MULTI-RATE TIME INTEGRATION ON OVERSET MESHES

BY

CORY MIKIDA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Advisors:

Professor Daniel J. Bodony
Professor Andreas Kloeckner

Abstract

In the computational fluid dynamic simulation of problems with complex geometries or multiscale spatio-temporal features, overset meshes can be effectively used. However, in the case of overset problems in which one or more of the meshes vary significantly in resolution, standard explicit time integrators limit the maximum allowable timestep across the entire simulation domain to that of the finest mesh, per the well-known Courant-Friedrichs-Lewy (CFL) condition. What therefore results is a potentially high amount of computational work that theoretically need not be performed on the grids with coarser resolution and a commensurately larger timestep. With the targeted use of multi-rate time integrators, separate meshes can be marched at independent rates in time to avoid wasteful computation while maintaining accuracy and stability. This work features the application of such integrators (specifically, multi-rate Adams-Bashforth (MRAB) integrators) to the simulation of overset mesh-described problems using a parallel Fortran code. The thesis focuses on the overarching mathematical theory, implementation via code generation, proof of numerical accuracy and stability, and demonstration of serial and parallel performance capabilities. Specifically, the results of this study directly indicate the numerical efficacy of MRAB integrators, outline a number of outstanding code challenges, demonstrate the expected reduction in time enabled by MRAB, and emphasize the need for proper load balancing through spatial decomposition in order for parallel runs to achieve the predicted time-saving benefit.

To my parents

Acknowledgments

I am first and foremost forever grateful for the guidance of my primary advisors on this project, Professors Daniel Bodony and Andreas Kloeckner, of the University of Illinois' Aerospace Engineering and Computer Science Departments, respectively. Without the knowledge you both have imparted to me in countless meetings and reviews of my work, I surely wouldn't be here today.

Additionally, I wouldn't be here without my parents, who doubtlessly taught me the value and importance of hard work, perseverance, and, above all, family. Speaking of which, I also thank the rest of my family: my siblings Eric, Craig, and Kate, my sister-in-law, Ariel, and the rest of my numerous aunts, uncles, cousins, and grandparents, who, for brevity's sake, I'll refrain from naming here.

I thank my friends and groupmates Fabian Dettenrieder, Shakti Saurabh, Mahesh Natarajan, Nek Sharan, Michael Banks, Bryson Sullivan, Mohammad Mehrabadi, Shreyas Bidadi, Wentao Zhang, and Qi Zhang for all of their helpful input and endless wisdom in the ways of fluid mechanics, computer science, and places to eat on Green Street. I also thank my friends here in Illinois and back home in New York - you've been crucial in maintaining my sanity in these past few years.

This material is based in part upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number DE-NA0002374.

Contents

Nomenclature

List of Abbreviations

NNSA	National Nuclear Security Administration
DOE	Department of Energy
XPACC	The Center for Exascale Simulation of Plasma-Coupled Combustion
DNS	Direct Numerical Simulation
CFL	Courant-Friedrichs-Lewy
RK4	fourth-order Runge-Kutta
AB	Adams-Bashforth
SRAB	single-rate Adams-Bashforth
MRAB	multi-rate Adams-Bashforth
SBP	summation-by-parts operator
SAT	simultaneous-approximation-term
RHS	right-hand side
HO-OG	high-order overset grid
MOL	method of lines
ODE	ordinary differential equation
MPRK	multi-rate partitioned Runge-Kutta

List of Symbols

Greek Symbols

ρ	Fluid density
τ_{ij}	Viscous stress tensor
γ	Ratio of specific heats
δ_{ij}	Kronecker delta tensor
μ	First coefficient of Newtonian fluid viscosity
λ	Second coefficient of Newtonian fluid visosity

Roman Symbols

Pr	Prandtl number
Re	Reynolds number
p	Pressure
t	Time
u_i	Velocity in the ith coordinate direction
x_j	jth coordinate direction
q_j	Heat flux in the jth coordinate direction
E	Total energy per unit volume
$\vec{F}$	Flux vector
k	Thermal conductivity
C_p	Specific heat at constant pressure
c	Advection speed
h	Step size
H	Macrostep size

Subscripts, Superscripts and Accents

$(\cdot)_\infty$	Environmental or ambient state

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In the explicit direct numerical simulation (DNS) of computational fluid dynamic problems, the maximum timestep allowable for stable integration of the governing equations is often limited by the well-known Courant-Freidrichs-Lewy condition (CFL). This is given by the simple formula,

$$\mathrm{CFL} = \frac{c\Delta t}{\Delta x}$$

where Δx is the minimum characteristic length of the computational grid, and c is the maximum advection speed of the physical phenomena being simulated. One can quickly conclude that (especially in complex multiscale and multiphysics problems) the timestep taken when integrating over the entire computational domain using a standard explicit single-rate integrator can be limited—sometimes severely—by what occurs on a small portion of the domain, be it due to quickly advecting physical phenomenon or locally-high grid resolution. The result of this limitation is that much wasteful work must be done in large parts of the domain, where the local CFL number is lower, in order to ensure the stable advancement of a particularly fast physical process or fine mesh, giving a suboptimal distribution of work.

With multi-rate integration, solution components differing in timescale can be integrated with independent time steps, allowing computational work to be avoided on the slow components while the fast components remain stable and well-resolved in time. This will improve the performance of the application in serial, and in parallel, provided the proper decomposition of the domain is used;

however, a critical consideration when implementing these integrators is their ability to maintain accuracy and stability of the solution.

In this study, we implement a multi-rate Adams-Bashforth integrator in a massively parallel fluid solver, taking advantage of the solver's overset mesh capabilities to segregate solution components with differing timescales. We do so with particular focus on the resulting improvement in performance, reduction in work through right-hand-side evaluations, and accompanying changes in stability regions and accuracy. What results from this effort is a number of conclusions about the viability and extensibility of these integrators to other problems and applications.

1.2 Historical Perspective

1.2.1 DNS Simulations Using Overset Meshes

The general method of overset meshes (also known as the Chimera method [32]) is an approach that attempts numerical simulation of conservation-law equations by discretizing the domain using multiple, independent overlapping meshes. In this work, the individual meshes are structured, so that the entire domain can be considered locally structured and globally unstructured. Typically, this method is used as an effective way of handling complex geometries or moving-body problems. The earliest known appearance of a composite mesh method was in a numerical application towards the solution of elliptic partial differential equations [39], but similar methods were soon thereafter applied to inviscid transonic flow [18] and the Euler equations [4]. In the years since, high-order overset-grid (HO-OG) approaches have been developed [31] and applied to numerous problems [17], [29], [8].

A further concise summary of recent developments in the usage of overset meshes to solve problems in compressible viscous fluids, along with a discussion of stable and accurate interpolation, is given by [6]. This work also demonstrates a few examples of the overset methods at work, largely on computational aeroacoustic problems, occasionally with moving grids. Additionally, a discussion of various applications of overset can be found in [20].

1.2.2 Multi-rate Time Integration

Some of the earliest work on multi-rate multistep methods, like the ones discussed here, was performed in 1974 by Gear [12]. Ten years later, in 1984, Gear and Wells [13] further pioneered these methods, with a specific focus on their automation. The primary conclusion therein was that the problems of automation were mostly a function of software organization. In 1997, Engstler and Lubich [11] used Richardson extrapolation along with simple Euler schemes to attempt a similarly automated process of multi-rate integration. The resulting method was implemented in a Fortran code and applied to an astrophysics example.

Around the same time as Gear and Wells were pioneering the field of multi-rate, in 1983, Osher and Sanders [21] introduced numerical approximations to conservation laws that changed the global CFL restrictions to local ones. As for the more recent work in this field, in 2001, Dawson and Kirby [10] attempted local time stepping based on the formulations of Osher and Sanders, attaining only first-order accuracy in time in spite of a second-order finite volume approach. Tang and Warnecke [38] expanded on this work in 2006 to produce second-order accuracy via more refined projections of solution increments at each local timestep.

In 2001, Gunther, Kvaerno, and Rentrop [14] introduced multi-rate partitioned Runge-Kutta (MPRK) schemes, starting from a discussion of Rosenbrock-Wanner methods, and based on strategies introduced by Gunther and Rentrop in 1993 [15]. This method focuses on coupling the active and latent solution components primarily via interpolation and extrapolation of state variables, echoing the earlier work presented in [12] and [13].

In 2007, Savcenco et. al [24] developed a self-adjusting timestepping strategy primarily using implicit Rosenbrock methods on stiff ODEs. The schemes developed are based on local temporal error estimation, refining fast-moving solution components with a repeatedly decreasing timestep and using interpolation to obtain values of slower-moving components at the intermediate times requisite to model the coupling between the components. In this study, two timestep estimation strategies are tested: a simple method of repeated bisection and a more involved two-level recursive approach. The methods presented are largely notable for their simplicity, but incur overhead in

their repeated calculation of the solution for the refined components during error estimation.

Also in 2007, Constantinescu and Sandu [9] developed multi-rate timestepping methods for hyperbolic conservation laws based on a method of lines (MOL) approach with partitioned Runge-Kutta schemes. The implemented schemes inherit the strong stability preservation of their single-rate inspiration—that is, the integrators ensure that a certain norm or semi-norm of the solution does not increase— and are second-order accurate in time. The resulting family of schemes also conserve the system invariants, and rigorous proofs of positivity, maximum principle preservation, and total variation boundedness are documented. The separation of solution components is here done on an entirely spatial basis. More recently, in 2009, Sandu and Constantinescu [23] developed explicit multi-rate Adams-Bashforth methods similar to the ones discussed in this thesis, but their application is limited to one-dimensional hyperbolic conservation laws, and their accuracy is limited to second order by the interface region.

Recently, Seny and Lambrechts expanded on the explicit multi-rate Runge-Kutta schemes of Constantinescu and Sandu to discontinuous Galerkin computations for large-scale geophysical flows, introducing the method in 2010 [25] and in the forthcoming years demonstrating its efficacy in both serial and parallel implementations for various problems. Specifically, their latest work in 2014 [26] focuses on efficient parallelization of the method, using a multi-constraint partitioning library to effectively ensure that the same number of cells are active on each processor for a given multi-rate stage, whilst also reducing the number of inter-processor communications and thus minimizing idle time. Unfortunately, the method demonstrated, however, is once again only of second order accuracy.

As for stability, a more rigorous discussion of the stability of a multi-rate method (namely, for numerical integration of a system of first-order ODEs that can be readily separated into subsystems) is given by Andrus in [2]. This study is based on a method that combines a fourth-order Runge-Kutta scheme (for the fast component) with a similar third-order scheme (for the slow component), introduced by the same author in [3].

For further reference on the multi-rate Adams-Bashforth schemes implemented here, thorough analyses (especially empirical analyses of stability and effect of various method design choices) are

given in Kloeckner [16]. The method is also discussed in a particle-in-cell context by Stock [33]. More generally, these methods are multistep methods, and so the work presented in [12], [13], [23], is also most critically relevant and will be referenced as needed in Chapter 2.

1.3 The Structure of the Thesis

In Chapter 2, we will discuss the methods behind our implementation, including (but not limited to) the Navier-Stokes equations on overset meshes, summation-by-parts (SBP) operators, simultaneous-approximation-term (SAT) boundary conditions, interpolation on overset meshes, single-rate and multi-rate Adams-Bashforth time integration, and the software tools used to run the forthcoming simulations. In Chapter 3, we focus on confirming the numerical accuracy and stability of the derived multi-rate integrators using a small-scale test case, and in Chapter 4 we document the current state of performance of the developed integrators on that same case. In Chapter 5, we summarize the obtained results, conclude the study, and discuss future work.

Chapter 2

Overset Meshes and Adams-Bashforth Integration

2.1 The Navier-Stokes Equations on Overset Meshes

Below, we discuss the governing equations to be simulated, along with a number of special characteristics of the solver, operators used, and boundary conditions employed, along with a brief discussion of interpolation on overset meshes.

2.1.1 Governing Equations

For the equations that follow, a non-dimensionalization is employed such that we have

$$t = \frac{t^*_\infty}{L^*/c^*_\infty} \quad x_i = \frac{x^*_i}{L^*} \quad \rho = \frac{\rho^*}{\rho^*_\infty} \qquad\qquad u_i = \frac{u^*_i}{c^*_\infty}$$

$$p = \frac{p^*_\infty}{\rho^*_\infty c^{*2}_\infty} \quad \mu = \frac{\mu^*}{\mu^*_\infty} \quad \lambda = \frac{\lambda^*}{\mu^*_\infty} \quad T = \frac{T^*}{c^{*2}_\infty / C^*_p, \infty} = \frac{T^*}{(\gamma - 1) T^*_\infty}.$$

From these non-dimensional variables, the Reynolds number is defined as

$$\mathrm{Re} = \frac{\rho^*_\infty c^*_\infty L}{\mu^*_\infty}$$

Likewise, the Prandtl number is given by

$$\mathrm{Pr} = \frac{C^*_p \mu^*_\infty}{k^*_\infty}$$

where C^*_p is the specific heat at constant pressure and k^*_∞ is the thermal conductivity.

The conservation equations to be marched in time are given as follows, in summation convention,

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\rho u_j = 0 \\
&\frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij} - \tau_{ij}) = 0 \\
&\frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_j}(\{\rho E + p\}u_j + q_j - u_i\tau_{ij}) = 0.
\end{aligned}$$

where ρ is the mass density, ρu_i is the momentum density, and ρE is the total energy density. These equations can be written in the compact form

$$\frac{\partial Q}{\partial t} + \frac{\partial \vec{F}_j}{\partial x_j} = 0$$

where $Q = [\rho, \rho u, \rho E]^T$ is the vector of conserved variables and $\vec{F} = \vec{F}^I - \vec{F}^V$ is the flux vector.

The Cartesian coordinates $(\vec{x}, t)$ can be mapped to another coordinate system $(\vec{\xi}, \tau)$ via the mappings

$$\vec{x} = X(\vec{\xi}, \tau) \qquad \text{with inverse} \qquad \vec{\xi} = \Xi(\vec{x}, t)$$

where $X^{-1} = \Xi$ and we only consider non-singular mappings such that X^{-1} exists and is well defined. Moreover we assume the time to be invariant, taking $t = \tau$. The Jacobian of the transformation is defined as $J = \det(\partial \Xi_i / \partial x_j)$ and is strictly positive. An application of the chain rule thus allows us to write

$$\frac{\partial}{\partial \tau}\left(\frac{Q}{J}\right) + \frac{\partial \hat{\vec{F}}_i^I}{\partial \xi_i} - \frac{\partial \hat{\vec{F}}_i^V}{\partial \xi_i} = \frac{S}{J}$$

after using the identities

$$\begin{aligned}
\frac{\partial}{\partial \xi_j}\left(\frac{1}{J}\frac{\partial \xi_j}{\partial x_i}\right) &= 0 \qquad \text{for } i = 1, \ldots, N \\
\frac{\partial}{\partial \tau}\left(\frac{1}{J}\right) + \frac{\partial}{\partial \xi_j}\left(\frac{1}{J}\frac{\partial \xi_j}{\partial t}\right) &= 0,
\end{aligned}$$

where N is the number of dimensions. If we define the weighted metric $\hat{\xi}_i = J^{-1}(\partial\xi/\partial x_i)$ and contravariant velocity $\hat{U} = u_j\hat{\xi}_j + \hat{\xi}_t$, with similar expressions for the remaining components, then the inviscid fluxes $\hat{F}_i^I$ are

$$\hat{\vec{F}}_1^I = \begin{bmatrix} \rho\hat{U} \\ \rho u\hat{U} + p\hat{\xi}_x \\ \rho v\hat{U} + p\hat{\xi}_y \\ (\rho E + p)\hat{U} - \hat{\xi}_t p \end{bmatrix} \qquad \text{and} \qquad \hat{\vec{F}}_2^I = \begin{bmatrix} \rho\hat{V} \\ \rho u\hat{V} + p\hat{\eta}_x \\ \rho v\hat{V} + p\hat{\eta}_y \\ (\rho E + p)\hat{V} - \hat{\eta}_t p \end{bmatrix}$$

in two dimensions and

$$\hat{\vec{F}}_1^I = \begin{bmatrix} \rho\hat{U} \\ \rho u\hat{U} + p\hat{\xi}_x \\ \rho v\hat{U} + p\hat{\xi}_y \\ \rho w\hat{U} + p\hat{\xi}_z \\ (\rho E + p)\hat{U} - \hat{\xi}_t p \end{bmatrix}, \qquad \hat{\vec{F}}_2^I = \begin{bmatrix} \rho\hat{V} \\ \rho u\hat{V} + p\hat{\eta}_x \\ \rho v\hat{V} + p\hat{\eta}_y \\ \rho w\hat{V} + p\hat{\eta}_z \\ (\rho E + p)\hat{V} - \hat{\eta}_t p \end{bmatrix}, \qquad \text{and} \quad \hat{\vec{F}}_3^I = \begin{bmatrix} \rho\hat{W} \\ \rho u\hat{W} + p\hat{\zeta}_x \\ \rho w\hat{V} + p\hat{\zeta}_y \\ \rho w\hat{W} + p\hat{\zeta}_z \\ (\rho E + p)\hat{W} - \hat{\zeta}_t p \end{bmatrix}$$

in three dimensions.

The viscous stress constitutive relation is given by

$$\tau_{ij} = \frac{\mu}{\mathrm{Re}}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \frac{\lambda}{\mathrm{Re}}\frac{\partial u_k}{\partial x_k}\delta_{ij}$$

and the transport coefficients μ, λ and k are modeled by the power law, with n defined for air,

$$\frac{\mu}{\mu_\infty} = \frac{k}{k_\infty} = \frac{\lambda}{\lambda_\infty} = \left(\frac{T}{T_\infty}\right)^n, \quad n = 0.666$$

For the viscous terms, we use the nonorthogonal strong form given by Tannehill, Anderson, and Pletcher [1]:

$$\frac{\partial}{\partial t}\left(\frac{\rho u_1}{J}\right) = \cdots \frac{\partial}{\partial \xi}\left(\hat{\xi}_i \tau_{i1}\right) + \frac{\partial}{\partial \eta}\left(\hat{\eta}_i \tau_{i1}\right) + \frac{\partial}{\partial \zeta}\left(\hat{\zeta}_i \tau_{i1}\right)$$
$$\frac{\partial}{\partial t}\left(\frac{\rho u_2}{J}\right) = \cdots \frac{\partial}{\partial \xi}\left(\hat{\xi}_i \tau_{i2}\right) + \frac{\partial}{\partial \eta}\left(\hat{\eta}_i \tau_{i2}\right) + \frac{\partial}{\partial \zeta}\left(\hat{\zeta}_i \tau_{i2}\right)$$
$$\frac{\partial}{\partial t}\left(\frac{\rho u_3}{J}\right) = \cdots \frac{\partial}{\partial \xi}\left(\hat{\xi}_i \tau_{i3}\right) + \frac{\partial}{\partial \eta}\left(\hat{\eta}_i \tau_{i3}\right) + \frac{\partial}{\partial \zeta}\left(\hat{\zeta}_i \tau_{i3}\right)$$
$$\frac{\partial}{\partial t}\left(\frac{\rho E}{J}\right) = \cdots \frac{\partial}{\partial \xi}\left(\hat{\xi}_i [u_j \tau_{ij} - q_i]\right) + \frac{\partial}{\partial \eta}\left(\hat{\eta}_i [u_j \tau_{ij} - q_i]\right) + \frac{\partial}{\partial \zeta}\left(\hat{\zeta}_i [u_j \tau_{ij} - q_i]\right)$$

2.1.2 Summation-by-Parts (SBP) Operators

In order to discretize the spatial derivatives present in the equations above, we make use of several finite difference operators that possess the summation-by-parts (SBP) property. Taking two matrices P, Q, we here state that these two matrices are SBP matrices of order p provided

- $P^{-1}Q\mathbf{v}$ is an order h^p approximation to $\partial/\partial x$, where h is the spatial step size in one dimension.
- P is a symmetric positive-definite matrix.
- $Q + Q^T = \text{diag}(-1, 0, 0, ..., 0, 1)$.

These conditions together ensure that the discrete version of the integration by parts property holds; that is,

$$\langle P^{-1}Q\mathbf{x}, \mathbf{y}\rangle_P = \mathbf{x}_N \mathbf{y}_N - \mathbf{x}_1 \mathbf{y}_1 - \langle \mathbf{x}, P^{-1}Q\mathbf{y}\rangle_P.$$

The resulting operators can be either explicit (in this case, P is purely diagonal) or implicit. This theory, originally presented in [34], can also be extended to higher dimensions using Kronecker products. For example, in two dimensions, the matrices $H^{-1}G_x$ and $H^{-1}G_y$ define the x- and y-derivatives on a two-dimensional grid:

$$H = P_x \otimes P_y$$
$$G_x = Q_x \otimes P_y$$
$$G_y = P_x \otimes Q_y$$

Note that this formulation assumes that we have P_x, Q_x, a pair of $n_x \times n_x$ SBP matrices of approximation order p, and P_y, Q_y, a pair of $n_y \times n_y$ SBP matrices of approximation order q.

Finally, we also note that these SBP operators do not guarantee strict stability for an initial boundary value problem. We must also apply the boundary conditions using a formulation that permits an energy estimate. More information on the boundary conditions is given in the next section.

More information on SBP operators of various order, including the coefficients themselves, can be found in [34], [7], [19]. For the results provided in Chapter 3, we use a third-order SBP operator.

2.1.3 Simultaneous-Approximation-Term (SAT) Boundary Conditions

As mentioned above, in order to facilitate an energy estimate (and thus prove energy stability), we must combine the SBP operators with a specific weak boundary treatment. In order to characterize this treatment, we examine a simple case: the continuous one-dimensional advection equation. Per reference [7], we take

$$\frac{\partial u}{\partial t} = \lambda \frac{\partial u}{\partial x}, \quad 0 \geq x \geq 1,$$

In order for this to be well-posed for $\lambda > 0$, we require the boundary condition $u(1,t) = g(t)$ at $x = 1$. We can then write the discrete system, using SBP operators P and Q, as

$$\frac{d}{dt}u = \lambda P^{-1} Q u - \tau \lambda q_{N,N} P^{-1} E_1 (u_N - g(t))$$

where $q_{N,N}$ is the bottom-right element of Q, $E_1 = (0, 0, ...0, 1)^T$, and where τ is a parameter set by the user. It can be shown via a brief analysis that $\tau \geq 1/2$ gives energy stability for this case. The semi-discrete problem has an energy estimate in the P-norm given by

$$\frac{d||\mathbf{u}||_P^2}{dt} = \left(\mathbf{u}, \frac{d\mathbf{u}}{dt}\right)_P + \left(\mathbf{u}, \frac{d\mathbf{u}}{dt}\right)_P = -\mathbf{u}^T(Q + Q^T)\mathbf{u} - 2\tau u_0^2 + 2\tau g(t) u_0 = (1 - 2\tau)u_0^2 - u_N^2 + 2\tau g(t) u_0$$

where $\mathbf{u}(t) = [u_0(t), ..., u_N(t)]^T$ on the discrete domain $x_j = nh$, $h = 1/N$, $n = 0, 1, ..., N$. We can

see that if we set $g(t) = 0$ and $\tau \geq 1/2$,

$$\frac{d||\mathbf{u}||_P^2}{dt} \leq 0 \Rightarrow ||\mathbf{u}||_P \leq K||\mathbf{f}||_P$$

where K is constant—that is, the approximation is Lax stable. Generally speaking, it is straightforward to observe that while increasing the penalty parameter τ increases the accuracy of the overall solution, it also increases the numerical stiffness of the problem.

A similar treatment can be used for the compressible Navier-Stokes equations as given here, and is discussed more thoroughly in [36], [37]. For now, we simply give a few of the critical details. Following the notation of Svärd and Nordstrom, we can give the penalized equation as

$$\frac{\partial \mathbf{q}}{\partial t} = R(q) + \sigma^{I1} P^{-1} E_1 A^+ (\mathbf{q} - \mathbf{g}^{I1}) + \frac{\sigma^{I2}}{\mathrm{Re}} P^{-1} E_1 I (\mathbf{q} - \mathbf{g}^{I2})$$

where σ^{I1} and σ^{I2} are the penalty parameters for the inviscid and viscous boundary conditions, respectively, Re is the Reynolds number, I is the identity matrix, and once again, $E_1 = (0, 0, ...0, 1)^T$. $R(q)$ represents the divergence of the fluxes in the governing equations, and $A^+ = T\Lambda^+ T^{-1}$ selects only the incoming characteristic variables $\mathbf{R} = T\mathbf{q}$ (where T is given by Pulliam and Chaussee [22], and transforms the conserved variables to characteristic variables). $\Lambda^+ = \Lambda - |\Lambda|$ is a diagonal matrix such that $\Lambda = \mathrm{diag}\{\hat{U}, \hat{U}, \hat{U} + c, \hat{U} - c\}|\nabla_x \xi|$ where $\hat{U}$ is the wall-normal component of the velocity. The target vectors for the inviscid and viscous penalty terms, respectively, are given for a no-slip, isothermal condition for a non-moving wall as

$$\mathbf{g}^{I1} = \begin{bmatrix} \rho \\ \rho(\mathbf{u} - (\mathbf{u} \cdot \mathbf{n})\mathbf{n}) \\ \frac{p}{\gamma - 1} + \frac{1}{2}\rho|\mathbf{u} - (\mathbf{u} \cdot \mathbf{n})\mathbf{n}|^2 \end{bmatrix}$$

$$\mathbf{g}^{I2} = [\rho, \mathbf{0}, \rho T_w/\gamma]^T$$

where T_w is the wall temperature. As for the penalty parameters, it is known that in order to attain numerical stability, we must have $\sigma^{I1} \leq -2$ and

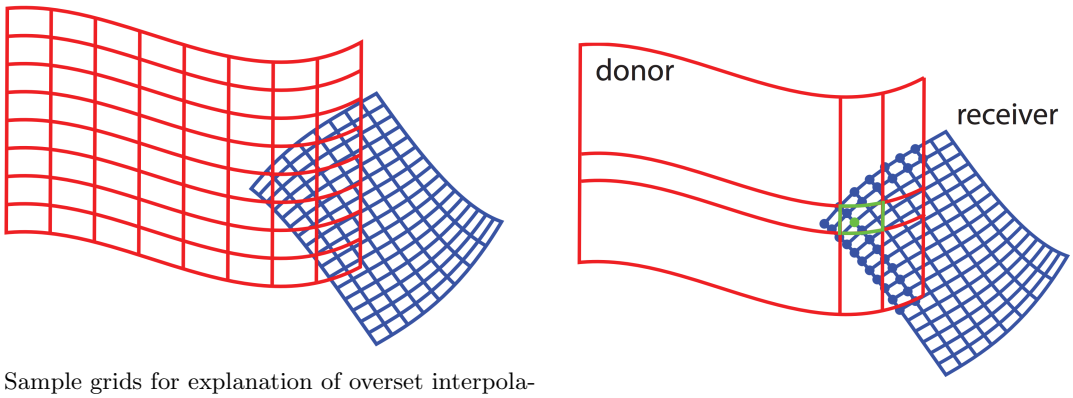
$$\sigma^{I2} \leq -\frac{1}{4p_0}\max\left(\frac{\gamma\mu}{\mathrm{Pr}\rho}, \frac{5\mu}{3\rho}\right)$$

where μ is the first viscosity coefficient, Pr is the Prandtl number, and γ is the ratio of specific heats. For more on the demonstrated accuracy of these boundary conditions as applied to aeroacoustic problems, see [5].

2.1.4 Overset Interpolation

Our interpolation between overset meshes relies on a Chimera framework that makes use of PEGASUS [35] and BELLERO [30] formats and tools. The process of communication between grids will be described in minimal detail here, but for more information the reader should refer to reference [6]. In general, the process can be broken down into a number of phases:

- Establish communication between grids. Each process computes the bounding box of each grid that it "owns," and, via collective communication, determines any collisions it may have with other grids on other processors.
- Hole cutting/fringe determination. After first classifying grids as either background grids or feature grids, we can use an integer-valued array to identify points as "fringe points" which will donate and receive data from other grids, and also to identify points on the background grid which are well within the boundaries of a feature grid, and can thus be deemed inactive.
- Donor-receiver pair search — see Figure 2.1. Fringe points on the receiver grid are paired with donor cells on the donor grid.
- Interpolation. State data from the points in the donor cell is transferred to the receiver point via Lagrangian interpolation, with corresponding weights determined as a function of Lagrange shape functions.



(a) Sample grids for explanation of overset interpolation.

(b) The receiver points and donor cell.

Figure 2.1: Outlining the overset procedure.

Further discussion of this implementation, including algorithms for hole-cutting and the donor-receiver pair search, is omitted here, but given in [6]. The majority of this process, including the collision determination, hole cutting, fringe identification, donor-receiver pair search, and calculation of the interpolation weights, is performed as a preprocessing step. The interpolation itself is performed at each Runge-Kutta substep in the case of an RK4-driven run, and is considered part of a given grid's right-hand side evaluation.

Once the donor data is sent (interpolated) to the receiver grids, it must be applied to the receiver state. There are two such methods of application that will be briefly discussed here - it is important to note that only the latter method (the SAT-based approach) is used in generating the results we will present later, and in fact is a critical step in attaining proper numerical convergence of the integrators.

2.1.4.1 Injection

In the compressible Navier-Stokes solver we will be using for the forthcoming simulations, the standard method of interpolation between overset meshes uses a simple injection procedure to apply the result of overset interpolation, in which the state values on the receiver grid are directly overwritten by the interpolated values from the donor grid.

One of the complications associated with this method of interpolation is that it operates directly

on the state being integrated (in our case, the vector $Q = [\rho, \rho\vec{u}, \rho E]^T$ for a given receiver point) rather than the right-hand side of our governing equations. Strictly speaking, ordinary differential equations and their discretizations depend strongly upon the idea that state values are only altered by explicit applications of a right-hand side, as opposed to outright replacements - the numerical issues introduced by injection-based interpolation when using MRAB integrators is readily seen in practice, and is a direct result of the use of right-hand side history values as a critical piece (as we will see in Section 2.2) of the time-marching scheme. In order for the scheme to maintain a history of right-hand side values that accurately models the evolution of the state, the result of interpolation must show up as a right-hand side term. When state is modified via other means, such as the injection we are discussing here, the right-hand sides being extrapolated through to step forward in time can no longer be considered a sound and complete approximation of the actual temporal behavior of the state.

2.1.4.2 SAT-Based Penalty Interpolation

An alternative interface treatment follows a methodology similar to the weak boundary treatment described in Section 2.1.3, and applies the interpolated values as a penalization term in the right-hand side via a target vector. As done in [27], we consider a single grid point on an overlapping interface - using the same notation as [27], we describe this interface as a $\kappa^{\pm}$ boundary where $\kappa = \xi$, η, or ζ. κ is the normal direction to the face the grid point lies on, and the $\pm$ superscript indicates inflow (+) or outflow (-). If we denote the solution at this grid point as $\mathbf{q}_{ijk}$, with the interpolated value from the donor grid given as $\hat{\mathbf{q}}_{ijk}$, we can express the discretized equation at this point as

$$\frac{d\mathbf{q}_{ijk}}{dt} = -(D_{\xi m}\mathbf{F}_m)_{ijk} - p_0^{-1}(\sigma^I K_\kappa^{\pm} + \sigma_1^V I_5)(\mathbf{q}_{ijk} - \hat{\mathbf{q}}_{ijk}) + \sigma_2^V \left((F_\kappa^V)_{ijk} - (\hat{F}_\kappa^V)_{ijk}\right)$$

where $(D_{\xi_m}, \mathbf{F}_m)_{ijk}$ denotes the derivatives of the fluxes, $\mathbf{F}_m = \mathbf{F}_m^I - \mathbf{F}_m^V$, p_0 is the (1,1) element of the P matrix, I_5 is an identity matrix of size 5×5, and $K_\kappa^{\pm} = T_\kappa \left(\frac{|\Lambda_\kappa| \pm \Lambda_\kappa}{2}\right) T_\kappa^{-1}$, where T and Λ are the same matrices mentioned in Section 2.1.3, given by Pulliam and Chaussee [22]. $(F_\kappa^V)_{ijk}$ denotes the viscous flux at the interface point, and all hatted terms indicate interpolated values.

Note also that if the grid point lies on an edge or a corner in 3 dimensions, the interface terms for each normal direction must be added. The penalty parameters σ are given by

$$\sigma^I \geq \frac{1}{2}, \quad \sigma_1^V = \frac{1}{2\mathrm{Re}}(\kappa_x^2 + \kappa_y^2 + \kappa_z^2), \quad \sigma_2^V = \pm\frac{1}{2}$$

for an inflow (+) or outflow (-) interface point. As opposed to the injection-based interpolation method described above, this scheme is dissipative and provably stable. This method of interpolation is more thoroughly outlined in [27] and [28].

The most critical of benefits associated with using this interpolation scheme in the multi-rate context is that it allows for application of the result of interpolation as a component of the right-hand side applied to the state—that is, interpolation contributes to a right-hand side term rather than a simple state replacement. Furthermore, it also more readily allows us to apply this "result" of interpolation selectively (i.e. to one grid at a time) depending on which right-hand side we are currently evaluating (fast or slow). These reasons allow interpolation to be incorporated into the Adams-Bashforth framework laid out in the forthcoming section as a component of the right-hand side histories, and, as a result, this method of interpolation is critical to attaining the numerical accuracy and stability we will show later.

2.2 Adams-Bashforth Integration

In this section, we describe our implementation of Adams-Bashforth methods of various order; both single-rate and multi-rate forms are discussed.

2.2.1 Single-rate Adams-Bashforth (SRAB) Integration

Here we give a brief derivation of a standard Adams-Bashforth integrator. We will step through the derivation of a third-order AB integrator in detail, while also giving the final results without derivation for fourth and fifth order AB integrators as well, which we will use in upcoming sections.

We start with a simple ODE given by

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y).$$

We can thus find the solution at the next timestep by taking

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s, y(s))\mathrm{d}s.$$

The operating principle of Adams-Bashforth methods and other similar linear multistep methods involves approximating the right-hand side function by a polynomial using past values of $f(t, y)$, extrapolating through that approximation, and integrating.

For the time being, we assume that $t_{n+1} - t_n = t_n - t_{n-1} = t_{n-1} - t_{n-2} = h$. In practice, and for the cases we will later discuss, this is not the case. The computational procedure for eliminating this restriction on our derived coefficients is discussed in Section 2.4.1. Assuming a constant value of h, the third-order extrapolation of $y(t_n)$ to time t_{n+1} is given as

$$y(t_{n+1}) = y(t_n) + \frac{23}{12}hf(t_n, y_n) - \frac{4}{3}hf(t_{n-1}, y_{n-1}) + \frac{5}{12}hf(t_{n-2}, y_{n-2}).$$

By similar procedure, the fourth and fifth-order Adams-Bashforth integrators are given as

$$\begin{aligned} y(t_{n+1}) = y(t_n) &+ \frac{55}{24}hf(t_n, y_n) - \frac{59}{24}hf(t_{n-1}, y_{n-1}) \\ &+ \frac{37}{24}hf(t_{n-2}, y_{n-2}) - \frac{3}{8}hf(t_{n-3}, y_{n-3}) \end{aligned}$$

$$\begin{aligned} y(t_{n+1}) = y(t_n) &+ \frac{1901}{720}hf(t_n, y_n) - \frac{1387}{360}hf(t_{n-1}, y_{n-1}) \\ &+ \frac{109}{30}hf(t_{n-2}, y_{n-2}) - \frac{637}{360}hf(t_{n-3}, y_{n-3}) \\ &+ \frac{251}{720}hf(t_{n-4}, y_{n-4}) \end{aligned}$$

It is clear from our derivation of the method that the length of the past history needed to

calculate a step (and, thus, the memory required) is directly dependent on the order of accuracy desired. A derivation of a first-order Adams-Bashforth integrator simply results in Euler's method.

We note that given that Adams-Bashforth integrators require past history values of the right-hand side to march forward in time, an alternative method is required for the first few time steps (the exact number of which is dependent on order) in order to establish history and "bootstrap" the method. In all cases discussed in upcoming sections, a standard 4th-order Runge-Kutta integrator is used to bootstrap the AB methods.

2.2.2 Multi-rate Adams-Bashforth Integration

With the fundamentals of Adams-Bashforth integration firmly established, we can now attempt a multi-rate implementation of the scheme. We replace our initial problem given in Section 2.3.1 with a two-component system comprised of fast and slow portions of the solution (henceforth using the notation of Kloeckner [16], where f indicates the "fast" variable and s the "slow"):

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} f(t) \\ s(t) \end{pmatrix} = \begin{pmatrix} a_{ff}(f,s) + a_{fs}(f,s) \\ a_{sf}(f,s) + a_{ss}(f,s) \end{pmatrix}$$

Assuming smoothly varying right-hand side terms $a_{ff}, a_{ss}, a_{sf}, a_{fs}$, we note that the linearity of the integral required for explicit Adams-Bashforth integration allows us to maintain separate histories (with independent time intervals) for each right-hand side term. With this in mind, we can set a slow (larger) time step H such that we maintain stability in the integration of the slow component, subsequently setting a fast time step h such that h is an integer multiple of H and, as a result we can define the ratio between the two, $k = H/h$, as the step ratio of the MRAB scheme. We note that the choice of rate for the two diagonal (self-influencing) right-hand side terms, a_{ff} and a_{ss}, is quite clear, whereas the rates of the coupling terms a_{fs} and a_{sf} is less so. As for the integration of each right-hand side component such that it influences the state, the procedure and coefficients (dependent on order) outlined in the section above are unchanged.

In the overset formulation with which we are concerned, we define the fast and slow components

of our Navier-Stokes solution as the conserved variables on each grid, that is (using a two-grid case as an example): $f = Q_1 = [\rho, \rho\vec{u}, \rho E]^T$, $s = Q_2 = [\rho, \rho\vec{u}, \rho E]^T$, where the subscripts of the vectors Q indicate global grid number, and in this instance we assume Grid 1 to be the grid with the fast-moving component of the solution, be it due to physical behavior or finer mesh quality (per the aforementioned CFL condition). In this case, the coupling terms a_{fs} and a_{sf} in the above formulation are embodied by the SAT-based interpolation scheme described in Section 2.2.4.

Within this simple two-component scheme, a number of design choices are available to the user, including but not limited to:

- The order in which we evaluate and advance the solution components. Should we advance the fast-evolving solution component through all of its substeps and wait to perform the single macro-step required for the slow component until the end (a "fastest-first" scheme, per the nomenclature of [16]), or should we pursue an algorithm in which the slow component is instead advanced first?

- If we have explicit coupling terms a_{fs} and a_{sf}, at what rates should they advance? Should they use the micro-timestep h or the macro-timestep H?

- For slowest-first evaluation schemes, should we re-extrapolate the slow state after additional state and right-hand side information is gathered at the substeps?

An in-depth discussion of the numerical effects of these choices in the context of the overset formulation described is deferred to Chapter 3. Additionally, empirical observations on the effects of these choices, among others, are made by Kloeckner [16]. For the results shown in Chapters 3 and 4 (with the exception of Section 3.2.4), we use a simple fastest-first scheme with no re-extrapolation.

2.3 Software Tools and Issues

Below, we give brief synopses of the software tools used in the simulations that follow.

2.3.1 PlasComCM

PlasComCM is a Fortran 90 code written to solve the compressible Navier-Stokes equations (see Section 2.1.1) for various orders on overset meshes. The code has been used to solve problems involving compressible turbulence, fluid-structure interaction, and sound generation and propagation, and is capable of simulating moving boundaries. Additionally, PlasComCM is currently being used in the University of Illinois' NNSA and DOE-funded PSAAPII center, the Center for Exascale Simulation of Plasma-Coupled Combustion (XPACC), and is thus the subject of various computer science-oriented optimizations and novel numerical and scientific simulation techniques. For more on PlasComCM and XPACC, see `https://bitbucket.org/xpacc-dev/plascomcm` and `https://xpacc.illinois.edu`, respectively.

2.3.2 Leap

Leap is a Python package used to describe integration methods (including multi-rate integrators) with flexible algorithms via a virtual machine, and is capable of describing both implicit and explicit time steppers in the form of instructions that can then be passed to Dagrt (see below) to generate Fortran or Python code. Leap has been primarily developed by Prof. Andreas Kloeckner and student Matt Wala at the University of Illinois at Urbana-Champaign. For more on Leap's functionality, dependencies, and capabilities, see `https://documen.tician.de/leap/`.

2.3.3 Dagrt

Dagrt, a second Python package, is a DAG-based runtime system which can generate Fortran or Python code implementing the integrators described by Leap for a given right-hand-side. In using this tool, and Leap, with a host application, the user needs to describe the data types to be operated on, along with the right-hand-sides that the host application uses, in a short Python driver. As with Leap, Dagrt has been developed by Prof. Andreas Kloeckner and student Matt Wala at the University of Illinois at Urbana-Champaign in the Computer Science department. More information can be found at `https://documen.tician.de/dagrt/`.

2.3.4 Data Ownership

A pervasive issue in implementing Leap-generated integrators within PlasComCM's existing codebase has been data ownership; that is, the host application (PlasComCM) "owns" the state and right-hand side data in its existing structures, but Leap-generated integrators also want to "own" this same state and right-hand side data in its own data structures whilst operating upon it and storing/accessing history required to march in time. At this time, there is not yet a policy for who is responsible for allocating and freeing the buffers that contain the results of state operations.

Thus far, the issues associated with data ownership have been circumvented using Fortran pointers - that is, before state-dependent right-hand side evaluations in Leap code are performed, we must first save PlasComCM's states in separate pointers, then point PlasComCM's data structures to Leap's state. Then the right-hand side evaluation occurs as normal, using PlasComCM structures and thus causing no PlasComCM-side issues. Finally, once the given right-hand side evaluation is complete, the PlasComCM structures are pointed back to the initially saved PlasComCM states.

A similar procedure must be carried out with the right-hand side: upon initialization of a given right-hand side evaluation, PlasComCM's right-hand side structures are pointed to Leap's result structures (fast or slow, dependent on the given right-hand side instance). However, since PlasComCM does not care explicitly about right-hand side histories, we need not save right-hand side data in separate pointers before evaluating a right-hand side, and similarly, we need not reset PlasComCM's right-hand side structures once the right-hand side evaluation is complete.

2.3.5 Timestep Adaptivity

While the packages Leap and Dagrt have been developed largely independently of PlasComCM and its various applications, we here give a brief discussion of one of the major capabilities added to Leap-generated integrators to better facilitate its interoperation with PlasComCM: timestep adaptivity. In the use of PlasComCM for its various fluid dynamic applications, a constant CFL time-marching mode is often used to evolve the solution in time, resulting in a constantly changing timestep. This is problematic for Leap-generated Adams-Bashforth integrators - as we have seen

above, the coefficients multiplying the right-hand side history values depend directly on timestep h, so calculating these coefficients once on initialization (as Leap had previously done) is no longer sufficient. Leap-generated AB integrators calculate the coefficients used in the integration using a Vandermonde system with monomial basis:

$$V^T \cdot \alpha = \int_0^{\Delta t} x^i dx$$

where α is the vector of coefficients to be solved for, and V^T is the transpose of the Vandermonde matrix with monomial basis, given by

$$V = \begin{bmatrix} 1 & t_1 & \dots & t_1^{n-1} \\ 1 & t_2 & \dots & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \dots & t_n^{n-1} \end{bmatrix}$$

Here, n is equal to the order of the integrator, and t_i are the time history values. These coefficients, when obtained, are used to extrapolate to the next state via

$$y(t_{i+1}) = y(t_i) + \alpha_1 f(t_{i-n}, y_{i-n}) + \alpha_1 f(t_{i-n-1}, y_{i-n-1}) + \ldots + \alpha_n f(t_i, y_i)$$

In order to handle a constantly changing timestep, these operations must now be performed once every macro-timestep. As a result, the Leap-generated Fortran code now makes use of LAPACK routines for the necessary linear algebra solves (not needed with a constant timestep), and so proper linking with the appropriate libraries is required to use the generated code. Furthermore, time history information for each solution component must now be stored and tracked by Leap integrators, whereas before this was unnecessary.

Chapter 3

Stability and Convergence of Multi-rate Integrators

3.1 Test Case

In this section, we outline the MRAB test case that will be used to produce the results discussed in this chapter, as well as in Chapter 4.

3.1.1 Physical Problem

The two-dimensional problem modeled by our test case is viscous flow over a cylinder with diameter $D = 0.6$ in the non-dimensional spatial domain $x/D = [-4, 4]$, $y/D = [-4, 4]$. The cylinder center is located at $x/D = -1.2$, $y/D = 0$. The initial condition used in this case models uniform subsonic flow in the positive x-direction, with a Mach number of 0.2. The Reynolds number is 200 and the Prandtl number is 0.72.

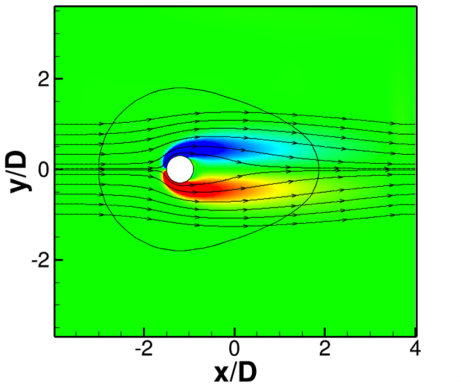


Figure 3.1: Physical case: flow past a cylinder.

3.1.2 Computational Model

The test case models the physical problem discussed above on two overset meshes: a coarser, base Cartesian grid, and a finer teardrop-shaped curvilinear grid surrounding the IBLANK region representing the stationary cylinder.

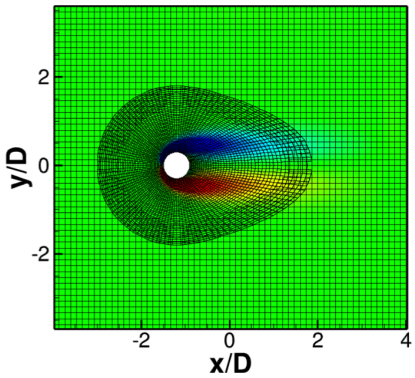


Figure 3.2: Overset grid configuration: flow past a cylinder.

Grid	N_x	N_y	N_{points}
1	61	61	3721
2	121	41	4961

Table 3.1: Grid sizes for test case.

For the purposes of multi-rate, it is important to characterize the disparity in timescales between the two grids, such that we can estimate the maximum allowable substep ratio we are able to attain. In order to gain a sense of this parameter, we query the minimum initial timesteps on each grid for a fixed CFL, and find the ratio between them to be about 12 (see Table 3.2 below).

Grid	Δt_{min}
1	0.261994
2	0.021430

Table 3.2: Measured timesteps for test problem shown in Figure 3.1.

3.1.3 Boundary Conditions

The boundary conditions employed in our simulation are given in Table 3.3 below:

Grid	Boundary Condition	Location	Direction
1	SAT Far-Field	Left ($x = -3$)	$+x$
1	Sponge	Left ($x = -3$)	$+x$
1	SAT Far-Field	Right ($x = 3$)	$-x$
1	Sponge	Right ($x = 3$)	$-x$
1	SAT Far-Field	Bottom ($y = -3$)	$+y$
1	Sponge	Bottom ($y = -3$)	$+y$
1	SAT Far-Field	Top ($y = 3$)	$-y$
1	Sponge	Top ($y = 3$)	$-y$
2	SAT Isothermal Wall	Cylinder Surface	Normal to Surface

Table 3.3: Description of boundary conditions for test case.

Note that the inner fine grid (Grid 2) is periodic, and therefore no boundary conditions are needed in the x-direction. As for the cylinder surface itself, we model it as an SAT isothermal wall. On Grid 1 (Cartesian base grid), all boundaries are modeled as SAT far-field (see Chapter 2). In addition, sponge boundaries with a cell depth of 6 are used.

3.2 Results

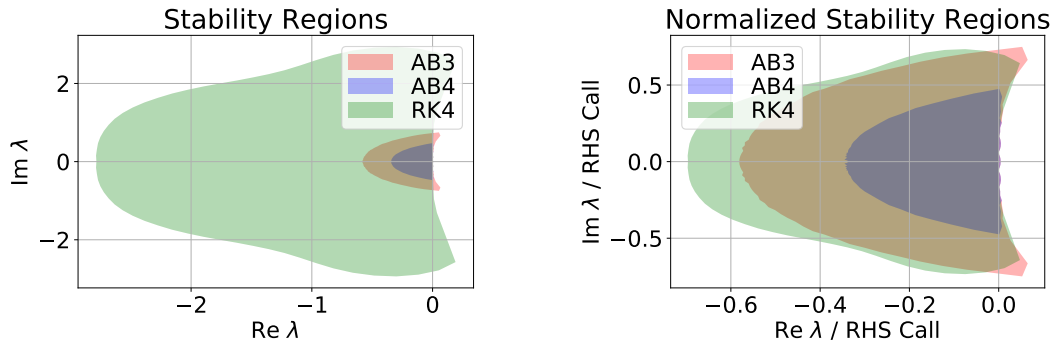
3.2.1 Stability

3.2.1.1 Theory

Before discussing the results of applying MRAB integrators to the case we have outlined, it will serve us well to first briefly discuss the stability regions of third and fourth-order SRAB integrators in comparison to PlasComCM's standard RK4 integration. To do so, we will define a simple differential equation

$$\frac{dy}{dt} = \lambda y$$

and plot in complex λ-space the regions within which each integrator remains stable:



(a) Stability plots for each integrator.

(b) Stability plots normalized by RHS calls.

Figure 3.3: Stability regions for AB integrators compared to RK4.

We see that in general, Adams-Bashforth integrators have a far more restrictive stability region than an explicit fourth-order Runge-Kutta integrator—however, it is important to also note that RK4 requires four right-hand side evaluations per timestep, whereas an Adams-Bashforth integrator only requires one. Figure 3.3b normalizes the stability regions of each integrator based on the number of right-hand side evaluations required per timestep, and as a result we see regions that are far more commensurate in size. This tells us that while SRAB integrators have far lower maximum stable timesteps (roughly one fifth that of RK4 for a third-order SRAB integrator, and about one ninth for fourth-order), they also require less computation per step.

3.2.1.2 Procedure

In characterizing the stability limits of our integrators with various step ratios, we follow a simple procedure: we increase the CFL value used by PlasComCM to set the macro-timestep until numerical instability is observed in the solution. The stability results that we present below for our third and fourth order integrators are given in terms of this metric, to the nearest 0.01. The CFL value reported is that for the slow, coarser grid. The step ratio is a user-selected independent parameter.

The CFL ratio we will report in our upcoming results is a measure of the efficiency of a given MRAB integrator, and is defined as $r = \mathrm{CFL}_{MRAB}/\mathrm{CFL}_{SRAB}$. Therefore, a MRAB integrator

with a step ratio of 2 should attain a CFL ratio of 2, a MRAB integrator with a step ratio of 3 should attain a CFL ratio of 3, and so on and so forth, until the speed ratio between the grids is reached—in our case, we have already determined this ratio to be about 12.

It is important to note that the CFL ratio does not measure effectiveness of a given MRAB integrator compared to RK4 in terms of the maximum stable timestep that the solver can take—for this comparison, the CFL limits themselves are given. Based on the theory outlined in Section 3.2.1.1, we expect the maximum stable CFL for the SRAB integrator of third order to be about one fifth that of the standard RK4 integrator, while the SRAB integrator of fourth order should have a maximum stable CFL about one ninth that of RK4.

3.2.1.3 Third Order

We present below a table comparing the stable CFL condition for our simulation using a fourth-order Runge-Kutta simulation to those of MRAB-driven simulations at various step ratios.

Integrator	CFL Limit	CFL Ratio
RK4 (PlasComCM)	4.17	5.02
Single-Rate Adams-Bashforth	0.83	1.00
MRAB (Step Ratio = 2)	1.66	2.00
MRAB (Step Ratio = 3)	2.49	3.00
MRAB (Step Ratio = 4)	3.33	4.01
MRAB (Step Ratio = 5)	4.16	5.01
MRAB (Step Ratio = 6)	5.00	6.02
MRAB (Step Ratio = 7)	5.81	7.00
MRAB (Step Ratio = 8)	6.30	7.59
MRAB (Step Ratio = 9)	6.30	7.59
MRAB (Step Ratio = 10)	6.30	7.59
MRAB (Step Ratio = 20)	6.30	7.59

Table 3.4: Stability results for third order Adams-Bashforth integrators.

The primary result of note here is that past a certain step ratio, we are unable to further increase the maximum stable CFL, and in fact, past a step ratio that is slightly more than half of the observed speed ratio for our case of interest (about 12), we see no real benefit to the increase, since the maximum stable macro-timestep remains unchanged. Above this step ratio, we expect

our performance to degrade, given that we are simply performing extra right-hand side evaluations (more substeps on the fast grid) with no commensurate macro-timestep gain.

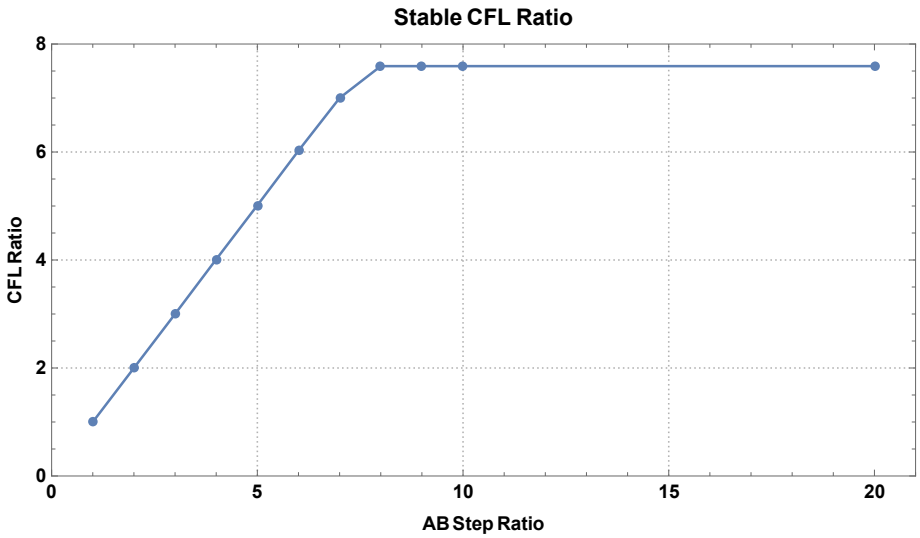


Figure 3.4: Stable CFL ratio as a function of step ratio - third order.

We also note that for MRAB step ratios greater than 5, we are able to drive simulations at higher CFL numbers than RK4 achieved without introducing instability. Given the reduced number of right-hand side evaluations required on the coarse grid, we expect performance benefit for these step ratios especially, given that therefore fewer iterations are required to stably reach the same end time. For now, a more thorough discussion of performance expectations and benefits is deferred to Chapter 4.

3.2.1.4 Fourth Order

Now we can perform the same tests on our fourth-order MRAB integrators, noting that the increase in order shrinks our stability region considerably, necessitating the use of lower CFL numbers for all step ratios.

As we would expect, we see the same trend for fourth order as we did for third order - past a step ratio of 8, we can continue to increase the step ratio, and the resulting integrators will drive our

Integrator	CFL Limit	CFL Ratio
RK4 (PlasComCM)	4.16	9.24
Single-Rate Adams-Bashforth	0.45	1
MRAB (Step Ratio = 2)	0.91	2.02
MRAB (Step Ratio = 3)	1.36	3.02
MRAB (Step Ratio = 4)	1.82	4.04
MRAB (Step Ratio = 5)	2.27	5.04
MRAB (Step Ratio = 6)	2.73	6.07
MRAB (Step Ratio = 7)	3.19	7.09
MRAB (Step Ratio = 8)	3.32	7.38
MRAB (Step Ratio = 9)	3.32	7.38
MRAB (Step Ratio = 10)	3.32	7.38
MRAB (Step Ratio = 20)	3.32	7.38

Table 3.5: Stability results for fourth-order Adams-Bashforth integrators.

simulations just fine, but we are no longer able to increase the maximum allowable CFL. This means that as we increase the step ratio, we are simply doing more work while using the same macro-timestep as with lower step ratios, thus making these higher step ratio integrators comparatively inefficient.

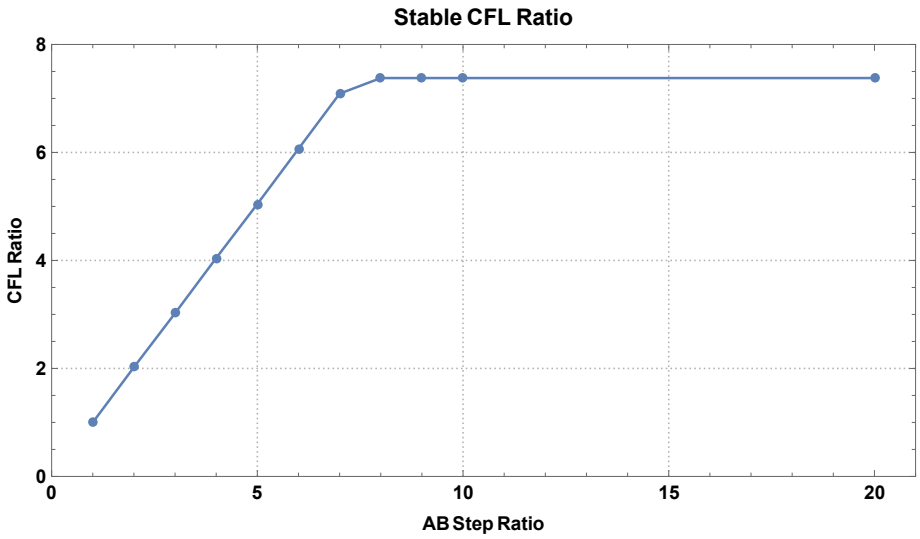


Figure 3.5: Stable CFL ratio as a function of step ratio - fourth order.

3.2.2 Validation With Small-Scale Results

In an attempt to validate the stability results we have above, we will now aim to replicate the trend we have observed using a small-scale one-dimensional advection case. The case models the advection of a Gaussian bump across an interface at which the SAT exchange discussed in Section 2.3.2 occurs. The code used to model this example was written by Nek Sharan to demonstrate the efficacy of the interpolation method discussed in Section 2.3.2, and in reference [27].

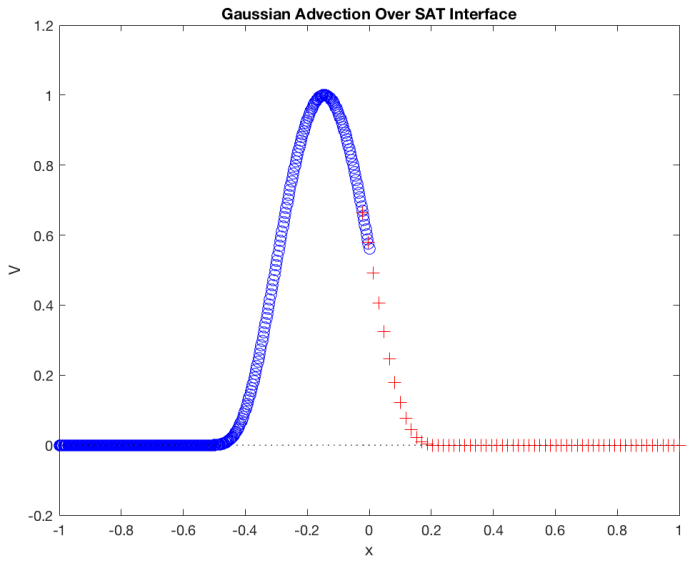


Figure 3.6: The small-scale case models one-dimensional advection of a Gaussian from a fine grid to a relatively coarse one.

While only in one dimension, SBP operators of order 3 as discussed in Section 2.3 are still used for the spatial discretization. The case is periodic in x, and the computational domain is $x \in [-1, 1]$, with the grid interface occurring at $x = 0$. For the specific case we will present results for, Grid 1 (the left grid) contains 54 equally spaced points, while Grid 2 (the right grid) contains 6 equally spaced points. In this case, the CFL condition tells us (assuming a constant advection speed) that the speed ratio between the grids should be exactly equal to 9.

By symbolically determining the resulting step matrix using MAXIMA (see `http://maxima.sourceforge.net/`) for each case as a function of the timestep and performing a search to determine the maximum

timestep below which the eigenvalues of the matrix are negative, we can roughly characterize the stability bounds for each method. Due to prohibitive costs of the step matrix formation (see Figure 3.7), we perform the stability search only for third order MRAB integrators for step ratios of 1 through 7. We give the results of this process here.

MRAB Step Ratio	Maximum Stable Real Timestep	Maximum Stable Imaginary Timestep
1	0.009	0.009
2	0.019	0.019
3	0.029	0.029
4	0.039	0.039
5	0.049	0.049
6	0.059	0.059
7	0.069	0.069

Table 3.6: Maximum stable timesteps for one-dimensional advection case.

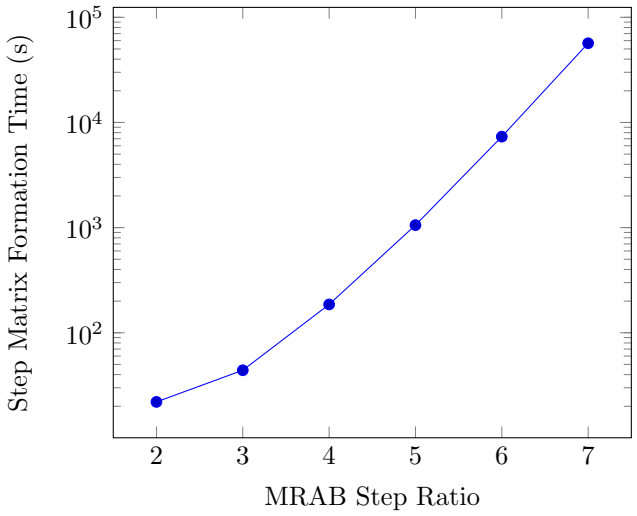


Figure 3.7: Total time spent composing the step matrices for a range of MRAB step ratios.

We see here that our MRAB integrators remain stable up to higher step ratios here, with the maximum stable real and imaginary timesteps linearly increasing through all step ratios tested. We expect that the failure to realize the stability plateau behavior manifested in the results given in Section 3.2.2 and Section 3.2.3 is due to lack of boundary conditions (as previously mentioned, the

one-dimensional case we test here is periodic). Furthermore, this case also does not include any diffusion, which is clearly present in the governing equations we are solving in the PlasComCM case tested.

3.2.3 Accuracy and Convergence

Below we examine the accuracy and convergence of the developed integrators at various step ratios, attempting to draw conclusions for various orders.

3.2.3.1 Procedure

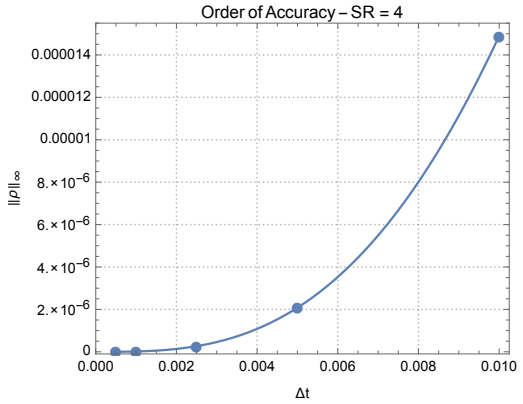
In presenting some accuracy results for our schemes, we will generate integrators for step ratios ranging from 1 (single-rate) to 8. As we have seen in the previous section, above this step ratio, the maximum stable CFL remains unchanged for our speed ratio of about 12. For each integrator, we attempt to calculate an order of accuracy using 4 - 5 data points consisting of the macro-timestep used and the maximum error obtained in the density contour after a solution time of 2.5 seconds.

3.2.3.2 Third Order

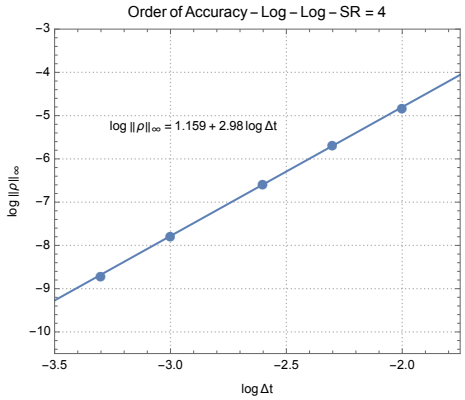
To show the execution of our procedure, we present the application and results of our convergence study for a third-order multi-rate integrator with a step ratio of 4. These same results for additional step ratios are given in the appendix.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.01	1.49E-05	-2.000	-4.827
0.005	2.08E-06	-2.301	-5.682
0.0025	2.67E-07	-2.602	-6.573
0.001	1.66E-08	-3.000	-7.780
0.0005	2.02E-09	-3.301	-8.694

Table 3.7: Convergence data for third-order MRAB, SR=4.



(a) Plot of density error curve for SR=4.



(b) Log-log plot for SR=4.

Figure 3.8: Plotted third-order convergence data for SR=4.

Below, we attempt to concisely present our results for order of accuracy obtained by all step ratios studied.

MRAB Step Ratio	Observed Order of Accuracy
1	2.969
2	3.238
3	2.988
4	2.980
5	2.983
6	2.967
7	2.973
8	2.974

Table 3.8: Convergence results for third-order Adams-Bashforth integrators.

Simply put, we see that an empirical order of accuracy of about 3 is obtained in all cases — this is consistent with our theoretical analysis presented in Chapter 2, and confirms that the implementation is sound.

3.2.3.3 Fourth Order

For testing of our fourth order integrators, nothing about our procedure changes, but we do note that the range of timesteps explored does become smaller across all step ratios - this is primarily

due to two factors:

- As discussed in the Section 3.2.1.1, the stability region for the Adams-Bashforth formulation shrinks as the order increases, and thus the stability regions for the fourth order integrators tested here are smaller than those of the third order integrators already tested, and furthermore, the stability region is significantly smaller than that of the standard fourth-order explicit Runge-Kutta formulation customarily used by PlasComCM.
- At smaller time steps like the lower limits of those examined in the case of the third-order integrators, the error in the density contour eventually becomes too low to be accurately measured by our analysis tools.

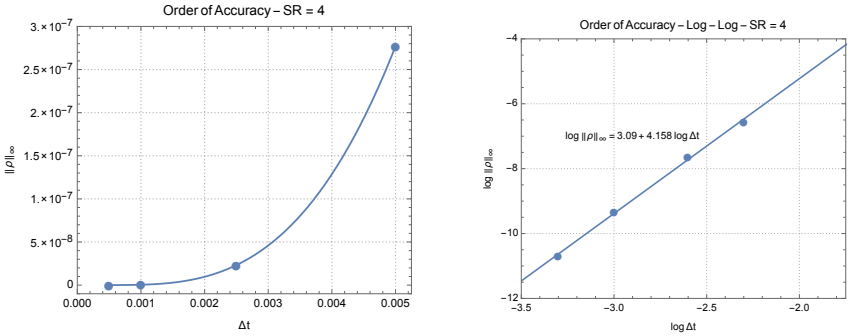
As in the previous subsection, we present a sample application of our convergence study to a fourth-order integrator with a step ratio of 4, followed by our results for all step ratios studied for fourth-order integrators here.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.005	2.77E-07	-2.301	-6.558
0.0025	2.32E-08	-2.602	-7.635
0.001	4.62E-10	-3.000	-9.335
0.0005	2.01E-11	-3.301	-10.697

Table 3.9: Convergence data for fourth-order MRAB, SR=4.

MRAB Step Ratio	Observed Order of Accuracy
1	3.872
2	4.169
3	4.126
4	4.158
5	3.879
6	4.040
7	4.164
8	4.225

Table 3.10: Convergence results for fourth-order Adams-Bashforth integrators.



(a) Plot of density error curve for SR=4. (b) Log-log plot for SR=4.

Figure 3.9: Plotted fourth-order convergence data for SR=4.

Once again, the order of accuracy is observed to be sufficiently close to 4 in all cases, suggesting that the scheme developed is sound.

3.2.3.4 Comparison With Runge-Kutta

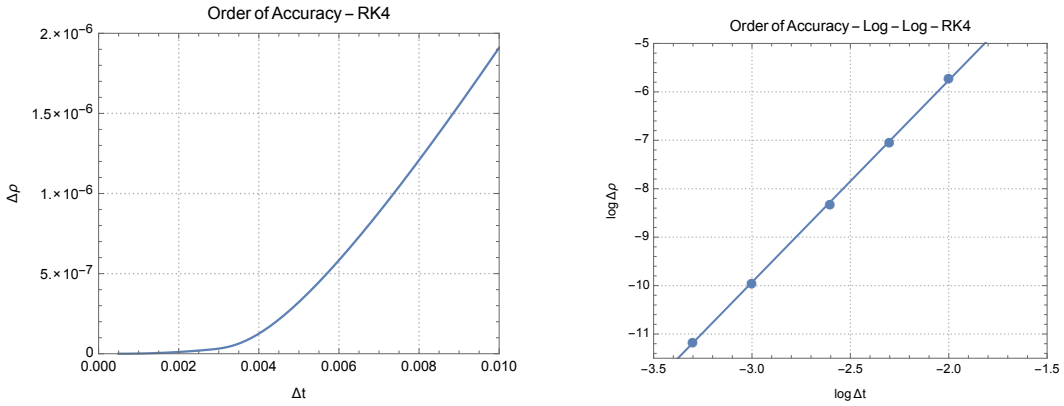
For comparison, we will examine the numerical performance of these Adams-Bashforth integrators, established above, in comparison with a standard explicit fourth-order Runge-Kutta scheme, commonly used for time marching by PlasComCM and given by the following set of equations:

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\
k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5),
\end{aligned} \tag{3.1}
$$

With the stability of RK4 for our simulation already well-established in the above sections (maximum stable CFL of 4.17), we can run a simple convergence test (as performed for MRAB) of this integrator, presenting those results below.

Δt	$\|\rho\|_\infty$	log(Δt)	$\log(\|\rho\|_\infty)$
0.01	1.91E-06	-2.000	-5.719
0.005	9.06E-08	-2.301	-7.043
0.0025	4.91E-09	-2.602	-8.309
0.001	1.15E-10	-3.000	-9.939
0.0005	6.99E-12	-3.301	-11.16

Table 3.11: Convergence data for RK4.



(a) Plot of density error curve for PlasComCM RK4.

(b) Log-log plot for RK4.

Figure 3.10: Plotted convergence data for RK4.

We see here that PlasComCM's standard RK4 integrator indeed achieves fourth-order accuracy.

3.2.4 Influence of MRAB Policy Decisions

We will now demonstrate the effect of a few policy decisions mentioned in Section 2.2.2 on the stability of MRAB integrators at various step ratios and orders, using the same case used to obtain the stability and convergence results given in the previous sections.

The lack of coupling right-hand-side terms allows us to limit our scope of study to the following two policy decisions:

- Term evaluation order: in the case of two rates, do we use a fastest-first or slowest-first scheme? In theory, a scheme in which the fast component is evaluated first at the microsteps

should be numerically superior, given that the evaluation of the slow that occurs at the end can then make use of more recently computed fast state.

- Should we re-extrapolate the slow state as more information becomes available? Note that this option is only available when using a slowest-first scheme, in which the overall step begins with an extrapolation of the slow state—in the case of a fastest-first scheme, our extrapolation for the slow state occurs at the end.

We will use an MRAB integrator with a fastest-first evaluation order as a baseline against which other candidates are measured - this is the integrator for which the above convergence and stability results have been obtained. For the purposes of this study, we will limit the scope of this investigation to the effect of these parameters on stability, and we will perform stability tests in the same manner described in Section 3.2.1. Note that we here restrict ourselves to studying step ratios ranging from 2 to 8 (where we see the maximum stable CFL for MRAB reach a plateau for both orders). We adopt a nomenclature similar to [16] here, such that the first capitalized letter indicates which right-hand side is evaluated first, and a lowercase "r" indicates re-extrapolation of the slow state.

The maximum stable CFL numbers attainable for a number of configurations based on changing these policy decisions are given in Table 3.11 (third order) and Table 3.12 (fourth order) below.

Policy Combo	SR=2	SR=3	SR=4	SR=5	SR=6	SR=7	SR=8
F	1.66	2.49	3.33	4.16	5.00	5.81	6.30
S	1.66	2.49	3.33	4.16	5.00	5.81	6.30
Sr	1.66	2.49	3.33	4.16	5.00	5.81	6.31

Table 3.12: Maximum stable CFL numbers for various policy combos - third order.

Policy Combo	SR=2	SR=3	SR=4	SR=5	SR=6	SR=7	SR=8
F	0.91	1.36	1.82	2.27	2.73	3.19	3.37
S	0.91	1.36	1.82	2.27	2.73	3.19	3.37
Sr	0.91	1.36	1.82	2.27	2.73	3.19	3.37

Table 3.13: Maximum stable CFL numbers for various policy combos - fourth order.

In short, we see that both the order of evaluation and re-extrapolation appear to have no effect on the stability as seen in the maximum stable CFLs the MRAB integrators attain—for both third and fourth order integrators we see the exact behavior seen in Section 3.3, with a plateau being reached at a step ratio of 8. The identical stability nature of the simple "F" and "S" schemes mirrors the result given in [16] for a simple 2×2 model system, and demonstrates our implementation to be insensitive to choices regarding order of evaluation. Similarly, the lack of change in the maximum stable CFL with the introduction of re-extrapolation of the slow state suggests that a single early extrapolation of the slow state is sufficient.

Chapter 4

Performance of Multi-Rate Integrators

Having in the prior chapter established the numerical accuracy and stability of the resulting multi-rate integrators, we now examine their performance, with specific focus on end-to-end simulation wallclock time and the reduction in the required right-hand-side evaluations.

4.1 Performance Model

We develop a rough performance model for how we would expect a multi-rate Adams-Bashforth integrator of a certain step ratio to perform in comparison to PlasComCM's standard Runge-Kutta integrator. In doing so, we make a number of assumptions:

- We assume that right-hand side evaluations make up the bulk of the cost of running a simulation.
- We assume that all comparison runs to validate this model will be performed at or near the maximum stable timestep of a given integrator.

In all performance modeling and testing, we run each integrator to the same end time — that is, we must scale the number of iterations each integrator is run to ensure that each reaches the same point in time. As an example: if the RK4 integrator can stably run at $\Delta t = 0.1$, but an SRAB integrator can only stably run at $\Delta t = 0.05$, we must run the SRAB integrator for 20 iterations in order to accurately compare to an RK4 run over 10 iterations.

As an example of this performance model in execution, we can compose a model for the case discussed in Chapter 3 modeling the cylinder in crossflow, for a number of step ratios, and for both orders:

Integrator	Total RHS Evaluations	% Reduction from RK4	Est. Speedup
RK4 (PlasComCM)	34244	0.00	1.00x
Single-Rate Adams-Bashforth	42805	-25.00	0.80x
MRAB (Step Ratio = 2)	33503	2.16	1.02x
MRAB (Step Ratio = 3)	30402	11.22	1.13x
MRAB (Step Ratio = 4)	28851	15.75	1.19x
MRAB (Step Ratio = 5)	27921	18.46	1.23x
MRAB (Step Ratio = 6)	27301	20.28	1.25x
MRAB (Step Ratio = 7)	26858	21.57	1.28x

Table 4.1: Evaluating RHS costs for MRAB integrators compared to RK4 - third order.

Integrator	Total RHS Evaluations	% Reduction from RK4	Est. Speedup
RK4 (PlasComCM)	34244	0.00	1.00x
Single-Rate Adams-Bashforth	77049	-125.00	0.44x
MRAB (Step Ratio = 2)	60305	-76.10	0.57x
MRAB (Step Ratio = 3)	54723	-59.80	0.63x
MRAB (Step Ratio = 4)	51932	-51.65	0.66x
MRAB (Step Ratio = 5)	50258	-46.76	0.68x
MRAB (Step Ratio = 6)	49142	-43.50	0.70x
MRAB (Step Ratio = 7)	48344	-41.18	0.71x

Table 4.2: Evaluating RHS costs for MRAB integrators compared to RK4 - fourth order.

Note that we need separate models for each order of accuracy because of the change in stability region between orders. Table 4.1 shows that based on right-hand side evaluations required to reach the same end time, we can expect speedup for all MRAB integrators for this specific case — however, based on Table 4.2, we speculate that fourth-order accurate MRAB integrators will fail to be profitable for this specific case.

More generally, we can do more to theorize the location of a performance-critical step ratio for a given implementation of multi-rate on this overset case — in other words, we can calculate the minimum step ratio above which the benefits of multi-rate Adams-Bashforth (reduction in

RHS evaluations per macro-timestep) should outweigh its drawbacks (reduction in maximum stable macro-timestep when compared to fourth-order Runge-Kutta). For a two-rate case, the resulting expression comparing the required work of the two integrators is as follows, noting from the stability results of Chapter 3 that a single-rate AB integrator needs to take a timestep roughly 5 times as small as that of RK4 in order to remain stable:

$$4(n_{points,1} + n_{points,2}) = 5(n_{points,1} + \frac{1}{\text{SR}}(n_{points,2}))$$

We can therefore solve for the minimum beneficial step ratio for third order for this specific case, given the grid sizes given in Chapter 3, expecting it to be :

$$\text{SR}_{crit} = \frac{5n_{points,2}}{4(n_{points,1} + n_{points,2}) - 5n_{points,1}}$$

$$\text{SR}_{crit} = 1.55$$

Noting that we only test integer step ratios here for convenience, what this practically tells us for this case is that while a single-rate Adams-Bashforth integrator will not provide performance benefit here, a multi-rate Adams-Bashforth integrator with a step ratio of 2 should. More generally speaking, this conclusion is essentially a mathematical statement of the fact that the benefit of multi-rate in the context of overset is highly dependent on both the speed ratio of the grids and the relative sizes of the grids in question.

4.2 Serial Performance

We can run a number of performance tests to the same end time with both PlasComCM's standard Runge-Kutta integrator and a number of MRAB integrators with varying step ratios, tracking in these runs a number of timings:

- End-to-end wallclock time of the full application
- Inclusive time spent in right-hand side calculation subroutines (that is, the total time spent

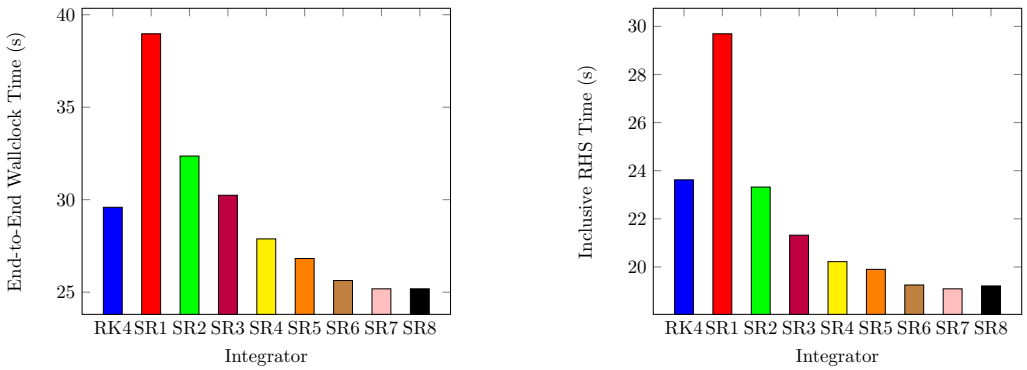
in right-hand side subroutines and all the subroutines that those subroutines call)

- Inclusive time spent in interpolation between grids
- Inclusive time spent in operator-related subroutines

We limit our performance testing here to third-order MRAB integrators only, noting that the small stability region of the fourth-order MRAB integrators clearly precludes us from attaining performance benefit using them for this specific case.

Integrator	RHS (s)	Operator (s)	Interp (s)	Total End-to-end (s)
RK4 (PlasComCM)	23.62	13.70	0.546	29.59
Single-Rate Adams-Bashforth	29.69	17.43	0.287	38.97
MRAB (Step Ratio = 2)	23.32	13.53	0.426	32.36
MRAB (Step Ratio = 3)	21.32	12.42	0.383	30.24
MRAB (Step Ratio = 4)	20.22	11.80	0.362	27.88
MRAB (Step Ratio = 5)	19.90	11.57	0.350	26.82
MRAB (Step Ratio = 6)	19.25	11.21	0.340	25.63
MRAB (Step Ratio = 7)	19.09	11.15	0.335	25.18
MRAB (Step Ratio = 8)	19.21	11.19	0.335	25.18

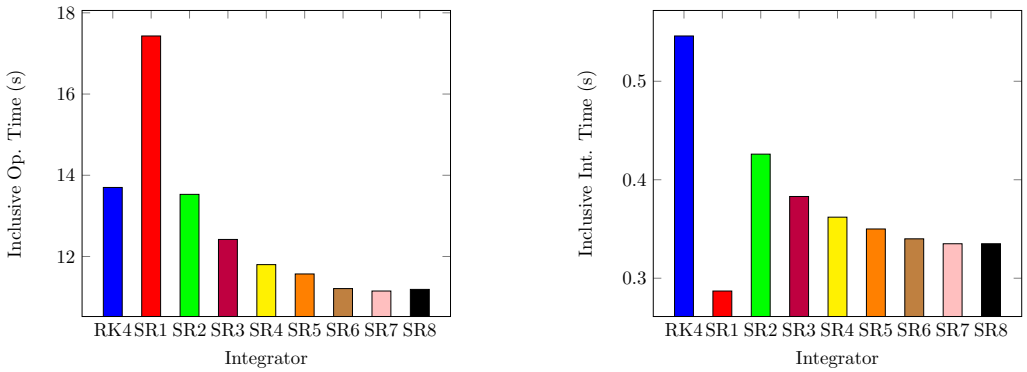
Table 4.3: Performance timings for third-order MRAB integrators.



(a) End-to-end timing.

(b) Serial RHS timing.

Figure 4.1: Plotted serial performance data - RHS, end-to-end.



(a) Serial operator timing.

(b) Serial interpolation timing.

Figure 4.2: Plotted serial performance data - operator, interpolation.

We see in Table 4.3 that the times given here correlate well with our performance model, indicating the expected speedup or slowdown in all cases. Our theoretical critical step ratio, as calculated above, matches the character of the performance results given here. Furthermore, we note in the case of serial performance profiling that the observed inclusive interpolation times are low as we would expect (when not running in parallel, this amounts to copying data between buffers) - we will see that the time spent in interpolation routines becomes more significant when we discuss parallel runs.

4.3 Parallel Performance

4.3.1 Small-Scale Runs

As with the serial runs, we here run each integrator to the same end time for various processor counts, tracking the time spent in certain subroutines and also the end-to-end wallclock time, and report the results. An important distinction to note is that we here document end-to-end wallclock time, but the times reported for certain subroutines are accumulated inclusive times (the total time spent by all processors in a given routine and all the routines it calls). In the tables that follow, green cells indicate lower time spent in a certain subroutine than that of the baseline (RK4) integrator. Plots of this data are also given in Appendix C.

Processors	RK4	SR=1	SR=2	SR=3	SR=4	SR=5	SR=6	SR=7	SR=8
2	17.15	23.19	29.59	28.10	26.45	25.36	24.75	24.34	24.37
4	13.59	17.82	10.56	10.16	10.05	9.565	9.393	9.365	9.397
8	7.202	9.009	7.271	7.128	6.826	6.713	6.764	6.719	6.715
16	4.496	5.686	4.904	4.771	4.930	4.559	4.477	4.521	4.497

Table 4.4: End-to-end performance timings for various processor counts.

Processors	RK4	SR=1	SR=2	SR=3	SR=4	SR=5	SR=6	SR=7	SR=8
2	24.372	30.952	24.15	22.34	21.15	20.57	20.21	19.96	20.04
4	27.313	34.450	27.87	25.48	24.56	23.86	23.53	23.30	23.39
8	37.116	45.344	36.27	32.67	30.79	29.73	29.64	29.11	29.14
16	46.776	57.918	47.12	42.51	41.82	39.31	38.43	38.33	38.23

Table 4.5: RHS performance timings for various processor counts.

Processors	RK4	SR=1	SR=2	SR=3	SR=4	SR=5	SR=6	SR=7	SR=8
2	14.096	17.756	13.88	12.73	12.04	11.73	11.50	11.29	11.38
4	15.232	19.394	15.40	14.16	13.58	13.23	13.08	12.87	12.99
8	18.952	24.062	18.80	17.16	16.42	15.88	15.77	15.52	15.59
16	22.948	29.170	23.16	21.04	20.58	19.54	19.21	19.10	19.10

Table 4.6: Operator performance timings for various processor counts.

Processors	RK4	SR=1	SR=2	SR=3	SR=4	SR=5	SR=6	SR=7	SR=8
2	4.732	5.137	24.82	23.35	22.90	22.16	21.94	21.79	21.98
4	18.29	15.75	2.096	2.847	4.883	5.045	5.532	6.01	6.45
8	11.28	3.742	3.611	6.119	9.023	10.43	11.86	12.60	13.36
16	13.59	1.839	5.260	8.054	13.11	14.12	15.45	16.71	17.56

Table 4.7: Interpolation performance timings for various processor counts.

Generally, what we see here is first and foremost that the accumulated inclusive time spent in RHS and operator related subroutines (Table 4.5 and Table 4.6, respectively) is almost always lowered by the use of multirate, in certain cases by up to 20%. The few exceptions occur at a step ratio of 2, where the reduction in right-hand side evaluations required is observed to be quite small based on the model proposed in Section 4.1, and the time spent in the RHS and operator routines

is quite close to that of RK4. Furthermore, the end-to-end wallclock times (Table 4.4), while not showing benefit at the 2-processor level (where load imbalance appears to dominate, as evidenced by the high wait times in interpolation), show that the reduction in right-hand side evaluations indeed leads to end-to-end speedup for the simulation run for a number of step ratios and core counts.

The inclusive time spent by all processors in interpolation-related subroutines (Table 4.7) highlights the need for a improved grid-to-grid communication - namely, in spite of the benefits we obtain from using a split send-receive communication model for the multi-rate integrators with our new SAT interpolation algorithm, we still see high inclusive times for higher step ratios—especially at higher processor counts—due to idle time spent in these subroutines waiting for other grids. We expect that for the highest processor count tested for this small-scale case (16 cores), this is what causes poor end-to-end results. In order to alleviate this issue and reduce time spent by processors waiting in MPI calls, rescaling of the decomposition (see Section 4.3.2) and/or different communication models need to be explored. For the time being, this is deferred to future work.

4.3.2 Decomposition

We note here that the usage of multi-rate integration in the overset sense naturally induces load imbalance within the application, given that in a given macro-timestep, we will be evaluating more right-hand sides on certain grids than on others. This motivates a change to PlasComCM's existing standard decomposition, which simply distributes processors to grids based on the ratio of that grid's number of points to the total number of points in the simulation. Rescaling this decomposition based on the multi-rate step ratio being used (a direct indication of how many right-hand side evaluations per macro-timestep a given processor is responsible for) can be demonstrated to have a strong influence on performance results at higher processor counts, especially regarding grid-to-grid communication. While the small-scale case and low core counts in the previous section preclude us from modifying the existing decomposition to obtain this benefit, the decomposition is appropriately scaled to produce the large-scale results included in the forthcoming section.

4.3.3 Large-Scale Runs

In order to demonstrate the efficacy of our MRAB integrators for larger problems, we will now time-march a three-grid system featuring roughly 41 million points and demonstrate performance benefit. The grids are shown in Figure 4.3, and are numerically described in Table 4.8. Note the percentage of total points in Grid 1 - this will be our slow grid, and therefore this is where the number of right-hand side evaluations required will be reduced.

Grid	Grid Type	No. of Points	% of Total
1	Cartesian	26,624,172	64.2
2	Cylindrical	13,210,890	31.9
3	Cartesian	1,626,625	3.9

Table 4.8: Description of grids for large-scale case.

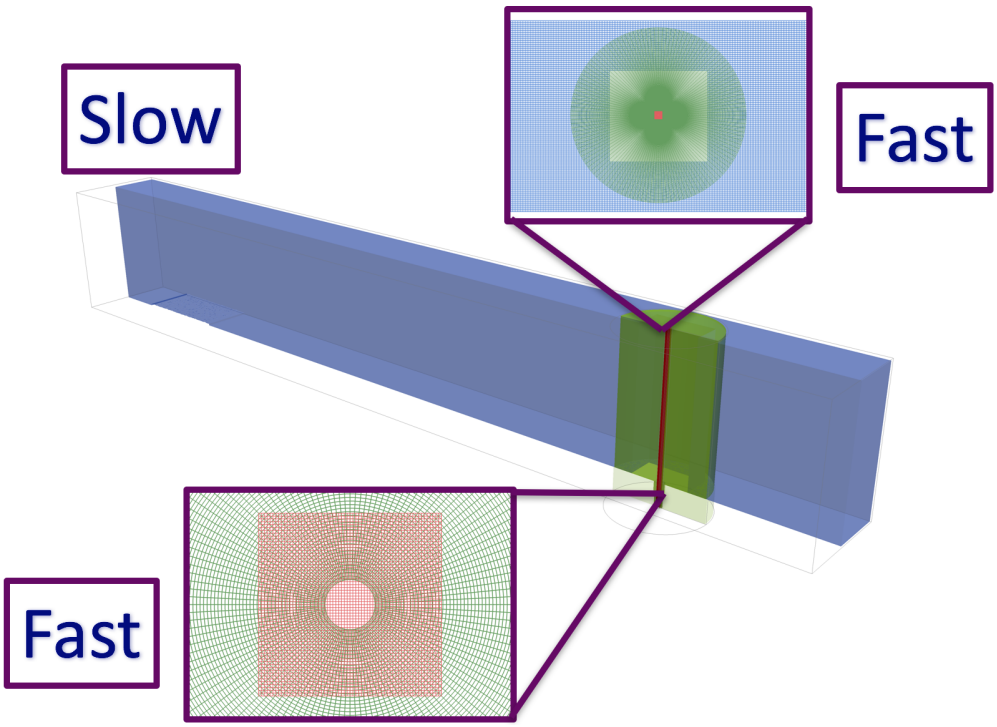


Figure 4.3: Jet-in-crossflow grid configuration for large-scale run.

The results given here are obtained from a 128-core run on Cab, a Linux commodity cluster at Lawrence Livermore National Laboratory, and (as in Section 4.3.1) document the accumulated inclusive time spent in certain PlasComCM right-hand side related subroutines, along with end-to-end wallclock time. These estimates are obtained exclusive of bootstrapping costs, instead examining the per-timestep costs of a third-order MRAB scheme with a step ratio of 2 versus that of PlasComCM's standard single-rate RK4 integrator, scaling these results for a number of timesteps such that the same end time is reached by each integrator. Based on the relative grid sizes, a performance model similar to that described in Section 4.1 is given here.

Integrator	Total RHS Evaluations	% Reduction from RK4	Est. Speedup
RK4 (PlasComCM)	165,846,748	0.00	1.00x
Single-Rate Adams-Bashforth	207,308,435	-25.00	0.80x
MRAB (Step Ratio = 2)	140,748,005	15.1	1.18x

Table 4.9: Evaluating RHS costs for MRAB integrators compared to RK4 - large-scale case.

Note that the grid sizes only allow a maximum step ratio of 2 - the resulting ideal speedup is therefore quite modest compared to the actual capabilities of Leap-generated multi-rate integrators applied to simulations with more wildly varying grid resolution. In any case, we ideally expect program speedup when marching to the same end time using our generated MRAB integrator.

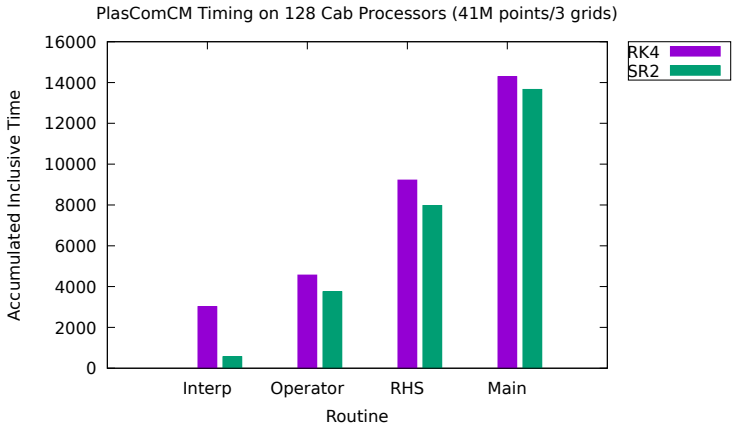


Figure 4.4: Performance timing data for large-scale case.

Generally speaking, we see here that the results show inclusive times similar to what we would expect for operator and right-hand side costs, and furthermore, we see drastic reduction in the time spent in interpolation-related subroutines - this is a direct result of implementation of multi-rate-specific overset interpolation schemes featuring selective communications and separate send-receive schemes (interleaved with useful right-hand side work). In the end, the main time marching loop sees about 5% speedup, which is below our expected ideal speedup. Looking at the accumulated inclusive times for a few other subroutines illustrates why the multi-rate code is underperforming here:

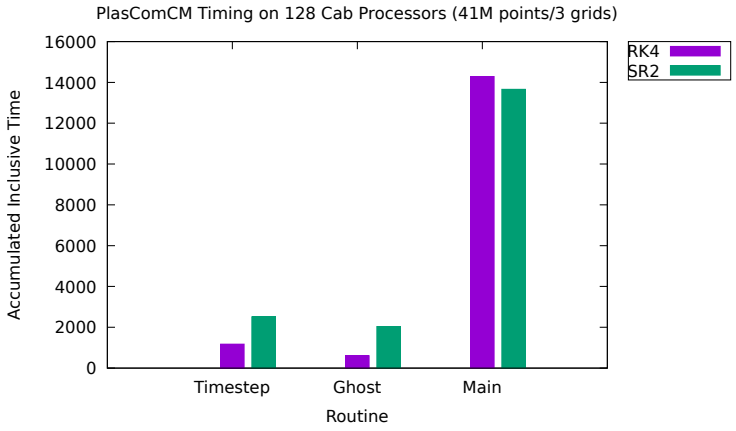


Figure 4.5: Additional timing data for large-scale case.

Our results show that time spent in timestep calculation and ghost cell updates (halo exchanges) is higher for MRAB than for RK4 - specifically, this additional time is being spent in communication-related portions of these routines, performing MPI functions. We therefore conclude that these inflated times are a direct result of the inherent load imbalance associated with the use of multirate.

Chapter 5

Conclusions

5.1 Discussion

In this study, we have developed multi-rate Adams-Bashforth (MRAB) integrators by taking advantage of an overset mesh formulation. After laying out the mathematical underpinnings of the methods at work, we demonstrated proper convergence of third and fourth order integrators on a viscous two-grid problem by comparing the result at a given time to the results of a standard Runge-Kutta integration to the same time with a sufficiently small timestep. We have also shown that the maximum stable timestep of the integrators linearly increases, eventually surpassing that of the standard RK4 integrator, but also eventually meeting a limiting point at a step ratio of slightly more than half of the speed ratio between grids. The latter result is confirmed by a number of smaller-scale numerical tests, and furthermore brief parameter study was also undertaken regarding the effects of various MRAB policy decisions on the stability of a given method, finding that the maximum stable CFL obtained remained unaffected by different orders of evaluation and the use of re-extrapolation. A study of the effect of the policy decisions on convergence characteristics is reserved for future work.

In terms of performance, we have developed a model that assumes right-hand-side evaluations of the compressible Navier-Stokes equations to compose the bulk of the work being done by the solver, and have from this model extracted relations for minimum profitable Adams-Bashforth step ratio. In running the integrators, we have found that while profitable step ratios are indeed reached (especially in terms of reduction in RHS and operator costs), the overall performance of the integrators is at times limited by inefficient communication between grids, along with the inherent

load imbalance that results from the use of multi-rate integrators. While the core counts on which the example here is tested are relatively low, we are able to demonstrate the importance of spatial decomposition in performance, motivating future work in the use of overdecomposition with AMPI (see below). We also briefly demonstrate the use of multi-rate within PlasComCM on a larger-scale jet in crossflow problem, showing overall performance benefit and identifying a number of code hotspots where the remaining load imbalance manifests.

5.2 Present and Future Work

In this section, we provide a brief summary of present and future research directions to be explored in the context of multi-rate integration on overset meshes.

5.2.1 Overdecomposition and AMPI

Adaptive MPI is an implementation of the MPI standard written on top of Charm++. Charm++ is an object-oriented parallel programming system based on C++, built on an adaptive runtime system. A Charm++ program is decomposed into parallel objects that communicate via asynchronous remote method invocation and can be migrated between nodes of a distributed system. The runtime system schedules tasks invoked on the parallel objects in a message-driven manner, which encourages programmers to over-decompose their problem into many more work units and data units than there are physical processors or cores on the target machine. Load balancers can be plugged into the runtime system to use the fact that all work and data units are migratable to dynamically balance the load across the whole system.

As suggested by the performance results above, multi-rate time integration is expected to induce load imbalance as different ranks operate on grids of different speeds. Assuming there is sufficient load imbalance, we can use one of the existing load balancing strategies included in Charm++ or write our own application specific strategies, with Dagrt passing its knowledge of the DAG through to the load balancer. The code generated by Leap and Dagrt is thread-safe and contains no global or static variables. Therefore, to use multirate time integration and AMPI together, we simply

generate the Fortran modules for multi-rate and then compile PlasComCM and the generated codes together using AMPI's compiler wrappers. At the time of this writing, limited performance runs have been performed with the two tools, using PlasComCM as the host application.

5.2.2 Multiphysics Implementation

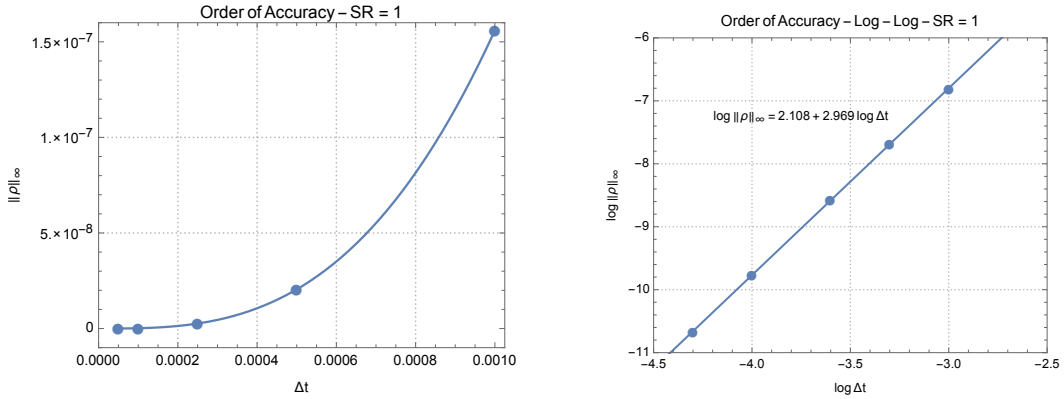
Still to come is an implementation of multi-rate integrators in the context of a compressible viscous fluid solver (like PlasComCM) that separates the timescales based on the physics dictating the rate of change of the given state, rather than using a purely spatial separation. As an example: a combustion simulation involving multiple species on a single grid could involve multiple fluids/states changing at different speeds, be it due to bulk fluid motion (and possibly turbulence) or chemical kinetics. The ability to run a multi-rate integrator on slow states and fast states rather than slow grids and fast grids is potentially critical to realizing the full potential of the method as a tool for improving the efficiency of large-scale fluid simulations, especially given that the involvement of certain physical mechanisms like plasma (the effects of which are not necessarily confined to a given grid) can greatly increase the effective timestep ratio between solution components beyond values that are practically possible for a grid-based implementation. In the future, schemes developed in a manner similar to the one described in this thesis may include the capability to distinguish timescales via both approaches, perhaps at the same time.

Appendix A

Third Order Convergence Results

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.001	1.56E-07	-3.000	-6.807
0.0005	2.05E-08	-3.301	-7.688
0.00025	2.63E-09	-3.602	-8.580
0.0001	1.71E-11	-4.000	-9.767
0.00005	2.15E-11	-4.301	-10.668

Table A.1: Convergence data for third-order MRAB, SR=1.



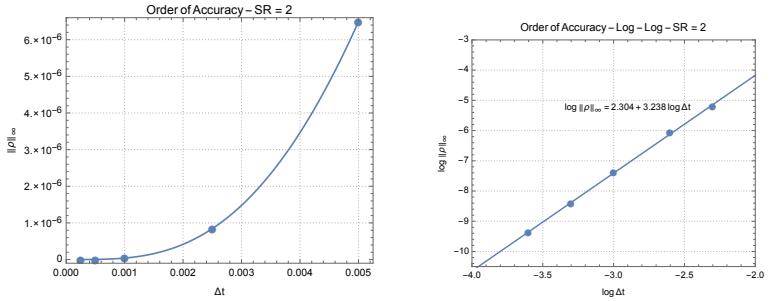
(a) Plot of density error curve for SR=1.

(b) Log-log plot for SR=1.

Figure A.1: Plotted third-order convergence data for SR=1.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.005	6.49E-06	-2.301	-5.188
0.0025	8.44E-07	-2.602	-6.074
0.001	4.07E-08	-3.000	-7.390
0.0005	4.04E-09	-3.301	-8.394
0.00025	4.26E-10	-3.602	-9.370

Table A.2: Convergence data for third-order MRAB, SR=2.



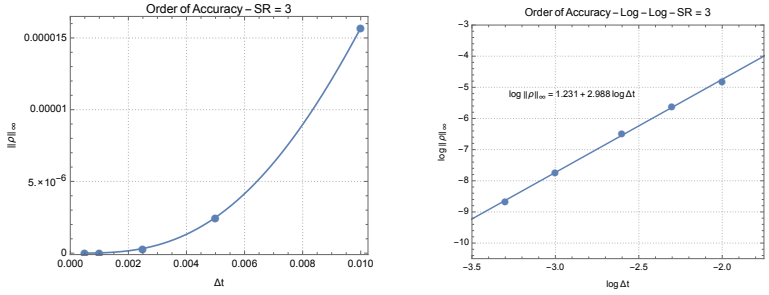
(a) Plot of density error curve for SR=2.

(b) Log-log plot for SR=2.

Figure A.2: Plotted third-order convergence data for SR=2.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.01	1.57E-05	-2.000	-4.804
0.005	2.49E-06	-2.301	-5.603
0.0025	3.26E-07	-2.602	-6.487
0.001	1.82E-08	-3.000	-7.740
0.0005	2.18E-09	-3.301	-8.661

Table A.3: Convergence data for third-order MRAB, SR=3.



(a) Plot of density error curve for SR=3.

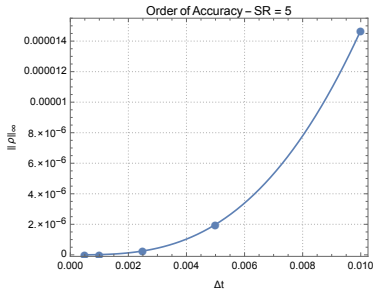
(b) Log-log plot for SR=3.

Figure A.3: Plotted third-order convergence data for SR=3.

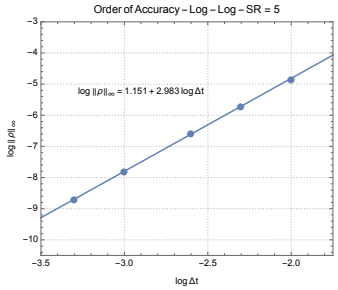
The convergence data for a step ratio of 4 is given in the main body of the thesis (Chapter 3).

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.01	1.47E-05	-2.000	-4.832
0.005	1.99E-06	-2.301	-5.701
0.0025	2.54E-07	-2.602	-6.595
0.001	1.60E-08	-3.000	-7.796
0.0005	1.96E-09	-3.301	-8.708

Table A.4: Convergence data for third-order MRAB, SR=5.



(a) Plot of density error curve for SR=5.

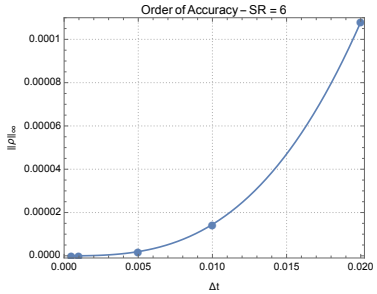


(b) Log-log plot for SR=5.

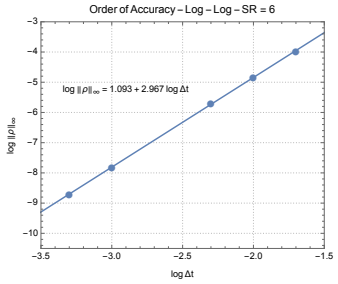
Figure A.4: Plotted third-order convergence data for SR=5.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.02	1.08E-04	-1.699	-3.967
0.01	1.45E-05	-2.000	-4.839
0.005	1.94E-06	-2.301	-5.712
0.001	1.56E-08	-3.000	-7.807
0.0005	1.93E-09	-3.301	-8.714

Table A.5: Convergence data for third-order MRAB, SR=6.



(a) Plot of density error curve for SR=6.

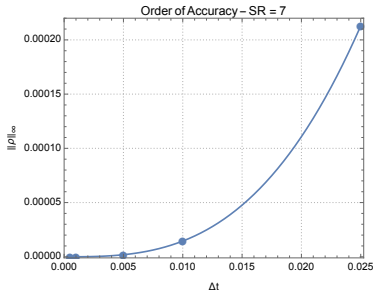


(b) Log-log plot for SR=6.

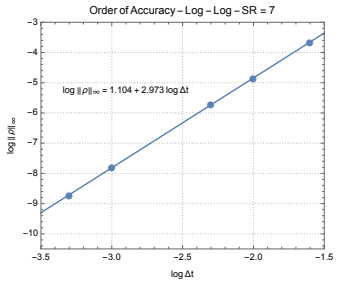
Figure A.5: Plotted third-order convergence data for SR=6.

Δt	$\|\rho\|_\infty$	log(Δt)	$\log(\|\rho\|_\infty)$
0.025	2.13E-04	-1.602	-3.672
0.01	1.45E-05	-2.000	-4.839
0.005	1.91E-06	-2.301	-5.719
0.001	1.54E-08	-3.000	-7.812
0.0005	1.91E-09	-3.301	-8.719

Table A.6: Convergence data for third-order MRAB, SR=7.



(a) Plot of density error curve for SR=7.

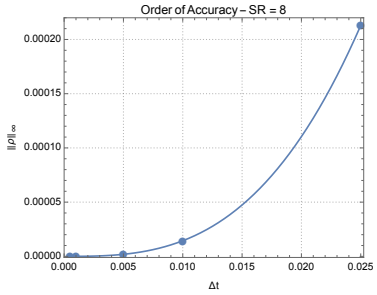


(b) Log-log plot for SR=7.

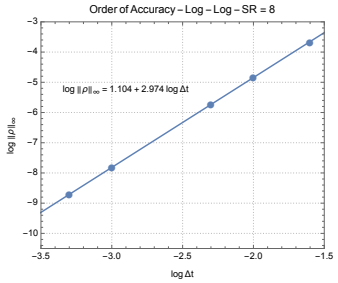
Figure A.6: Plotted third-order convergence data for SR=7.

Δt	$\|\rho\|_\infty$	log(Δt)	$\log(\|\rho\|_\infty)$
0.025	2.13E-04	-1.602	-3.671
0.01	1.44E-05	-2.000	-4.842
0.005	1.90E-06	-2.301	-5.721
0.001	1.53E-08	-3.000	-7.815
0.0005	1.90E-09	-3.301	-8.721

Table A.7: Convergence data for third-order MRAB, SR=8.



(a) Plot of density error curve for SR=8.



(b) Log-log plot for SR=8.

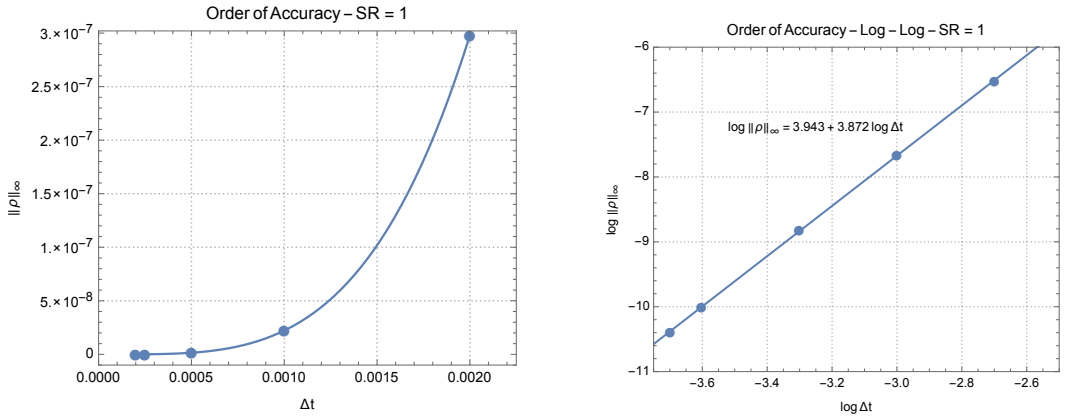
Figure A.7: Plotted third-order convergence data for SR=8.

Appendix B

Fourth Order Convergence Results

Δt	$\|\rho\|_\infty$	log(Δt)	$\log(\|\rho\|_\infty)$
0.002	2.98E-07	-2.699	-6.526
0.001	2.22E-08	-3.000	-7.654
0.0005	1.51E-09	-3.301	-8.821
0.00025	9.84E-11	-3.602	-10.007

Table B.1: Convergence data for fourth-order MRAB, SR=1.



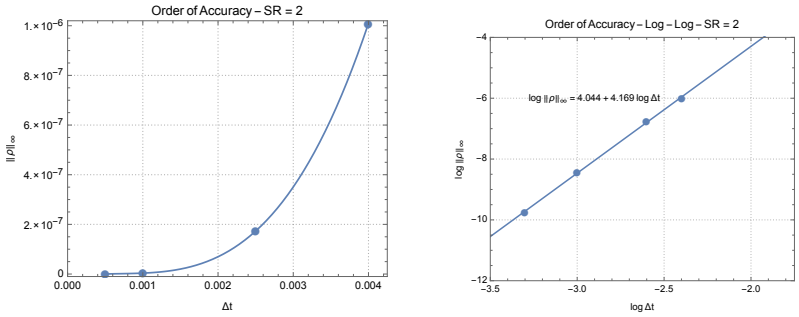
(a) Plot of density error curve for SR=1.

(b) Log-log plot for SR=1.

Figure B.1: Plotted fourth-order convergence data for SR=1.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.004	1.01E-06	-2.398	-6.000
0.0025	1.73E-07	-2.602	-6.762
0.001	3.67E-09	-3.000	-8.435
0.0005	1.79E-10	-3.301	-9.747

Table B.2: Convergence data for fourth-order MRAB, SR=2.



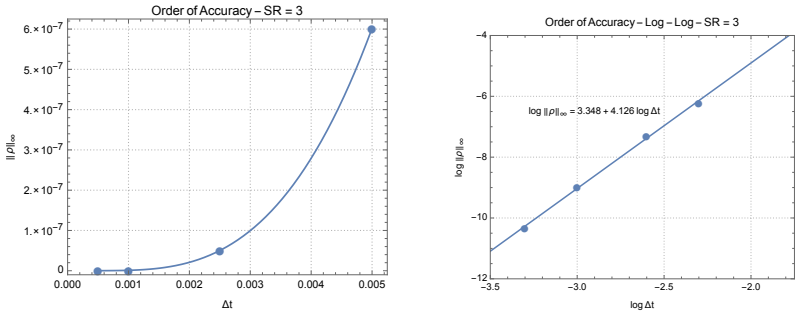
(a) Plot of density error curve for SR=2.

(b) Log-log plot for SR=2.

Figure B.2: Plotted fourth-order convergence data for SR=2.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.005	6.02E-07	-2.301	-6.220
0.0025	5.02E-08	-2.602	-7.299
0.001	1.03E-09	-3.000	-8.987
0.0005	4.70E-11	-3.301	-10.328

Table B.3: Convergence data for fourth-order MRAB, SR=3.



(a) Plot of density error curve for SR=3.

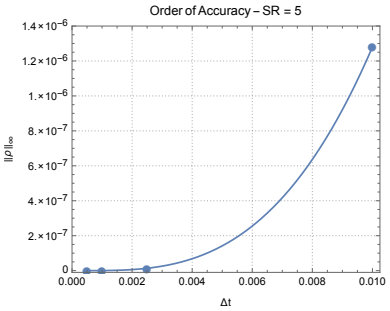
(b) Log-log plot for SR=3.

Figure B.3: Plotted fourth-order convergence data for SR=3.

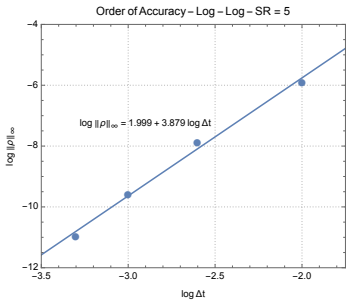
The convergence data for a step ratio of 4 is given in the main body of the thesis (Chapter 3).

Δt	$\|\rho\|_\infty$	log(Δt)	$\log(\|\rho\|_\infty)$
0.01	1.28E-06	-2.000	-5.893
0.0025	1.35E-08	-2.602	-7.870
0.001	2.64E-10	-3.000	-9.578
0.0005	1.11E-11	-3.301	-10.955

Table B.4: Convergence data for fourth-order MRAB, SR=5.



(a) Plot of density error curve for SR=5.

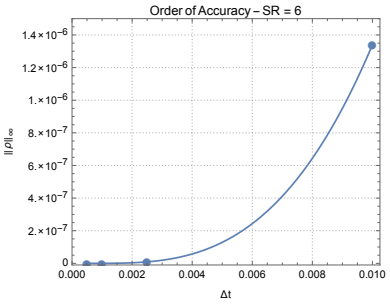


(b) Log-log plot for SR=5.

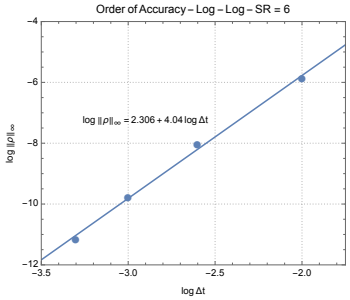
Figure B.4: Plotted fourth-order convergence data for SR=5.

Δt	$\|\rho\|_\infty$	log(Δt)	$\log(\|\rho\|_\infty)$
0.01	1.34E-06	-2.000	-5.872
0.0025	9.01E-09	-2.602	-8.045
0.001	1.74E-10	-3.000	-9.759
0.0005	7.14E-12	-3.301	-11.146

Table B.5: Convergence data for fourth-order MRAB, SR=6.



(a) Plot of density error curve for SR=6.

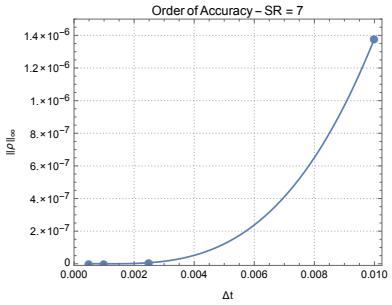


(b) Log-log plot for SR=6.

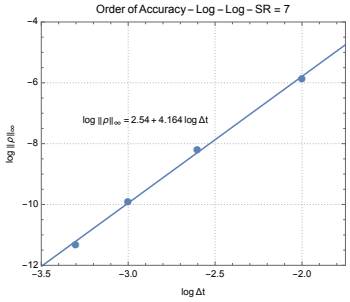
Figure B.5: Plotted fourth-order convergence data for SR=6.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.01	1.38E-06	-2.000	-5.860
0.0025	6.54E-09	-2.602	-8.184
0.001	1.25E-10	-3.000	-9.903
0.0005	5.05E-12	-3.301	-11.300

Table B.6: Convergence data for fourth-order MRAB, SR=7.



(a) Plot of density error curve for SR=7.

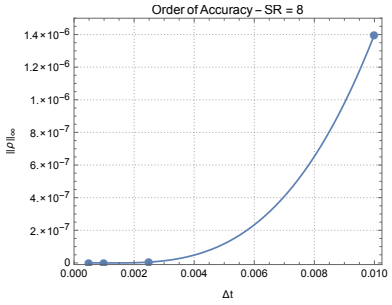


(b) Log-log plot for SR=7.

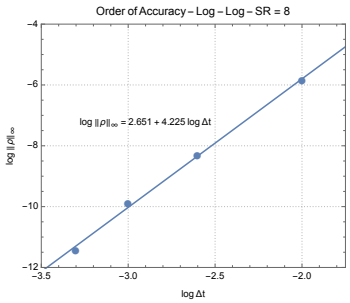
Figure B.6: Plotted fourth-order convergence data for SR=7.

Δt	$\|\rho\|_\infty$	log(Δt)	log($\|\rho\|_\infty$)
0.01	1.40E-06	-2.000	-5.854
0.0025	5.03E-09	-2.602	-8.298
0.001	1.27E-10	-3.000	-9.896
0.0005	3.83E-12	-3.301	-11.417

Table B.7: Convergence data for fourth-order MRAB, SR=8.



(a) Plot of density error curve for SR=8.



(b) Log-log plot for SR=8.

Figure B.7: Plotted fourth-order convergence data for SR=8.

Appendix C

Parallel Performance Plots

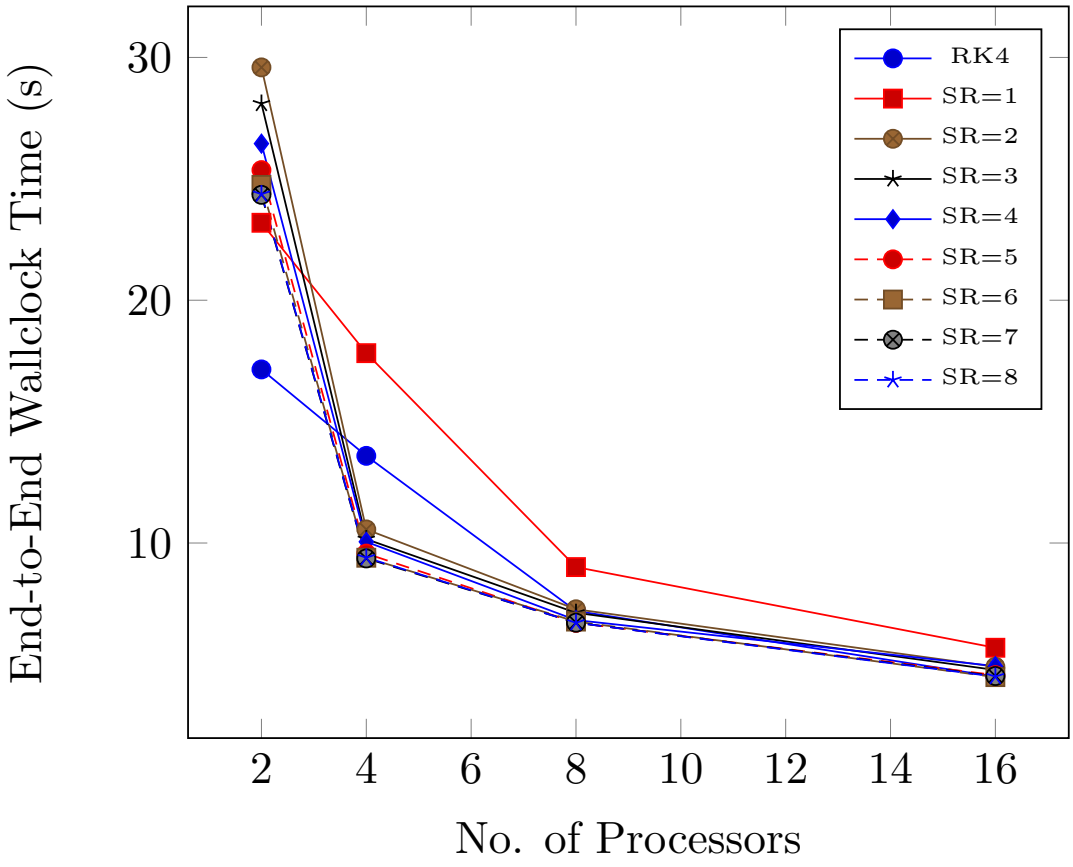


Figure C.1: Plotted end-to-end wallclock time for third-order MRAB integrators at various processor counts.

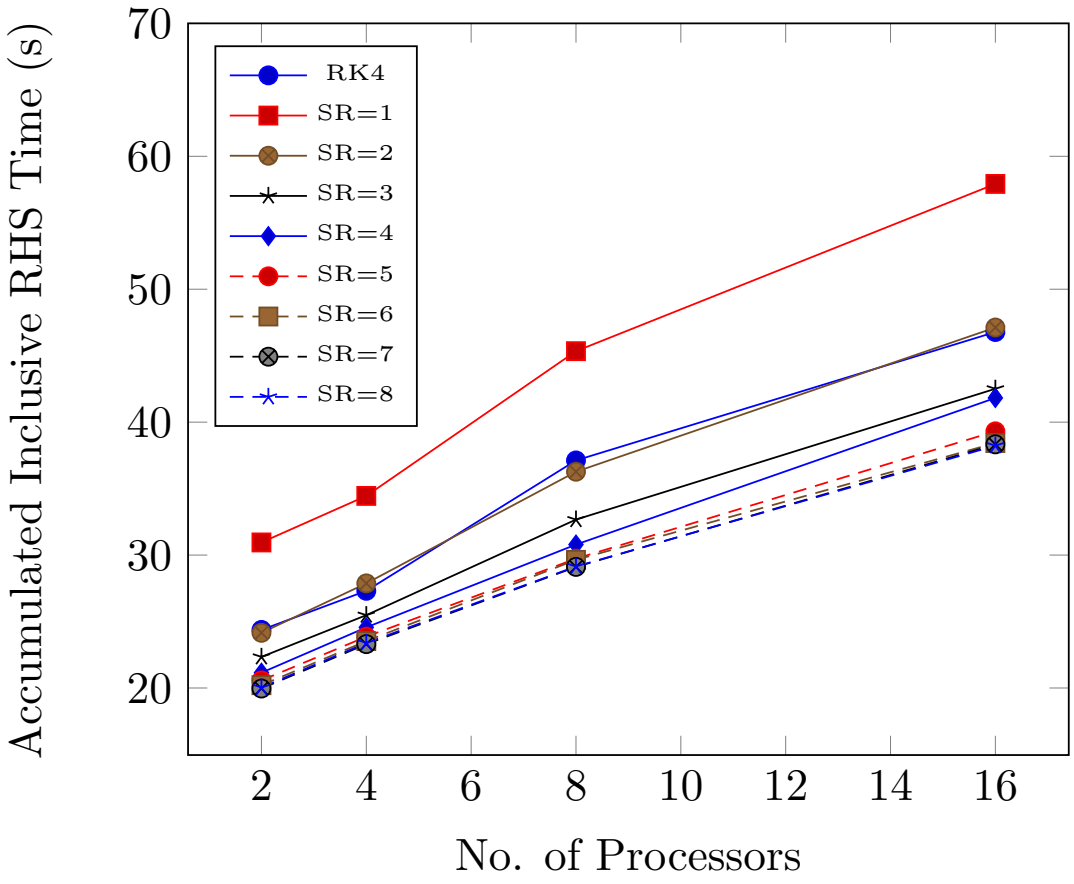


Figure C.2: Plotted accumulated inclusive RHS time for third-order MRAB integrators at various processor counts.

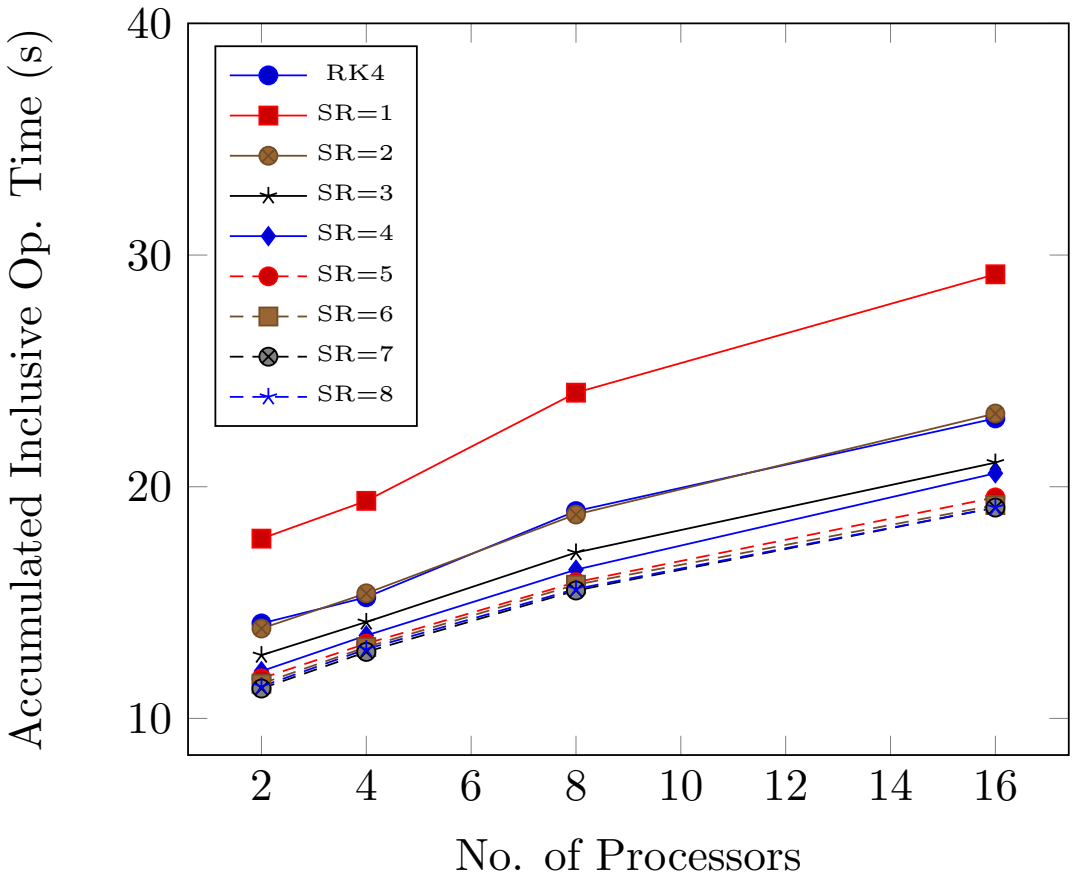


Figure C.3: Plotted accumulated inclusive operator time for third-order MRAB integrators at various processor counts.

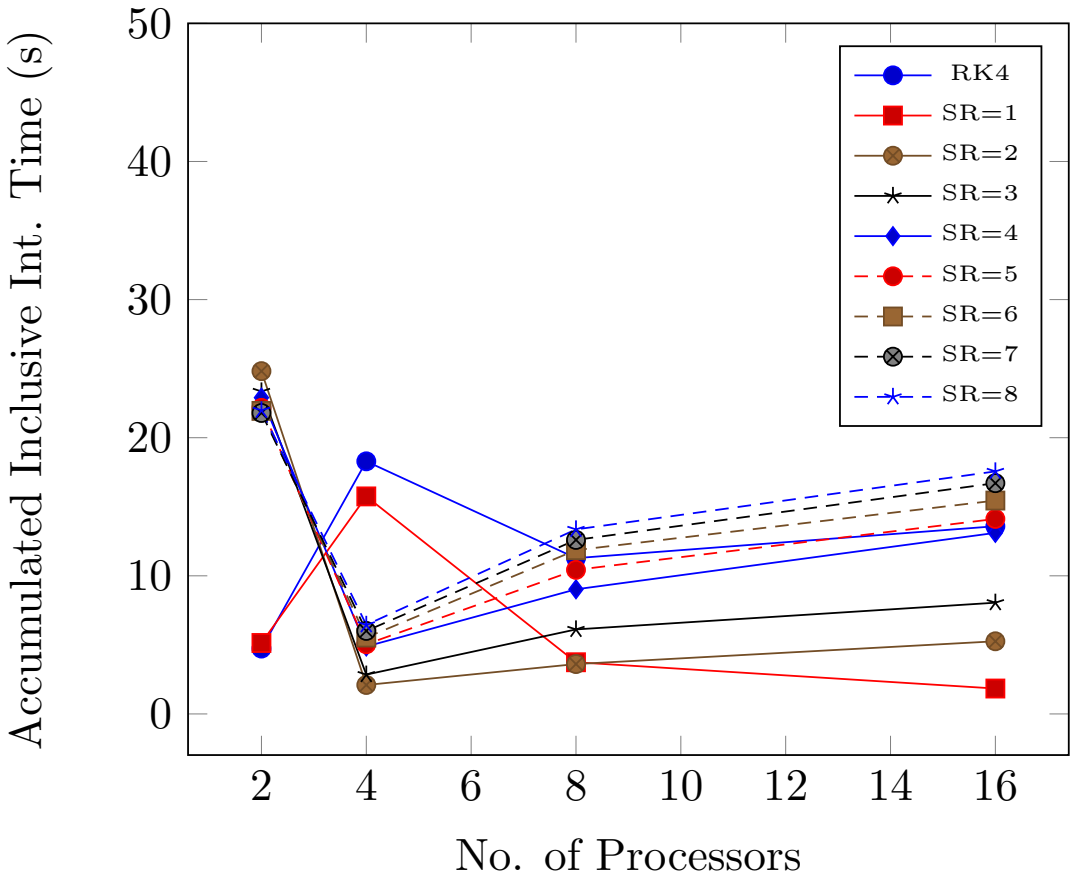


Figure C.4: Plotted accumulated inclusive interpolation time for third-order MRAB integrators at various processor counts.

Bibliography

[1] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational fluid dynamics and heat transfer*, Washington: Hemisphere, (1984).

[2] J. Andrus, *Stability of a multi-rate method for numerical integration of ode's*, Computers & Mathematics with applications, 25 (1993), pp. 3–14.

[3] J. F. Andrus, *Numerical solution of systems of ordinary differential equations separated into subsystems*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 605–611.

[4] J. Benek, J. Steger, and F. C. Dougherty, *A flexible grid embedding technique with application to the euler equations*, in 6th Computational Fluid Dynamics Conference Danvers, 1983, p. 1944.

[5] D. J. Bodony, *Accuracy of the simultaneous-approximation-term boundary condition for time-dependent problems*, Journal of Scientific Computing, 43 (2010), pp. 118–133.

[6] D. J. Bodony, G. Zagaris, A. Reichert, and Q. Zhang, *Provably stable overset grid methods for computational aeroacoustics*, Journal of Sound and Vibration, 330 (2011), pp. 4161–4179.

[7] M. H. Carpenter, D. Gottlieb, and S. Abarbanel, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes*, tech. report, NASA, 1993.

[8] H. Chou and J. Ekaterinaris, *A compact high-order cfd package for the flow solver overflow*, in 41st Aerospace Sciences Meeting and Exhibit, 2003, p. 1234.

[9] E. M. Constantinescu and A. Sandu, *Multirate timestepping methods for hyperbolic conservation laws*, Journal of Scientific Computing, 33 (2007), pp. 239–278.

[10] C. Dawson and R. Kirby, *High resolution schemes for conservation laws with locally varying time steps*, SIAM Journal on Scientific Computing, 22 (2001), pp. 2256–2281.

[11] C. Engstler and C. Lubich, *Multirate extrapolation methods for differential equations with different time scales*, Computing, 58 (1997), pp. 173–185.

[12] C. Gear, *Multirate methods for ordinary differential equations*, tech. report, Illinois Univ., Urbana (USA). Dept. of Computer Science, 1974.

[13] C. W. Gear and D. Wells, *Multirate linear multistep methods*, BIT Numerical Mathematics, 24 (1984), pp. 484–502.

[14] M. Günther, A. Kvaernø, and P. Rentrop, *Multirate partitioned runge-kutta methods*, BIT Numerical Mathematics, 41 (2001), pp. 504–514.

[15] M. Günther and P. Rentrop, *Multirate row methods and latency of electric circuits*, Applied Numerical Mathematics, 13 (1993), pp. 83–102.

[16] A. Klockner, *High-performance high-order simulation of wave and plasma phenomena*, PhD thesis, Brown University, 2010.

[17] Y. Lee and J. Baeder, *High-order overset method for blade vortex interaction*, in 40th AIAA Aerospace Sciences Meeting & Exhibit, 2002, p. 559.

[18] R. Magnus and H. Yoshihara, *Inviscid transonic flow over airfoils*, AIAA Journal, 8 (1970), pp. 2157–2162.

[19] K. Mattsson, M. Svärd, and J. Nordström, *Stable and accurate artificial dissipation*, Journal of Scientific Computing, 21 (2004), pp. 57–79.

[20] R. Noack and J. Slotnick, *A summary of the 2004 overset symposium on composite grids and solution technology*, in 43rd AIAA Aerospace Sciences Meeting and Exhibit, 2005, p. 921.

[21] S. Osher and R. Sanders, *Numerical approximations to nonlinear conservation laws with locally varying time and space grids*, Mathematics of computation, 41 (1983), pp. 321–336.

[22] T. H. Pulliam and D. Chaussee, *A diagonal form of an implicit approximate-factorization algorithm*, Journal of Computational Physics, 39 (1981), pp. 347–363.

[23] A. Sandu and E. M. Constantinescu, *Multirate explicit adams methods for time integration of conservation laws*, Journal of Scientific Computing, 38 (2009), pp. 229–249.

[24] V. Savcenco, W. Hundsdorfer, and J. Verwer, *A multirate time stepping strategy for stiff ordinary differential equations*, BIT Numerical Mathematics, 47 (2007), pp. 137–155.

[25] B. Seny, J. Lambrechts, R. Comblen, V. Legat, J.-F. Remacle, et al., *Multirate time stepping methods for accelerating explicit discontinuous galerkin computations.*, in 9th International workshop on Multiscale (Un)-structured mesh numerical Modeling for coastal, shelf, and global ocean dynamics, 2010.

[26] 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*, Journal of Computational Physics, 256 (2014), pp. 135–160.

[27] N. Sharan, *Time-stable high-order finite difference methods for overset grids*, PhD thesis, University of Illinois at Urbana-Champaign, 2016.

[28] N. Sharan, C. Pantano, and D. J. Bodony, *Energy stable overset grid methods for hyperbolic problems*, in 7th AIAA Theoretical Fluid Mechanics Conference, 2014, p. 2924.

[29] S. Sherer and M. Visbal, *Implicit large eddy simulations using a high-order overset grid solver*, in 34th AIAA Fluid Dynamics Conference and Exhibit, 2004, p. 2530.

[30] S. Sherer, M. Visbal, and M. Galbraith, *Automated preprocessing tools for use with a high-order overset-grid algorithm*, in 44th AIAA Aerospace Sciences Meeting and Exhibit, 2006, p. 1147.

[31] S. E. Sherer and J. N. Scott, *High-order compact finite-difference methods on general overset grids*, Journal of Computational Physics, 210 (2005), pp. 459–496.

[32] J. Steger, *The chimera method of flow simulation*, in Workshop on applied CFD, Univ of Tennessee Space Institute, vol. 188, 1991.

[33] A. Stock, *Development and application of a multirate multistep ab method to a discontinuous galerkin method based particle in cell scheme*, 2009.

[34] B. Strand, *Summation by parts for finite difference approximations for d/dx*, Journal of Computational Physics, 110 (1994), pp. 47–67.

[35] N. Suhs, S. Rogers, and W. Dietz, *Pegasus 5: an automated pre-processor for overset-grid cfd*, in 32nd AIAA Fluid Dynamics Conference and Exhibit, 2002, p. 3186.

[36] M. Svärd, M. H. Carpenter, and J. Nordström, *A stable high-order finite difference scheme for the compressible navier–stokes equations, far-field boundary conditions*, Journal of Computational Physics, 225 (2007), pp. 1020–1038.

[37] M. Svärd and J. Nordström, *A stable high-order finite difference scheme for the compressible navier–stokes equations: no-slip wall boundary conditions*, Journal of Computational Physics, 227 (2008), pp. 4805–4824.

[38] H.-z. Tang and G. Warnecke, *High resolution schemes for conservation laws and convection-diffusion equations with varying time and space grids*, Journal of computational mathematics, (2006), pp. 121–140.

[39] E. A. Volkov, *The method of composite meshes for finite and infinite regions with piecewise smooth boundary*, Trudy Matematicheskogo Instituta imeni VA Steklova, 96 (1968), pp. 117–148.