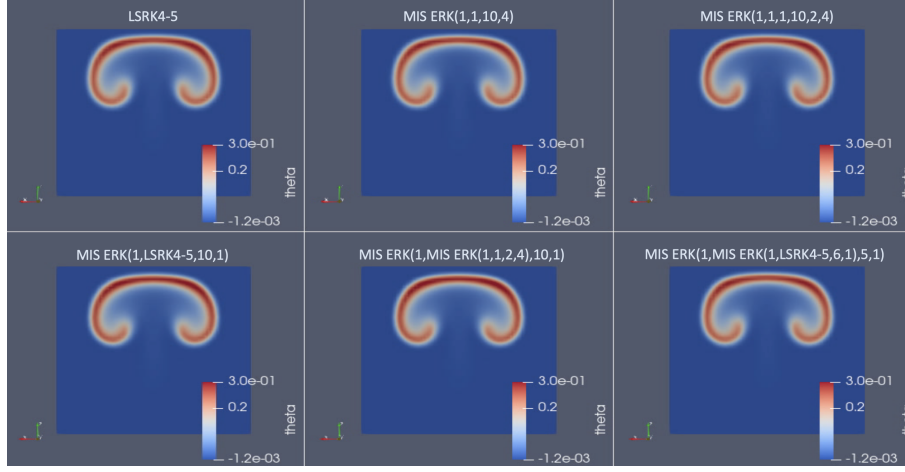


Figure 4.4.4. Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 2D Euler Model at $t = 700$ Seconds Using Splitting Option One.

4.5 NUMA 3D Euler Results

It is important to note that the 3D Euler case is only an extension of the 2D case where the spatial domain is now defined as $(x, y, z) \in [0, 1000]^3$ m. We use the same initial condition shown in Equation (4.4.1); however, $r = \sqrt{(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2}$ and we now use the following constants: $\theta_0 = 300$ K, $\theta_c = 0.5^\circ\text{C}$, $r_c = 250$ m, $(x_c, y_c, z_c) = (500, 500, 350)$ m, and simulation time is now defined for $t \in [0, 500]$ seconds. Additionally, for consistency with the 2D Euler case, the overall spatial discretization scheme uses inexact CG with 4^{th} order interpolating polynomials in 20^3 elements; therefore, the total number of grid points is $(NN_e + 1)^3 = 531,441$.

All observations made within the 2D Euler case were also observed for the 3D Euler model. Convergence rates, tabular simulation data and simulation figures for evolution of the potential temperature field within the 3D rising thermal bubble are shown. Tables 4.5.1 and 4.5.2 display the results for approximating the 3D Euler equations using the same time-integration methods used within the 2D Euler case when RHS splitting option one and two are used, respectively.

Table 4.5.1. Single-rate Versus Multirate Comparison for NUMA's 3D Euler Model for $K = 1$ and $\nu = 1$ Using Linearized Splitting Option One.

$J_1 = \text{MIS ERK}(1, \text{LSRK4-5}, 6, 1)$, $J_2 = \text{MIS ERK}(1, 1, 2, 4)$, $J_3 = \text{LSRK4-5}$, $K = 1$, and $\nu = 1$							
Time-Integration Method	LSRK4-5	MIS ERK (1, J_1 , 5, 1)	MIS ERK (1, J_2 , 10, 1)	MIS ERK (1, J_3 , 10, 1)	MIS ERK (1, 1, 10, 4)	MIS ERK (1, 1, 1, 10, 2, 4)	MIS ERK (1, 1, 1, 1, 1, 4)
Max Δt	0.033	0.99	0.41	0.33	0.036	0.056	0.028
Diff. Count	75760	506	1220	1516	138890	89290	178580
Advec. Count	75760	2530	122000	75800	1388900	892900	178580
Gravity Count	75760	75900	244000	75800	1388900	1785800	178580
Total RHS Evaluations	227280	78936	367220	153116	2916690	2767990	535740
Max Courant	1.33	39.83	16.49	13.28	1.44	2.25	1.13
Wall Clock Time (min)	18.61	8.85	51.10	12.38	237.78	335.21	79.46
RHS Comp Time (min)	16.08	7.84	49.96	11.26	235.13	332.97	75.71
Loss of Mass	0.00E+00	0.00E+00	2.14E-16	2.14E-16	2.14E-16	6.42E-16	0.00E+00
S_{comp}/F_{comp}	—	0.97	2.88	0.00	-41.60	-186.78	-2.00

*Note, all NUMA simulations for the 3D Euler model were computed on the Naval Postgraduate School's Hamming supercomputer using 64 cores on AMD Epyc processors.

Table 4.5.2. Single-rate Versus Multirate Comparison for NUMA's 3D Euler Model for $K = 1$ and $\nu = 1$ Using Linearized Splitting Option Two.

$J_1 = \text{MIS ERK}(1, \text{LSRK4-5}, 6, 1)$, $J_2 = \text{MIS ERK}(1, 1, 2, 4)$, $J_3 = \text{LSRK4-5}$, $K = 1$, and $\nu = 1$							
Time-Integration Method	LSRK4-5	MIS ERK (1, J_1 , 5, 1)	MIS ERK (1, J_2 , 10, 1)	MIS ERK (1, J_3 , 10, 1)	MIS ERK (1, 1, 10, 4)	MIS ERK (1, 1, 1, 10, 2, 4)	MIS ERK (1, 1, 1, 1, 1, 4)
Max Δt	0.033	0.99	0.41	0.33	0.036	0.056	0.028
Diff. Count	75760	506	1220	1516	138890	89290	178580
Advec. Count	75760	2530	122000	75800	1388900	892900	178580
Gravity Count	75760	75900	244000	75800	1388900	1785800	178580
Total RHS Evaluations	227280	78936	367220	153116	2916690	2767990	535740
Max Courant	1.33	39.83	16.49	13.28	1.44	2.25	1.13
Wall Clock Time (min)	18.61	10.24	50.45	12.38	237.78	330.80	75.53
RHS Comp Time (min)	16.08	9.23	49.31	11.26	235.13	328.56	71.69
Loss of Mass	0.00E+00	9.38E-10	1.71E-14	2.14E-16	2.14E-16	4.28E-16	0.00E+00
S_{comp}/F_{comp}	—	0.97	2.88	0.00	-41.60	-186.78	-2.00

Similar to the 2D case, when splitting option two is used, the MIS ERK (1, (MIS ERK(1, LSRK4-5, 6, 1)), 5, 1) and MIS ERK(1, MIS ERK(1, 1, 2, 4), 10, 1) methods are not able to conserve mass; however, the methods which use 4th order extrapolation around all three processes do conserve mass with this splitting option. Convergence rates for all methods are shown in Figure 4.5.1.

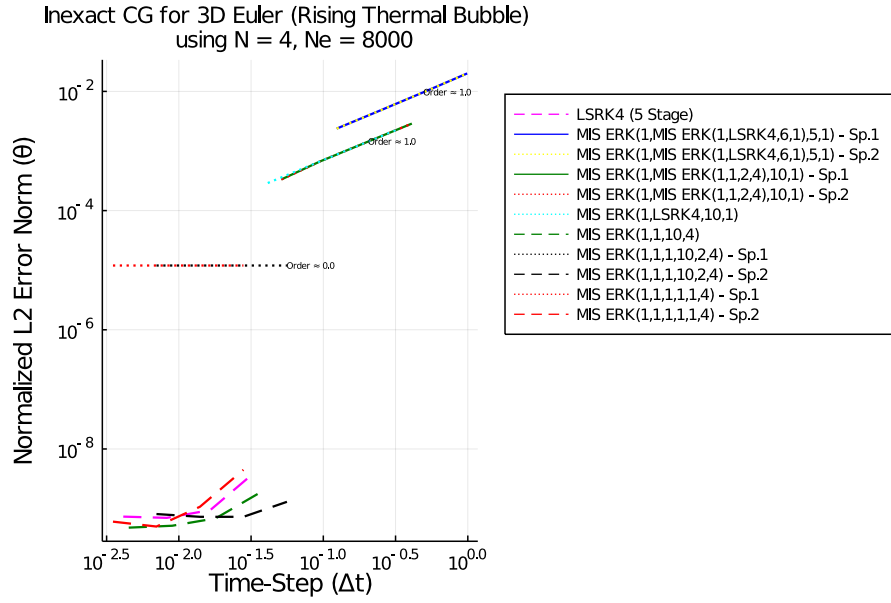


Figure 4.5.1. Convergence Rates for the NUMA 3D Euler (Rising Thermal Bubble) Model for RHS Splitting Option One.

Finally, the potential temperature simulation results for the 3D rising thermal bubble at times $t = 0, 400$ and 500 are displayed in Figures 4.5.2 - 4.5.3.

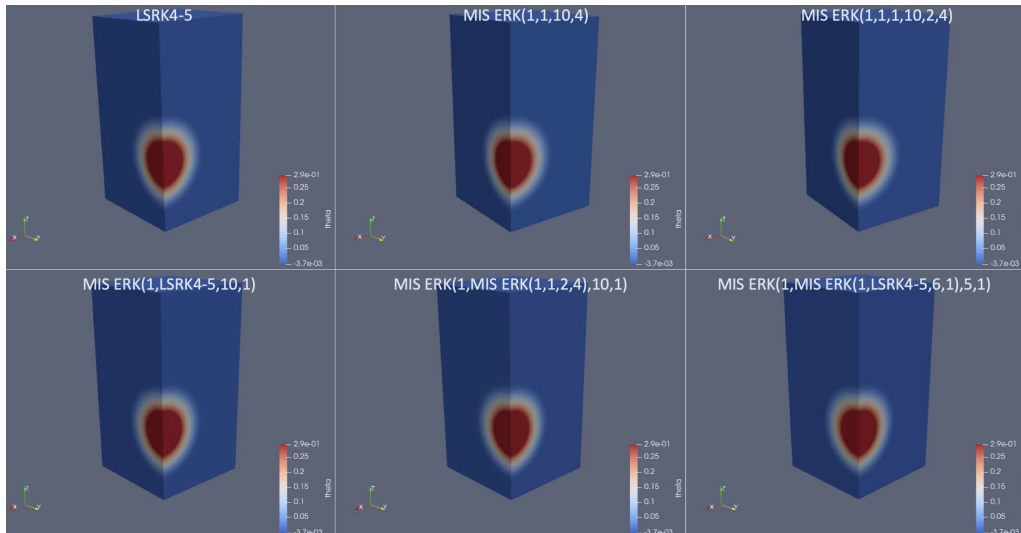


Figure 4.5.2. Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 3D Euler Model at $t = 0$ Seconds Using Splitting Option One.

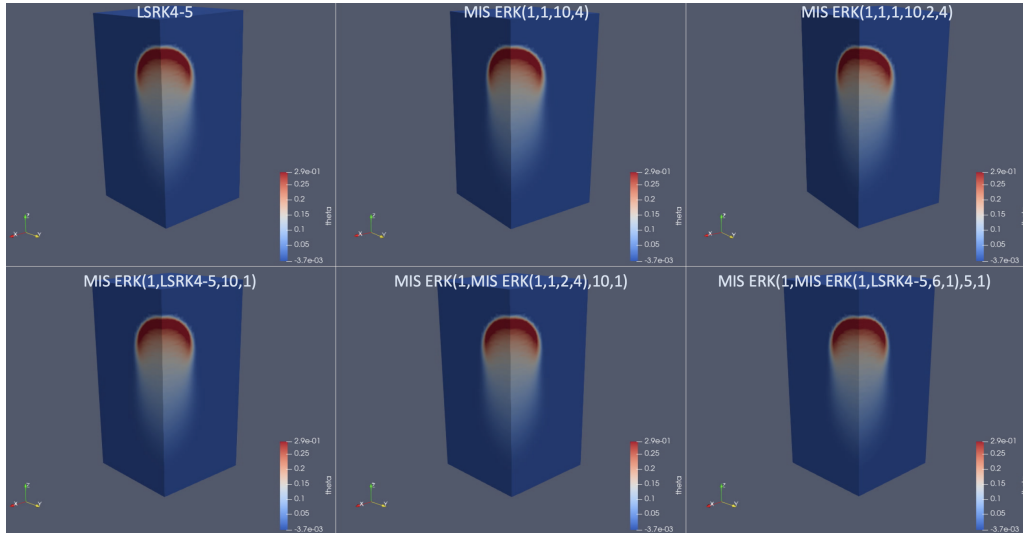


Figure 4.5.3. Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 3D Euler Model at $t = 400$ Seconds Using Splitting Option One.

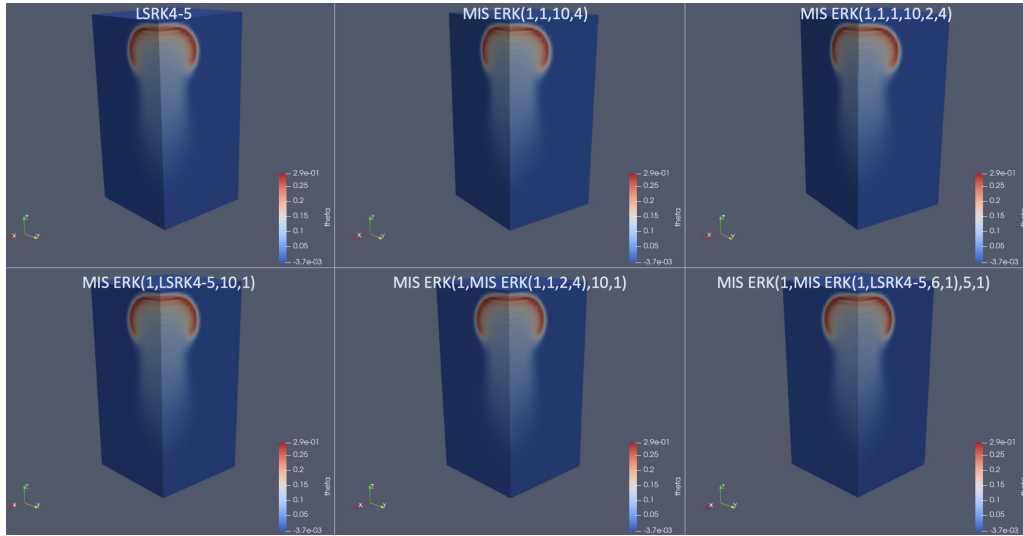


Figure 4.5.4. Single-rate Versus Multirate Potential Temperature (θ) Field for NUMA 3D Euler Model at $t = 500$ Seconds Using Splitting Option One.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:

Conclusion

Within this research we demonstrated how the process of extrapolation for time-integration methods can be applied to both single-rate (SR) and multirate (MR) methods within a general framework that is used with or without substepping. We further highlighted how our multirate extrapolation method, which uses the generalized multirate infinitesimal step (MIS) method as its underlying framework (i.e. base method) allows for a wide range of time-integration options, such as an iterative approach, embedded approach, or combination of the two. We specifically used the shallow water and Euler equations as proxies for the types of problems one should consider when using our MR method, where we found that the extrapolated MIS method is best suited for problems with multiple processes where the slowest process is computationally more expensive than the fastest process within the model. Without loss of generality, since most MR methods that exist are at best 3^{rd} order accurate, our focus was on 4^{th} order schemes (though our extrapolation methods can achieve any order). Furthermore, we demonstrated that our method is particularly useful if one needs high-order accuracy in both time and space. In summary, our MR method showed promising results for the two partial differential equation (PDE) test cases used and we will continue to look for ways to improve the overall efficiency of the method.

For future work we recommend applying the extrapolation process to other MR base methods. Although our initial approach used a generalized MR method, there may exist other MR schemes better suited as base methods for extrapolation. We also suggest that other PDE models be explored in order to widen the range of acceptable problems best suited for using an extrapolated multirate time-integration approach. Additionally, our method would benefit from determining what the optimal number of substeps between different processes should be when all other parameters are fixed. Furthermore, we only considered an explicit approach to our MR extrapolation scheme; therefore, one should look at how the extrapolated MR method would perform when either fully implicit or implicit-explicit (IMEX) methods are introduced. Finally, our method was based on Richardson extrapolation, however, constructing an extrapolated MIS method using Gragg extrapolation would be a major accomplishment if even powers of h can be removed within a MR framework.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:

Butcher Tableau Coefficients for Single-Rate ERK, ARK and LSRK Methods

This appendix contains the coefficients in Butcher tableau form for all single-rate, ERK, Additive ERK (ARK), and low-storage RK (LSRK) methods used within this dissertation.

ERK1 (Forward Euler) Method:

Table A.0.1. Coefficient Tableau for the 1st Order Forward Euler Method

0	0
	1

ERK2 (Midpoint) Method:

Table A.0.2. Coefficient Tableau for the 2nd Order Explicit Midpoint Method

0	
$\frac{1}{2}$	$\frac{1}{2}$
	0 1

ERK3 (Kutta) Method:

Table A.0.3. Coefficient Tableau for the 3rd Order (Kutta) RK Method

0		
$\frac{1}{2}$	$\frac{1}{2}$	
1	-1	2
	$\frac{1}{6}$	$\frac{2}{3}$ $\frac{1}{6}$

ERK4 (Classical) Method:

Table A.0.4. Coefficient Tableau for the 4th Order (Classical) RK Method

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

ERK4 (3/8 Rule) Method:

Table A.0.5. Coefficient Tableau for the 4th Order (3/8 Rule) RK Method

0				
$\frac{1}{3}$	$\frac{1}{3}$			
$\frac{2}{3}$	$-\frac{1}{3}$	1		
1	1	-1	1	
	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

ERK5 (Butcher) Method:

Table A.0.6. Coefficient Tableau for the 5th Order (Butcher) RK Method

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$				
$\frac{1}{2}$	0	$-\frac{1}{2}$	1			
$\frac{3}{4}$	$\frac{3}{16}$	0	0	$\frac{9}{16}$		
1	$-\frac{3}{7}$	$\frac{2}{7}$	$\frac{12}{7}$	$-\frac{12}{7}$	$\frac{8}{7}$	
	$\frac{7}{90}$	0	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$

ODE45 (Dormand-Prince) Method:

Table A.0.7. Coefficient Tableau for the 5th Order (Dormand-Prince) RK Method

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{8}{10}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0

ARK1 Method:

Table A.0.8. Coefficient Tableau for the ARK1 (1st Order, 2-Stage) Method

0	
1	1
	1 0

ARK2 Method by Giraldo/Constantinescu [54]:

Table A.0.9. Coefficient Tableau for the ARK2 (2nd Order, 3-Stage) Method

0			
0.5857864376269049	0.5857864376269049		
1	0.0285954792089683	0.9714045207910316	
	0.3535533905932737	0.3535533905932737	0.2928932188134524

ARK3 Method by Kennedy/Carpenter [55]:

Table A.0.10. Coefficient Tableau for the ARK3 (3^{rd} Order, 4-Stage) Method

0				
0.871733043016918	0.871733043016918			
0.6	0.5275890119763004	0.0724109880236996		
1	0.3990960076760701	-0.4375576546135194	1.0384616469374492	
	0.18764102434672383	-0.595297473576955	0.9717899277217721	0.435866521508459

ARK4 Method by [55]:

Table A.0.11. Coefficient Tableau for the ARK4 (4^{th} Order, 6-Stage) Method

0						
0.5	$\frac{1}{2}$					
0.332	$\frac{13861}{62500}$	$\frac{6889}{62500}$				
0.62	$-\frac{116923316275}{2393684061468}$	$-\frac{2731218467317}{15368042101831}$	$\frac{9408046702089}{11113171139209}$			
0.85	$-\frac{451086348788}{2902428689909}$	$-\frac{2682348792572}{7519795681897}$	$\frac{12662868775082}{11960479115383}$	$\frac{3355817975965}{11060851509271}$		
1	$\frac{647845179188}{3216320057751}$	$\frac{73281519250}{8382639484533}$	$\frac{552539513391}{3454668386233}$	$\frac{3354512671639}{8306763924573}$	$\frac{4040}{17871}$	
	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$

ARK5 Method by [55]:

Table A.0.12. Coefficient Tableau for the ARK5 (5^{th} Order, 8-Stage) Method

0									
0.41	$\frac{41}{100}$								
0.25992958444838016	$\frac{367902744464}{2072280473677}$	$\frac{677623207551}{8224143866563}$							
0.19815048669250362	$\frac{1268023523408}{10340822734521}$	0	$\frac{1029933939417}{13636558850479}$						
0.92	$\frac{14463281900351}{6315353703477}$	0	$\frac{66114435211212}{5879490589093}$	$-\frac{54053170152839}{4284798021562}$					
0.24	$\frac{14090043504691}{34967701212078}$	0	$\frac{15191511035443}{11219624916014}$	$-\frac{18461159152457}{12425892160975}$	$-\frac{281667163811}{9011619295870}$				
0.6	$\frac{19230459214898}{13134317526959}$	0	$\frac{21275331358303}{2942455364971}$	$-\frac{38145345988419}{4862620318723}$	$-\frac{1}{8}$	$-\frac{1}{8}$			
1	$\frac{19977161125411}{11928030595625}$	0	$-\frac{40795976796054}{6384907823539}$	$\frac{177454434618887}{12078138498510}$	$\frac{782672205425}{8267701900261}$	$\frac{69563011059811}{9646580694205}$	$\frac{7356628210526}{4942186776405}$		
	$-\frac{872700587467}{9133579230613}$	0	0	$\frac{22348218063261}{9555858737531}$	$-\frac{1143369518992}{8141816002931}$	$\frac{39379526789629}{19018526304540}$	$\frac{32727382324388}{42900044865799}$	$\frac{41}{200}$	

A.0.1 Low-Storage Runge-Kutta (LSRK) Methods

In addition to ERK and ARK methods, there are also s -stage, m -order, low-storage Runge-Kutta (LSRK) methods specifically designed to limit the increased storage requirements of higher-order RK methods, which typically require $s + 1$ storage registers of size N . Much research has been conducted within this area, with well known resources including the low-storage family of solutions by Williamson (1980) [64], van der Houwen (1977) [65], and Ketcheson (2010) [66], as well as the works by Carpenter-Kennedy (1994) [61] and Niegemann-Diehl-Busch (2012) [67].

We will focus on the two-register of length N (i.e., $2N$), LSRK algorithm originally developed by [64]:

$$\left. \begin{aligned} Q_1^{(1)} &= q^n, \\ Q_2^{(i+1)} &= A_i Q_2^{(i)} + \Delta t R(t^n + c_i \Delta t, Q_1^{(i)}), \\ Q_1^{(i+1)} &= Q_1^{(i)} + B_i Q_2^{(i)}, \\ q^{n+1} &= Q_1^{(s)} \end{aligned} \right\} i = 1, \dots, s. \quad (\text{A.0.1})$$

Similar to ERK methods, LSRK methods can also be written using a modified coefficient tableau, which is arranged into two arrays:

$$\begin{array}{c|c} A_1 & B_1 \\ \vdots & \vdots \\ A_s & B_s \end{array}, \quad (\text{A.0.2})$$

where $A_1 = 0$ for all self-starting methods, and the LSRK coefficients A_j and B_j are determined using the following equations,

$$\begin{aligned} B_j &= a_{j+1,j}, & (j \neq s) \\ B_s &= b_s, \\ A_j &= (b_{j-1} - B_{j-1})/b_j, & (j \neq 1, b_j \neq 0) \\ A_j &= (a_{j+1,j-1} - c_j)/B_j, & (j \neq 1, b_j = 0) \end{aligned} \quad (\text{A.0.3})$$

which provide the relationship $\{\mathbf{c}, A, \mathbf{b}^\top\} \rightarrow \{\mathbf{A}, \mathbf{B}\}$, that maps an ERK's standard Butcher tableau coefficients to its equivalent LSRK coefficients. Additionally, we do not show the

time coefficient values, $\mathbf{c}$, as the condition: $c_i = \sum_{j=1}^{i-1} a_{i,j}$ for $i = 2, \dots, s$, still holds for the LSRK method described in equation (A.0.1). Note, this mapping does not hold for all methods, which means that not every ERK method can be written in low-storage format using only $2N$ registers, as shown by both [64] and [61].

LSRK4-5 Method: In this work, we will compare results using the $2N$ -storage 4^{th} order, 5-stage LSRK scheme developed by [61].

Table A.0.13. Coefficient Tableau for the LSRK4-5 (4^{th} Order, 5-Stage) Method

A	B	C
0.0	0.14965902199922912	0.0
-0.41789047449985195	0.37921031299962726	0.14965902199922912
-1.192151694642677	0.8229550293869817	0.37040095736420475
-1.6977846924715279	0.6994504559491221	0.6222557631344432
-1.5141834442571558	0.15305724796815198	0.9582821306746903

Note, we included column C , which contains the c_i time values for each stage of the method.

APPENDIX B:

Analytic Solution to 1D Shallow Water Equations

This appendix provides the analytic solution to the IBVP within equations (1.1.11) - (1.1.13). First, we will verify that the analytic solution indeed solves the linear PDE. Second, we will “manufacture the solution” to the non-linear PDE. This manufactured solution will then be added into our numerical solution as a source or forcing function to ensure the numerical methods we use accurately solve the IBVP.

Let’s first start by writing the shallow water equations with only linear terms:

$$\frac{\partial}{\partial t} \begin{pmatrix} h_s \\ U \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} U \\ gh_B h_s \end{pmatrix} = \begin{pmatrix} 0 \\ gh_s \frac{\partial h_B}{\partial x} \end{pmatrix}, \quad (\text{B.0.1})$$

where if we assume that $g = h_B = 1$, then equation (B.0.1) becomes:

$$\frac{\partial}{\partial t} \begin{pmatrix} h_s \\ U \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} U \\ h_s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \quad (\text{B.0.2})$$

As mentioned in Chapter 1, the PDE system above is hyperbolic and therefore represents an IBVP that requires both IC and BC, where we will use the following IC:

$$\begin{aligned} h_s(x, 0) &= \frac{1}{2} \cos(c\pi x), \\ U(x, 0) &= 0, \end{aligned} \quad (\text{B.0.3})$$

and no flux BC at the endpoints of our spatial domain:

$$U(0, t) = U(1, t) = 0. \quad (\text{B.0.4})$$

Note that the analytic solution to equations (B.0.1) - (B.0.4) is:

$$\begin{aligned} h_s(x, t) &= \frac{1}{2} \cos(c\pi x) \cos(c\pi t), \\ U(x, t) &= \frac{1}{2} \sin(c\pi x) \sin(c\pi t), \end{aligned} \quad (\text{B.0.5})$$

where c in Equations B.0.3 and B.0.5 is constant and represents the wave frequency. In order to verify that these solutions satisfy the IBVP we simply substitute their respective derivatives into equation (B.0.2) where:

$$\begin{aligned}\frac{\partial h_s}{\partial t} &= -\frac{c\pi}{2} \cos(c\pi x) \sin(c\pi t), \\ \frac{\partial U}{\partial x} &= \frac{c\pi}{2} \cos(c\pi x) \sin(c\pi t), \\ \frac{\partial U}{\partial t} &= \frac{c\pi}{2} \sin(c\pi x) \cos(c\pi t), \\ \frac{\partial h_s}{\partial x} &= -\frac{c\pi}{2} \sin(c\pi x) \cos(c\pi t).\end{aligned}\tag{B.0.6}$$

From equation (B.0.6) it is easy to verify that equation (B.0.5) does indeed satisfy equation (B.0.2). Now let's use these same analytic solutions within the non-linear SWE, where:

$$\frac{\partial}{\partial t} \begin{pmatrix} h_s \\ U \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} U \\ \frac{U^2}{h} + \frac{1}{2}gh_s^2 \end{pmatrix} = \begin{pmatrix} 0 \\ -gh_B \frac{\partial h_s}{\partial x} \end{pmatrix}.\tag{B.0.7}$$

As previously assumed, we will let $g = h_B = 1$, where, $h = h_s + h_B$, and the PDE system in equation (B.0.7) simplifies to:

$$\frac{\partial}{\partial t} \begin{pmatrix} h_s \\ U \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} U \\ h_s \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} 0 \\ \frac{U^2}{h_s+1} + \frac{1}{2}h_s^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.\tag{B.0.8}$$

From equation (B.0.8) we notice that the only difference from equation (B.0.2) is the addition of the third term on the LHS, which contains the non-linear terms. We already know that the analytic solutions satisfy the first two terms on the LHS of equation (B.0.8); therefore, we will use the method of manufactured solutions to determine what the additional non-linear term is equal to.

Let's begin by taking the partial derivative, such that:

$$\frac{\partial}{\partial x} \left(\frac{U^2}{h_s+1} + \frac{1}{2}h_s^2 \right) = \left[\frac{(h_s+1)2U \frac{\partial U}{\partial x} - U^2 \frac{\partial h_s}{\partial x}}{(h_s+1)^2} + h_s \frac{\partial h_s}{\partial x} \right].\tag{B.0.9}$$

Notice that we already know the analytic solutions for U , h_s , $\frac{\partial h_s}{\partial x}$, and $\frac{\partial U}{\partial x}$. Therefore, at any given point in time, $t \geq 0$, we can compute the respective components of equation (B.0.9)

within the spatial domain of our IBVP using the following equations:

$$\begin{aligned}
(h_s + 1)2U \frac{\partial U}{\partial x} &= \left(\frac{c\pi}{2} \sin(c\pi x) \cos(c\pi x) \sin^2(c\pi t)\right) \left(1 + \frac{1}{2} \cos(c\pi x) \cos(c\pi t)\right), \\
-U^2 \frac{\partial h_s}{\partial x} &= \frac{c\pi}{8} \sin^3(c\pi x) \sin^2(c\pi t) \cos(c\pi t), \\
(h_s + 1)^2 &= \frac{1}{4} \cos^2(c\pi x) \cos^2(c\pi t) + \cos(c\pi x) \cos(c\pi t) + 1, \\
h_s \frac{\partial h_s}{\partial x} &= -\frac{c\pi}{4} \cos(c\pi x) \cos^2(c\pi t) \sin(c\pi x).
\end{aligned} \tag{B.0.10}$$

The manufactured solution for the non-linear terms, $S_{NL}(x, t)$, is therefore defined by the RHS of equation (B.0.9) using the analytic equations in (B.0.10). If we now add S_{NL} to the RHS of the PDE system in equation (B.0.8), then:

$$\frac{\partial}{\partial t} \begin{pmatrix} h_s \\ U \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} U \\ h_s \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} 0 \\ \frac{U^2}{h_s+1} + \frac{1}{2}h_s^2 \end{pmatrix} = \begin{pmatrix} 0 \\ S_{NL}(x, t) \end{pmatrix}, \tag{B.0.11}$$

where the analytic solution, $S_{NL}(x, t)$, in the RHS of (B.0.11) will cancel out the numerical non-linear solution in the third term of the LHS.

Let's now consider the case in which both the gravity, g , and the bathymetry are not equal to one (i.e., $g \neq h_B \neq 1$); however, are still constant. Again, we start by substituting the analytic solutions from equation (B.0.5) into the PDE system in equation (B.0.7). First notice that the only change to equation (B.0.9) is:

$$\frac{\partial}{\partial t} \left(\frac{U^2}{h_s + h_B} + \frac{1}{2}gh_s^2 \right) = \left[\frac{(h_s + h_B)2U \frac{\partial U}{\partial x} - U^2 \frac{\partial h_s}{\partial x}}{(h_s + h_B)^2} + gh_s \frac{\partial h_s}{\partial x} \right], \tag{B.0.12}$$

which modifies the equations in (B.0.10) to:

$$\begin{aligned}
(h_s + h_B)2U \frac{\partial U}{\partial x} &= (2h_B + \cos(c\pi x) \cos(c\pi t)) \left(\frac{c\pi}{4} \sin(c\pi x) \cos(c\pi x) \sin^2(c\pi t)\right), \\
-U^2 \frac{\partial h_s}{\partial x} &= \frac{c\pi}{8} \sin^3(c\pi x) \sin^2(c\pi t) \cos(c\pi t),
\end{aligned} \tag{B.0.13}$$

and

$$\begin{aligned} (h_s + h_B)^2 &= \frac{1}{4} \cos^2(c\pi x) \cos^2(c\pi t) + h_B \cos(c\pi x) \cos(c\pi t) + h_B^2, \\ gh_s \frac{\partial h_s}{\partial x} &= -\frac{gc\pi}{4} \cos(c\pi x) \cos^2(c\pi t) \sin(c\pi x). \end{aligned} \quad (\text{B.0.14})$$

Therefore, the four equations within (B.0.13) and (B.0.14) added together becomes the new analytic non-linear source term, $S_{NL}(x, t)$. Additionally, notice that when $g = h_B = 1$, $\frac{\partial U}{\partial t} = -\frac{\partial h_s}{\partial x}$; however, once we assume that $g \neq h_B \neq 1$, then we must include a linear source term, $S_L(x, t)$, defined by:

$$\begin{aligned} \frac{\partial U}{\partial t} + gh_B \frac{\partial h_s}{\partial x} &= \frac{c\pi}{2} \sin(c\pi x) \cos(c\pi t) - gh_B \frac{c\pi}{2} \sin(c\pi x) \cos(c\pi t), \\ &= \frac{c\pi}{2} \sin(c\pi x) \cos(c\pi t) \left(1 - gh_B\right). \end{aligned} \quad (\text{B.0.15})$$

Notice that when $g = h_B = 1$, the RHS of equation (B.0.15) equals zero (i.e., $S_L(x, t) = 0$). However, when this assumption is removed, we must now include the linear manufactured solution from equation (B.0.14) to the PDE system, such that:

$$\frac{\partial}{\partial t} \begin{pmatrix} h_s \\ U \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} U \\ gh_B h_s \end{pmatrix} + \frac{\partial}{\partial x} \begin{pmatrix} 0 \\ \frac{U^2}{h_s + h_B} + \frac{1}{2} gh_s^2 \end{pmatrix} = \begin{pmatrix} 0 \\ S_L(x, t) + S_{NL}(x, t) \end{pmatrix}, \quad (\text{B.0.16})$$

where the analytic solutions $S_L(x, t)$ and $S_{NL}(x, t)$ in the RHS of (B.0.16) will cancel out both the numerical linear and non-linear solutions in the second and third terms of the LHS, when the equations in (B.0.5) are used as the PDE systems analytic solutions for h_s and U .

APPENDIX C:

Derivation of Linearly Implicit Euler Method

This appendix provides the derivation for the linearly implicit Euler method found in Equation (1.4.8). As previously mentioned, this method is also known as backward Euler, and is written as follows:

$$q_{n+1} = q_n + hf(q_{n+1}), \quad (\text{C.0.1})$$

where the initial value problem being solved is:

$$q'(t) = f(t, q(t)), \quad q(t_0) = q_0, \quad t \geq t_0. \quad (\text{C.0.2})$$

Since the RHS of Equation (C.0.2) cannot be directly solved for using q_{n+1} , we can instead approximate $f(q_{n+1})$ using a first order Taylor series expansion of f . First, we define

$$\Delta q = q_{n+1} - q_n, \quad (\text{C.0.3})$$

and then rewrite Equation (C.0.1) using (C.0.3) as follows:

$$q_{n+1} = q_n + hf(q_{n+1}), \quad (\text{C.0.4a})$$

$$\Delta q + q_n = q_n + hf(\Delta q + q_n), \quad (\text{C.0.4b})$$

$$\Delta q = hf(\Delta q + q_n). \quad (\text{C.0.4c})$$

Now we approximate $f(\Delta q + q_n)$ in Equation (C.0.4c) with its first order Taylor series approximation, such that:

$$f(\Delta q + q_n) \approx f(q_n) + f'(q_n)\Delta q, \quad (\text{C.0.5})$$

so that Δq can be approximated as follows:

$$\Delta q = h(f(q_n) + f'(q_n)\Delta q). \quad (\text{C.0.6})$$

If we now solve Equation (C.0.6) for $f(q_n)$ we find that:

$$hf(q_n) = \Delta q - hf'(q_n)\Delta q, \quad (\text{C.0.7a})$$

$$= (\mathcal{I} - hf'(q_n))\Delta q. \quad (\text{C.0.7b})$$

Notice that we have used the identity matrix, $\mathcal{I}$, in Equation (C.0.7b) since q_n can represent a vector when working with systems of equations, such that f' represents a Jacobian matrix.

Next, we can solve for Δq such that:

$$\Delta q = \left[\mathcal{I} - hf'(q_n) \right]^{-1} (hf(q_n)). \quad (\text{C.0.8})$$

Finally, substituting Δq from Equation (C.0.8) into Equation (C.0.3) and rearranging terms we can write:

$$q_{n+1} = q_n + \Delta q, \quad (\text{C.0.9a})$$

$$= q_n + \left[\mathcal{I} - hf'(q_n) \right]^{-1} (hf(q_n)). \quad (\text{C.0.9b})$$

The final expression for the linearly implicit Euler method in Equation (C.0.9b) can now be used for our multi-rate problem, where if we assume that the RHS of our IVP in Equation (C.0.2) is defined as $f(t, q(t)) = S(t, q) + F(t, q)$, where S and F represent the slow and fast processes, respectively, then:

$$q_{n+1} = q_n + \left[\mathcal{I} - h(S + F)'(q_n) \right]^{-1} \left(hS(q_n) + hF(q_n) \right), \quad (\text{C.0.10})$$

which is the expression for the linearly implicit Euler method we mentioned in Chapter 1.

APPENDIX D:

Numerical Results of Non-Linear SWE

The shallow water equation model results were computed using the Julia computing language. All simulations were run for the following time interval: $0 \leq t \leq 0.4$. The convergence rates, approximate numerical error, and total number of slow and fast RHS evaluations were saved. Additionally, we computed the slow to fast computational ratio using Equation (3.2.3) for each MR method, compared to a SR ERK method from Appendix A. For easy reference, the slow to fast computational ratio, $\frac{S_{comp}}{F_{comp}}$, defined in Chapter 3 is restated:

$$\frac{S_{comp}}{F_{comp}} > \frac{FE_{MR} - \alpha E_{SR}}{E_{SR} - SE_{MR}}, \text{ for } 0 \leq \alpha \leq 1, \quad (\text{D.0.1})$$

This ratio provides a lower bound necessary for our extrapolated MR method to outperform a SR method of equivalent order. The terms above are defined as:

- E_{SR} - is the total number of RHS evaluations for the single-rate method.
- SE_{MR} - is the total number of slow RHS evaluations for the multirate method.
- FE_{MR} - is the total number of fast RHS evaluations for the multirate method.
- S_{comp} - is the computational time to apply the slow RHS operator.
- F_{comp} - is the computational time to apply the fast RHS operator.
- T_{comp} - is the total computational time to apply the RHS operator.

We also make the following assumptions:

- $S_{comp} + \alpha F_{comp} \leq T_{comp}$, for $0 \leq \alpha \leq 1$,
- $SE_{MR} < E_{SR} \leq FE_{MR}$,
- $E_{SR} = SE_{SR} = FE_{SR}$ (The total number of slow and fast RHS evaluations are equal for a single-rate method).

Notice, that T_{comp} represents the total cost of applying the RHS operator, where the cheapest option is to assume the fast computation is embedded inside of the slow computation (i.e., $\alpha = 0$). This case implies that the fast operator is free to compute, and thus $T_{comp} = S_{comp}$. Our results assume the worst case scenario (i.e., $\alpha = 1$), where both the fast and slow RHS operators are independent computations within our code.

Table D.0.1. Extrapolated MIS ERK(1, 1, m , p) Results for $g = 10, h_B = 1$.

Parameters Used: $I = 1, J = 1, g = 10, h_B = 1, c = 2, N = 16, Ne = 25$ with $1 \leq m \leq 6$ and $1 \leq p \leq 5$										
Single-Rate Equivalent using MIS ERK (Outer Order, Inner Order, Sub-Stepping, Extrapolation Order)										
Method	MIS ERK(1,1,1,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.038	N/A	0.038	N/A	0.046	-0.50	0.052	-0.50	0.06	-0.50
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	19590		39180		64736		100212		136488	
Fast Evals	9795		19590		32368		50106		68244	
Total Evals	29385		58770		97104		150318		204732	
Error	10^{-5}		10^{-8}		10^{-12}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,2,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.061	0.33	0.074	0.29	0.09	1.10	0.136	0.94	0.135	3.54
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	12204		20120		33088		38318		60654	
Fast Evals	12204		25150		41360		46529		71682	
Total Evals	24408		45270		74448		84847		132336	
Error	10^{-4}		10^{-9}		10^{-11}		10^{-14}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,3,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.07	0.69	0.112	0.27	0.126	0.92	0.158	1.44	0.194	1.51
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	10636		13296		23632		32984		42218	
Fast Evals	15954		26592		47264		63612		78679	
Total Evals	26590		39888		70896		96596		120897	
Error	10^{-4}		10^{-8}		10^{-11}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,4,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.075	1.04	0.148	0.28	0.151	1.04	0.192	1.43	0.24	1.40
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	9926		10060		19720		27146		34122	
Fast Evals	19852		27665		54230		71743		86856	
Total Evals	29778		37725		73950		98889		120978	
Error	10^{-4}		10^{-8}		10^{-11}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,5,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.078	1.40	0.188	0.26	0.2	0.83	0.226	1.43	0.27	1.54
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	9544		7920		14888		23058		30338	
Fast Evals	23860		27720		52108		77409		97909	
Total Evals	33404		35640		66996		100467		128247	
Error	10^{-4}		10^{-7}		10^{-10}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,6,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.08	1.76	0.222	0.27	0.236	0.82	0.262	1.40	0.302	1.61
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	9306		6708		12624		19894		27126	
Fast Evals	27918		28509		53652		80997		106038	
Total Evals	37224		35217		66276		100891		133164	
Error	10^{-4}		10^{-7}		10^{-10}		10^{-13}		10^{-13}	

Table D.0.2. Extrapolated MIS ERK(1, 2, m , p) Results for $g = 10$, $h_B = 1$.

Parameters Used: $I = 1, J = 2, g = 10, h_B = 1, c = 2, N = 16, N_e = 25$ with $1 \leq m \leq 6$ and $1 \leq p \leq 5$										
Single-Rate Equivalent using MIS ERK (Outer Order, Inner Order, Sub-Stepping, Extrapolation Order)										
Method	MIS ERK(1,2,1,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.036	-19.51	0.044	9.00	0.075	6.23	0.068	-6.15	0.082	-5.20
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	20678		33836		39704		76636		99858	
Fast Evals	31017		67672		79408		147798		186099	
Total Evals	51695		101508		119112		224434		285957	
Error	10^{-5}		10^{-8}		10^{-11}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,2,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.074	2.14	0.076	3.25	0.091	7.26	0.158	4.35	0.146	-0.42
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	10060		19592		32728		32984		56100	
Fast Evals	30180		83266		139094		134292		29300	
Total Evals	40240		102858		171822		167276		85400	
Error	10^{-4}		10^{-9}		10^{-11}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,3,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	2.21	0.112	2.58	0.125	5.28	0.148	8.62	0.186	8.45
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	7920		13296		23824		35210		44044	
Fast Evals	35640		86424		154856		218805		262262	
Total Evals	43560		99720		178680		254015		306306	
Error	10^{-4}		10^{-8}		10^{-11}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,4,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	3.23	0.148	2.35	0.161	4.58	0.183	7.27	0.213	8.39
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	7920		10060		18496		28476		38456	
Fast Evals	47520		88025		161840		237978		307648	
Total Evals	55440		98085		180336		266454		346104	
Error	10^{-4}		10^{-8}		10^{-11}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,5,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	4.25	0.185	2.21	0.198	4.21	0.219	6.61	0.247	7.89
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	7920		8048		15040		23800		33154	
Fast Evals	59400		88528		165440		249900		333047	
Total Evals	67320		96576		180480		273700		366201	
Error	10^{-4}		10^{-8}		10^{-10}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,6,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	5.27	0.222	2.13	0.236	3.98	0.256	6.20	0.283	7.50
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	7920		6708		12624		20356		28952	
Fast Evals	71280		88881		167268		257358		350056	
Total Evals	79200		95589		179892		277714		379008	
Error	10^{-4}		10^{-8}		10^{-9}		10^{-13}		10^{-13}	

Table D.0.3. Extrapolated MIS ERK(1, 3, m , p) Results for $g = 10, h_B = 1$.

Parameters Used: $I = 1, J = 3, g = 10, h_B = 1, c = 2, N = 16, Ne = 25$ with $1 \leq m \leq 6$ and $1 \leq p \leq 5$										
Single-Rate Equivalent using MIS ERK (Outer Order, Inner Order, Sub-Stepping, Extrapolation Order)										
Method	MIS ERK(1,3,1,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,1,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,1,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,1,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,1,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.046	11.38	0.046	17.32	0.058	-69.46	0.068	-14.63	0.08	-5.58
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	16184		32368		51344		76636		139590	
Fast Evals	48552		137564		218212		312018		418770	
Total Evals	64736		169932		269556		388654		558360	
Error	10^{-4}		10^{-8}		10^{-12}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,3,2,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,2,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,2,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,2,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,2,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.091	3.44	0.094	5.10	0.101	12.26	0.117	27.03		
Converge. Rate	1st		2nd		3rd		4th			
Slow Evals	8182		15840		29488		44548			
Fast Evals	49092		138600		258020		372294			
Total Evals	57274		154440		287508		416842			
Error	10^{-4}		10^{-8}		10^{-11}		10^{-13}			
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,3,3,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,3,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,3,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,3,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,3,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	5.27	0.14	4.25	0.147	8.63	0.162	15.07		
Converge. Rate	1st		2nd		3rd		4th			
Slow Evals	7920		10636		20256		32172			
Fast Evals	71280		140927		268392		406746			
Total Evals	79200		151563		288648		438918			
Error	10^{-4}		10^{-8}		10^{-11}		10^{-13}			
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,3,4,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,4,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,4,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,4,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,4,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	7.30	0.185	3.96	0.193	7.54	0.208	12.28		
Converge. Rate	1st		2nd		3rd		4th			
Slow Evals	7920		8048		15432		25060			
Fast Evals	95040		142852		273918		424230			
Total Evals	102960		150900		289350		449290			
Error	10^{-4}		10^{-8}		10^{-10}		10^{-13}			
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,3,5,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,5,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,5,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,5,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,5,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	9.34	0.198	4.67	0.24	6.97	0.254	11.07	0.273	14.01
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	7920		7520		12408		20524		30008	
Fast Evals	118800		167320		276078		435402		608344	
Total Evals	126720		174840		288486		455926		638352	
Error	10^{-4}		10^{-8}		10^{-10}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,3,6,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,6,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,6,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,6,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,3,6,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.094	11.38	0.232	4.64	0.285	6.70	0.3	10.39	0.318	16.69
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	7920		6420		10448		17374		35130	
Fast Evals	142560		171735		279484		443037		632340	
Total Evals	150480		178155		289932		460411		667470	
Error	10^{-4}		10^{-7}		10^{-10}		10^{-13}		10^{-13}	

Table D.0.4. Extrapolated MIS ERK(1, 1, m , p) Results for $g = 10, h_B = 10$.

Parameters Used: $I = 1, J = 1, g = 10, h_B = 10, c = 2, N = 16, Ne = 25$ with $1 \leq m \leq 6$ and $1 \leq p \leq 5$										
Single-Rate Equivalent using MIS ERK (Outer Order, Inner Order, Sub-Stepping, Extrapolation Order)										
Method	MIS ERK(1,1,1,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,1,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0052	N/A	0.0052	N/A	0.0065	-0.50	0.0072	-0.50	0.0084	-0.50
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	143152		286304		458088		723716		974798	
Fast Evals	71576		143152		229044		361858		487399	
Total Evals	214728		429456		687132		1085574		1462197	
Error	10^{-7}		10^{-10}		10^{-14}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,2,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,2,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0074	0.68	0.0104	0.25	0.0126	1.15	0.0194	0.82	0.019	3.34
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	100594		143152		236320		268604		430980	
Fast Evals	100594		178940		295400		326162		509340	
Total Evals	201188		322092		531720		594766		940320	
Error	10^{-5}		10^{-11}		10^{-13}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,3,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,3,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0083	1.18	0.0156	0.25	0.0177	0.94	0.022	1.41	0.027	1.52
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	89686		95436		168224		236852		303270	
Fast Evals	134529		190872		336448		456786		565185	
Total Evals	224215		286308		504672		693638		868455	
Error	10^{-5}		10^{-10}		10^{-12}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,4,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,4,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0089	1.61	0.0209	0.25	0.0227	0.89	0.0268	1.40	0.034	1.35
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	83640		71236		131176		194432		240834	
Fast Evals	167280		195899		360734		513856		613032	
Total Evals	250920		267135		491910		708288		853866	
Error	10^{-5}		10^{-9}		10^{-12}		10^{-13}		10^{-13}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,5,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,5,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0092	2.10	0.0261	0.25	0.028	0.84	0.0317	1.39	0.0377	1.54
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	80912		57044		106344		164388		217206	
Fast Evals	202280		199654		372204		551874		700983	
Total Evals	283192		256698		478548		716262		918189	
Error	10^{-5}		10^{-9}		10^{-12}		10^{-13}		10^{-12}	
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,1,6,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,1,6,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0095	2.52	0.0313	0.25	0.0332	0.82	0.0368	1.36	0.041	1.71
Converge. Rate	1st		2nd		3rd		4th		5th	
Slow Evals	78358		47568		89688		141596		199716	
Fast Evals	235074		202164		381174		576498		780708	
Total Evals	313432		249732		470862		718094		980424	
Error	10^{-5}		10^{-9}		10^{-12}		10^{-13}		10^{-12}	

Table D.0.5. Extrapolated MIS ERK(1, 2, m , p) Results for $g = 10$, $h_B = 10$.

Parameters Used: $I = 1, J = 2, g = 10, h_B = 10, c = 2, N = 16, Ne = 25$ with $1 \leq m \leq 6$ and $1 \leq p \leq 5$										
Single-Rate Equivalent using MIS ERK (Outer Order, Inner Order, Sub-Stepping, Extrapolation Order)										
Method	MIS ERK(1,2,1,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,1,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0052	N/A	0.0064	6.00	0.0107	5.89	0.0097	-6.71		
Converge. Rate	1st		2nd		3rd		4th			
Slow Evals	143152		232624		278280		537194			
Fast Evals	214728		465248		556560		1036017			
Total Evals	357880		697872		834840		1573211			
Error	10^{-7}		10^{-10}		10^{-13}		10^{-13}			
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,2,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,2,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0104	2.00	0.0106	3.11			0.0225	4.05		
Converge. Rate	1st		2nd				4th			
Slow Evals	71576		140452				231588			
Fast Evals	214728		596921				942894			
Total Evals	286304		737373				1174482			
Error	10^{-5}		10^{-10}				10^{-13}			
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,3,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,3,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0111	3.03	0.157	2.47						
Converge. Rate	1st		2nd							
Slow Evals	67062		94828							
Fast Evals	301779		616382							
Total Evals	368841		711210							
Error	10^{-5}		10^{-9}							
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,4,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,4,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0111	4.35	0.0209	2.23						
Converge. Rate	1st		2nd							
Slow Evals	67062		57044							
Fast Evals	402372		627484							
Total Evals	469434		684528							
Error	10^{-5}		10^{-9}							
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,5,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,5,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0111	5.67	0.0261	2.11						
Converge. Rate	1st		2nd							
Slow Evals	67062		57044							
Fast Evals	502965		627484							
Total Evals	570027		684528							
Error	10^{-5}		10^{-9}							
Multirate ERK (Outer Order, Inner Order, Substepping, Extrapolation Order)										
Method	MIS ERK(1,2,6,1)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,2)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,3)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,4)	$\frac{S_{comp}}{F_{comp}}$	MIS ERK(1,2,6,5)	$\frac{S_{comp}}{F_{comp}}$
Max Courant	0.0111	6.99	0.0313	2.04	0.0331	4.02	0.0359	6.07		
Converge. Rate	1st		2nd		3rd		4th			
Slow Evals	67062		47568		89960		145152			
Fast Evals	603558		630276		1191970		1835136			
Total Evals	670620		677844		1281930		1980288			
Error	10^{-5}		10^{-9}		10^{-12}		10^{-13}			

List of References

- [1] Z. Zhang and J.C. Moore. “Chapter 14 - Climate and Earth System Models”. In: *Mathematical and Physical Fundamentals of Climate Change*. Ed. by Zhihua Zhang and John C. Moore. Boston: Elsevier, 2015, pp. 457–472. doi: <https://doi.org/10.1016/B978-0-12-800066-3.00014-0>.
- [2] F.X. Giraldo and T. Warburton. “A high-order triangular discontinuous Galerkin oceanic shallow water model”. In: *IJNMF* 56.7 (2008), pp. 899–925.
- [3] F.X. Giraldo. *An Introduction to Element-Based Galerkin Methods on Tensor-Product Bases*. Vol. 24. Springer, 2020. doi: 10.1007/978-3-030-55069-1.
- [4] F.X. Giraldo and M. Restelli. “High-order semi-implicit time-integration of a triangular discontinuous Galerkin oceanic shallow water model”. English. In: *IJNMF* 63.9 (July 2010), pp. 1077–1102. doi: 10.1002/fld.2118.
- [5] G. Walz. “The History of Extrapolation Methods in Numerical Analysis”. In: *Lehrstuhl für Mathematik IV, Universität Mannheim* (1991).
- [6] L.F. Richardson. “The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in Masonry Dam”. In: *Philosophical Transactions of the Royal Society of London. Series A. Containing Papers of a Mathematical or Physical Character* 210 (1910), pp. 307–357.
- [7] L.F. Richardson. “The Deferred Approach to the Limit”. In: *Philosophical Transactions of the Royal Society of London. Series A. Containing Papers of a Mathematical or Physical Character* 226 (1927), pp. 299–349.
- [8] W. Romberg. “Vereinfachte Numerische Integration”. In: *Norske Vid. Selsk. Forhdl* 28 (1955), pp. 30–36.
- [9] E.M. Constantinescu and A. Sandu. “On extrapolated multirate methods”. In: *Progress in Industrial Mathematics at ECMI 2008*. Ed. by Hans-Georg Bock, Frank Hoog, Avner Friedman, Arvind Gupta, Helmut Neunzert, William R. Puleyblank, Torgeir Rusten, Fadil Santosa, Anna-Karin Tornberg, Vincenzo Capasso, Robert Mattheij, Helmut Neunzert, Otmar Scherzer, Alistair D. Fitt, John Norbury, Hilary Ockendon, and Eddie Wilson. Vol. 15. Mathematics in Industry. 10.1007/978-3-642-12110-4_52. Springer Berlin Heidelberg, 2010, pp. 341–347.
- [10] W.B. Gragg. “On Extrapolation Algorithms for Ordinary Initial Value Problems”. In: *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* 2.3 (1965), pp. 384–403.

- [11] E. Hairer, S.P. Norsett, and G. Wanner. *Solving ordinary differential equations I: nonstiff problems*. Springer, 2008. DOI: 10.1007/978-3-540-78862-1.
- [12] P. Deuffhard. “Order and stepsize control in extrapolation methods”. In: *Numerische Mathematik* 41.3 (1983), pp. 399–422. DOI: 10.1007/BF01418332.
- [13] D.I. Ketcheson and U. bin Waheed. “A Comparison of High-Order Explicit Runge–Kutta, Extrapolation, and Deferred Correction Methods in Serial and Parallel”. In: *Communications in Applied Mathematics and Computational Science* 9.2 (June 2014), pp. 175–200. DOI: 10.2140/camcos.2014.9.175.
- [14] L.F. Shampine. “Stability of the leapfrog/midpoint method”. In: *Applied Mathematics and Computation* 208 (Feb. 2009), pp. 293–298. DOI: 10.1016/j.amc.2008.11.029.
- [15] L.F. Shampine and L.S. Baca. “Smoothing the extrapolated midpoint rule”. In: *Numerische Mathematik* 41 (1983), pp. 165–175.
- [16] P. Deuffhard and F. Bornemann. *Scientific Computing with Ordinary Differential Equations*. New York: Springer, 2002.
- [17] P.D. Williams. “A Proposed Modification to the Robert-Asselin Time Filter”. In: *Monthly Weather Review* 137.8 (1Aug. 2009), pp. 2538–2546. DOI: 10.1175/2009MWR2724.1.
- [18] R. Bulirsch and J. Stoer. *Introduction to Numerical Analysis*. New York: Springer-Verlag, 1991.
- [19] P. Deuffhard. “Recent Progress in Extrapolation Methods for Ordinary Differential Equations”. In: *SIAM Review* 27.4 (1985), pp. 505–535.
- [20] D.I. Ketcheson, L. Lóczi, and M. Parsani. “Internal Error Propagation in Explicit Runge–Kutta Methods”. In: *SIAM Journal on Numerical Analysis* 52.5 (Jan. 2014), pp. 2227–2249. DOI: 10.1137/130936245.
- [21] J.J. Biesiadecki and R.D. Skeel. “Dangers of Multiple Time Step Methods”. In: *Journal of Computational Physics* 109.2 (1993), pp. 318–328. DOI: <https://doi.org/10.1006/jcph.1993.1220>.
- [22] C. Engstler and C. Lubich. “Multirate extrapolation methods for differential equations with different time scales”. English. In: *Computing* 58.2 (1997), pp. 173–185. DOI: 10.1007/BF02684438.
- [23] M. Günther and P. Rentrop. “Multirate ROW methods and latency of electric circuits”. In: *Applied Numerical Mathematics* 13.1 (1993), pp. 83–102. DOI: [https://doi.org/10.1016/0168-9274\(93\)90133-C](https://doi.org/10.1016/0168-9274(93)90133-C).
- [24] J.R. Rice. “Split Runge–Kutta method for simultaneous equations”. In: *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics* (1960), p. 151.

- [25] S. Skelboe and P.U. Anderson. “Stability properties of backward differentiation multirate formulas”. In: *Applied Numerical Mathematics* 5 (Feb. 1989), pp. 151–160. doi: 10.1016/0168-9274(89)90031-7.
- [26] A. Bartel and M. Günther. “A multirate W-method for electrical networks in state–space formulation”. In: *Journal of Computational and Applied Mathematics* 147.2 (2002), pp. 411–425. doi: [https://doi.org/10.1016/S0377-0427\(02\)00476-4](https://doi.org/10.1016/S0377-0427(02)00476-4).
- [27] E.M. Constantinescu and A. Sandu. “Multirate Timestepping Methods for Hyperbolic Conservation Laws”. In: *Journal of Scientific Computing* 33.3 (Dec. 2007), pp. 239–278. doi: 10.1007/s10915-007-9151-y.
- [28] C. Dawson and R. Kirby. “High Resolution Schemes for Conservation Laws with Locally Varying Time Steps”. In: *SIAM Journal on Scientific Computing* 22.6 (2001), pp. 2256–2281. doi: 10.1137/S1064827500367737. eprint: <https://doi.org/10.1137/S1064827500367737>.
- [29] M. Günther, A. Kværnø, and P. Rentrop. “Multirate Partitioned Runge-Kutta Methods”. In: *BIT Numerical Mathematics* 41.3 (2001), pp. 504–514. doi: <https://doi.org/10.1023/A:1021967112503>.
- [30] R. Kirby. “On the Convergence of High Resolution Methods with Multiple Time Scales for Hyperbolic Conservation Laws”. In: *Mathematics of Computation* 72.243 (2003), pp. 1239–1250.
- [31] J.B. Klemp, W.C. Skamarock, and J. Dudhia. “Conservative split-explicit time integration methods for the compressible nonhydrostatic equations”. In: *MWR* 135.8 (2007), pp. 2897–2913.
- [32] A. Kværnø. “Stability of multirate Runge-Kutta schemes”. In: *International Journal of Differential Equations and Applications* 1 (Jan. 2000), pp. 97–105.
- [33] A. Kværnø and P. Rentrop. *Low Order Multirate Runge-Kutta Methods in Electric Circuit Simulation*. 1999.
- [34] A. Sandu and E.M. Constantinescu. “Multirate explicit Adams methods for time integration of conservation laws”. In: *Journal of Scientific Computing* 38.2 (2009), pp. 229–249. doi: 10.1007/s10915-008-9235-3.
- [35] V. Savcenco, W. Hundsdorfer, and J. G. Verwer. “A multirate time stepping strategy for stiff ordinary differential equations”. In: *BIT Numerical Mathematics* 47.1 (2007), pp. 137–155. doi: <https://doi.org/10.1007/s10543-006-0095-7>.
- [36] M. Schlegel, O. Knöth, M. Arnold, and R. Wolke. “Multirate Runge-Kutta schemes for advection equations”. English. In: *Journal of Computational and Applied Mathematics* 226.2 (Apr. 2009), pp. 345–357. doi: 10.1016/j.cam.2008.08.009.

- [37] J. Wensch, O. Knoth, and A. Galant. “Multirate infinitesimal step methods for atmospheric flow simulation”. English. In: *BIT Numerical Mathematics* 49.2 (June 2009), pp. 449–473. doi: 10.1007/s10543-009-0222-3.
- [38] L.D. Carciopolo, L. Bonaventura, A. Scotti, and L. Formaggia. “A conservative implicit multirate method for hyperbolic problems”. English. In: *Computational Geosciences* 23.4 (Aug. 2019), pp. 647–664. doi: 10.1007/s10596-018-9764-2.
- [39] E.M. Constantinescu and A. Sandu. “Extrapolated Implicit-Explicit Time Stepping”. In: *SIAM J. Sci. Comput.* 31.6 (Jan. 2010), pp. 4452–4477. doi: 10.1137/080732833.
- [40] E.M. Constantinescu and A. Sandu. “Extrapolated multirate methods for differential equations with multiple time scales”. English. In: *Journal of Scientific Computing* 56.1 (July 2013), pp. 28–44. doi: 10.1007/s10915-012-9662-z.
- [41] M. Günther and A. Sandu. “Multirate generalized additive Runge Kutta methods”. In: *Numerische Mathematik* 133.3 (July 2016), pp. 497–524. doi: 10.1007/s00211-015-0756-z.
- [42] M. Günther, C. Hachtel, and A. Sandu. “Multirate GARK schemes for multiphysics problems”. English. In: *Scientific Computing in Electrical Engineering (SCEE 2014)*. Ed. by A Bartel, M Clemens, M Gunther, and EJW TerMaten. Vol. 23. Mathematics in Industry. 10th International Conference on Scientific Computing in Electrical Engineering (SCEE), Wuppertal, Germany, JUL 22-25, 2014. Heidelberg Platz 3, D-14197 Berlin, Germany: Springer-Verlag Berlin, 2016, pp. 115–121. doi: 10.1007/978-3-319-30399-4_12.
- [43] M. Schlegel, O. Knoth, M. Arnold, and R. Wolke. “Implementation of multirate time integration methods for air pollution modelling”. English. In: *Geoscientific Model Development* 5.6 (2012), pp. 1395–1405. doi: 10.5194/gmd-5-1395-2012.
- [44] M. Schlegel, O. Knoth, M. Arnold, and R. Wolke. “Multirate Implicit-Explicit Time Integration Schemes in Atmospheric Modelling”. English. In: *Numerical Analysis and Applied Mathematics, Vols I-III*. Ed. by G Psihoyios and C Tsitouras. Vol. 1281. AIP Conference Proceedings. International Conference on Numerical Analysis and Applied Mathematics, Rhodes, Greece, SEP 19-25, 2010. European Soc Comp Methods Sci & Engn. 2 Huntington Quadrangle, Ste 1NO1, Melville, NY 11747-4501 USA: Amer Inst Physics, 2010, pp. 1831+. doi: 10.1063/1.3498252.
- [45] M. Schlegel, O. Knoth, M. Arnold, and R. Wolke. “Numerical solution of multiscale problems in atmospheric modeling”. In: *Applied Numerical Mathematics* 62.10 (2012). Selected Papers from NUMDIFF-12, pp. 1531–1543. doi: <https://doi.org/10.1016/j.apnum.2012.06.023>.

- [46] B. Seny, J. Lambrechts, R. Comblen, V. Legat, and J.-F. Remacle. “Multirate time stepping for accelerating explicit discontinuous Galerkin computations with application to geophysical flows”. English. In: *International Journal for Numerical Methods in Fluids* 71.1 (Jan. 2013), pp. 41–64. doi: 10.1002/fld.3646.
- [47] B. Seny, J. Lambrechts, T. Toulorge, V. Legat, and J.-F. Remacle. “An Efficient Parallel Implementation of Explicit Multirate Runge-Kutta Schemes for Discontinuous Galerkin Computations”. In: *J. Comput. Phys.* 256.C (Jan. 2014), pp. 135–160. doi: 10.1016/j.jcp.2013.07.041.
- [48] O. Knöth and R. Wolke. “Implicit-explicit Runge-Kutta methods for computing atmospheric reactive flows”. In: *Applied Numerical Mathematics* 28.2 (1998), pp. 327–341. doi: [https://doi.org/10.1016/S0168-9274\(98\)00051-8](https://doi.org/10.1016/S0168-9274(98)00051-8).
- [49] J.C. Butcher. *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, 1987.
- [50] J.C. Butcher. “A history of Runge-Kutta methods”. In: *Applied Numerical Mathematics* 20.3 (1996), pp. 247–260. doi: [https://doi.org/10.1016/0168-9274\(95\)00108-5](https://doi.org/10.1016/0168-9274(95)00108-5).
- [51] C. Runge. *Über die numerische Auflösung von Differentialgleichungen*. Vol. 46. 1895.
- [52] K. Heun. *Neue Methoden zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen*. Vol. 45. 1900.
- [53] E.J. Nyström. “Über die numerische Integration von Differentialgleichungen”. In: 50.13 (1925).
- [54] F.X. Giraldo, J.F. Kelly, and E.M. Constantinescu. “Implicit-Explicit Formulations of a Three-Dimensional Nonhydrostatic Unified Model of the Atmosphere (NUMA)”. English. In: *SIAM Journal on Scientific Computing* 35.5 (2013), B1162–B1194. doi: 10.1137/120876034.
- [55] C.A. Kennedy and M.H. Carpenter. “Additive Runge-Kutta schemes for convection-diffusion-reaction equations”. In: *Applied Numerical Mathematics* 44.3 (2003), pp. 139–181.
- [56] Y. Hadjimichael, C.B. Macdonald, D.I. Ketcheson, and J.H. Verner. “Strong Stability Preserving Explicit Runge–Kutta Methods of Maximal Effective Order”. In: *SIAM Journal on Numerical Analysis* 51.4 (Jan. 2013), pp. 2149–2165. doi: 10.1137/120884201.
- [57] E.M. Constantinescu and A. Sandu. “Optimal explicit strong-stability-preserving general linear methods”. In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 3130–3150. doi: 10.1137/090766206.

- [58] C.W. Gear and D.R. Wells. “Multirate linear multistep methods”. English. In: *BIT Numerical Mathematics* 24.4 (1984), pp. 484–502. doi: 10.1007/BF01934907.
- [59] M. Korch, T. Rauber, and C. Scholtes. “Mixed-Parallel Implementations of Extrapolation Methods with Reduced Synchronization Overhead for Large Shared-Memory Computers”. In: *2010 IEEE 16th International Conference on Parallel and Distributed Systems*. 2010, pp. 131–138. doi: 10.1109/ICPADS.2010.12.
- [60] U. Nowak, R. Ehrig, and L. Oeverdieck. “Parallel extrapolation methods and their application in chemical engineering”. In: *High-Performance Computing and Networking*. Ed. by Peter Sloot, Marian Bubak, and Bob Hertzberger. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 419–428.
- [61] M.H. Carpenter and C.A. Kennedy. *Fourth-order 2N-storage Runge-Kutta schemes*. Tech. rep. NASA TM-109112. Langley Research Center, Hampton, VA: National Aeronautics and Space Administration, 1994.
- [62] C.A. Kennedy, M.H. Carpenter, and R.M. Lewis. “Low-storage, Explicit Runge–Kutta Schemes for the Compressible Navier–Stokes Equations”. In: *Applied Numerical Mathematics* 35.3 (2000), pp. 177–219. doi: [https://doi.org/10.1016/S0168-9274\(99\)00141-5](https://doi.org/10.1016/S0168-9274(99)00141-5).
- [63] F.X. Giraldo and M. Restelli. “A study of spectral element and discontinuous Galerkin methods for the Navier–Stokes equations in nonhydrostatic mesoscale atmospheric modeling: Equation sets and test cases”. In: *Journal of Computational Physics* 227.8 (2008), pp. 3849–3877. doi: <https://doi.org/10.1016/j.jcp.2007.12.009>.
- [64] J.H. Williamson. “Low-storage Runge-Kutta schemes”. In: *Journal of Computational Physics* 35.1 (Mar. 1980), pp. 48–56.
- [65] P.J. Van Der Houwen. *Construction of Integration Formulas for Initial Value Problems*. North Holland, Amsterdam, 1977.
- [66] D.I. Ketcheson. “Runge-Kutta Methods with Minimum Storage Implementations”. In: *J. Comput. Phys.* 229.5 (Mar. 2010), pp. 1763–1773. doi: 10.1016/j.jcp.2009.11.006.
- [67] J. Niegemann, R. Diehl, and K. Busch. “Efficient low-storage Runge-Kutta schemes with optimized stability regions”. English. In: *Journal of Computational Physics* 231.2 (Jan. 2012), pp. 364–372. doi: 10.1016/j.jcp.2011.09.003.

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California