Accelerated Temporal Schemes for High-Order Unstructured Methods

Siavash Hedayati Nasab

A Thesis

in

The Department

of

Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

October 2021

Abstract

Accelerated Temporal Schemes for High-Order Unstructured Methods

Siavash Hedayati Nasab, Ph.D.

Concordia University, 2021

The ability to discretize and solve time-dependent Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs) remains of great importance to a variety of physical and engineering applications. Recent progress in supercomputing or high-performance computing has opened new opportunities for numerical simulation of the partial differential equations (PDEs) that appear in many transient physical phenomena, including the equations governing fluid flow. In addition, accurate and stable space-time discretization of the partial differential equations governing the dynamic behavior of complex physical phenomena, such as fluid flow, is still an outstanding challenge. Even though significant attention has been paid to high and low-order spatial schemes over the last several years, temporal schemes still rely on relatively inefficient approaches.

Furthermore, academia and industry mostly rely on implicit time marching methods. These implicit schemes require significant memory once combined with high-order spatial discretizations. However, since the advent of high-performance general-purpose computing on GPUs (GPGPU), renewed interest has been focused on explicit methods. These explicit schemes are particularly appealing due to their low memory consumption and simplicity of implementation.

This study proposes low and high-order optimal Runge-Kutta schemes for FR/DG high-order spatial discretizations with multi-dimensional element types. These optimal stability polynomials improve the stability of the numerical solution and speed up the simulation for high-order element types once compared to classical Runge-Kutta methods. We then develop third-order accurate Paired Explicit Runge-Kutta (P-ERK) schemes for locally stiff systems of equations. These third-order P-ERK schemes allow Runge-Kutta schemes with different number of active stages to be assigned based on local stiffness criteria, while seamlessly pairing at their interface. We then generate families of schemes optimized for the high-order flux reconstruction spatial discretization. Finally, We propose optimal explicit schemes for Ansys^{TM} Fluent finite volume density-based solver, and we investigate the effect of updating and freezing reconstruction gradient in intermediate Runge-Kutta schemes. Moreover, we explore the impact of optimal schemes combined with the updated gradients in scale-resolving simulations with Fluent's finite volume solver. We then show that even though freezing the reconstruction gradients in intermediate Runge-Kutta stages can reduce computational cost per time step, it significantly increases the error and hampers stability by limiting the time-step size.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisor Prof. Brian C. Vermeire, for his consistent support and his insightful comments. He is truly an inspirational teacher in discrete mathematics and high-order computational fluid dynamics and every aspect of my life.

I would like to thank Ansys$^{\text{TM}}$ and MITACS for having an internship in Fluent's research and development department, where I had a wonderful and valuable experience. I am especially grateful to Dr. Jean-Sébastien Cagnone for his assistance, guidance and insight in the finite volume method and Ansys Fluent solver. Also, I am thankful to John Stokes for his support during my internship at Ansys.

I am grateful to my friends and my fellow labmates in the Computational Aerodynamics Lab at Concordia. Our friendship, above all was the greatest joy during my studies.

Last but not least, my deepest thanks go to my parents, who have always been there for me whenever I needed them. I will be grateful forever for their love and support.

Contents

List of Figures

List of Tables

List of Symbols

Acronyms

AIMEX	Accelerated Implicit Explicit Runge-Kutta
AoA	Angle of Attack
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrich-Lewy Number
CPU	Central Processing Unit
DG	Discontinuous Galerkin
DOF	Degrees of Freedom
DNS	Direct Numerical Simulation
FDM	Finite Difference Method
FEM	Finite Element Method
FLOPS	Floating Point Operations per Second
FR	Flux Reconstruction
FVM	Finite Volume Method
GMRES	Generalized Minimal RESidual method
GPU	Graphical Processing Unit
ILES	Implicit Large Eddy Simulation
IMEX	Implicit Explicit Runge-Kutta
LES	Large Eddy Simulation

MUSCL	Monotonic Upstream-Centered Scheme for Conservation Laws
NASA	National Aeronautic and Space Administration
PDE	Partial Differential Equation
P-ERK	Paired Explicit Runge-Kutta
RANS	Reynolds Averaged Navier-Stokes
RK	Runge-Kutta
SD	Spectral Difference
SDIRK	Singly-Diagonally Implicit Runge-Kutta
SGS	Subgrid Scale
SV	Spectral Volume
TGV	Taylor Green Vortex
TVD	Total Variation Diminishing

Greek Letters

$\boldsymbol{\alpha}$	Advection Velocity
γ	Specific Heat Ratio
δ_{ij}	Kronecker delta
Δt	time-step size
Δx	Grid space size
ϵ	Turbulent kinetic energy dissipation rate
κ	Heat conductivity
μ	Dynamic viscosity
ν	Kinematic viscosity
ν_t	Eddy viscosity
$\boldsymbol{\xi}$	Coordinate vector in reference space
τ	Viscous shear stress

ρ	Density
η	Kolmogorov length scale
Ω	Volume of computational domain

Roman Letters

C_d	Drag coefficient
C_l	Lift coefficient
C_p	Pressure Coefficient
D	Diameter
e	Internal energy
E_k	Turbulent kinetic energy
$\mathbf{f}$	External force
F	Flux vector
k	Solution polynomial degree
M	Semi-discret matrix
Ma	Mach number
m	Mass
N_e	Number of elements
N_{DOF}	Number of degrees of freedom
P_0	Pressure
$\mathbf{P}$	Momentum
P	Stability polynomial
p	Order of accuracy in time
Pr	Prandtl number
Q	Conserved variable
q	Heat transfer

$\overline{R}$	Gas constant
Re	Reynolds number
$\mathbf{u}$	Velocity vector
S	Source term
$\mathbb{S}$	Control surface
t	Time
$\mathbf{x}$	Coordinate vector in physical space
W	Work done by the system

Mathematical Operators

∇	Nabla operator
∂	Partial derivative
$\lvert\bullet\rvert$	Magnitude of a vector or modulus of complex number
$\otimes$	Cross product of vectors

Superscripts and Subscripts

$\bullet_\infty$	Free-stream condition
$\bullet_i$	Numerator of coordinates in tensor notation
$\bullet_j$	Numerator of coordinates in tensor notation
$\bullet^T$	Transpose

Chapter 1

Introduction

This chapter provides a brief summary and background relevant to this thesis. It begins with an introduction to Computational Fluid Dynamics (CFD) and lays out the necessary elements of a CFD simulation, including numerical schemes and turbulence models. The objective of this study is to develop accurate and efficient temporal schemes. Therefore, we provide a brief background on time marching methods and a framework for developing new schemes. A comprehensive review of temporal schemes is presented in Chapter 3.

1.1 Computational Fluid Dynamics

Numerical approximations to the solutions of the governing equations describing fluid flow and heat transfer have been an ongoing pursuit for mathematicians and engineers. Recently, via the development of powerful processors and super computers, this approach has become practical. The overall objective of Computational Fluid Dynamics, CFD, is to take the continuous governing equations of fluid flow and generate an approximate solution on a discrete domain [2]. CFD is now one of the primary tools, in addition to experimental and theoretical methods for solving fluid-dynamics problems. CFD has several advantages compared to other methods. It can be applied to complex geometries and to problems for which analytical solutions are not available. Moreover, it provides datasets that cannot be obtained via experimental approaches [2]. Through ongoing improvements of computer hardware, CFD is now applied in many diverse fields, including engineering, physics, and meteorology.

In CFD, we discretize the fluid domain by generating meshes and transforming our linear or non-linear Partial Differential Equations (PDE) to coupled algebraic equations [3]. Discretization is the process whereby the domain is subdivided into elements. The equations are expressed in a discrete form on each element by using a spatial discretization method such as the finite difference method (FDM), finite element method (FEM), finite volume method (FVM), discontinuous Galerkin method (DGM) [1, 4], spectral volume (SV) [5], or flux reconstruction approach (FR) [6, 7]. The FDM requires a structured grid arrangement and can achieve high-order accuracy by including a larger number of grid points in stencils to compute derivatives at the solution points [8]. The FEM consist of a globally continuous solution, where adjacent elements share the same value at their interface. The FVM is geometrically flexible and can be formulated to use both structured and unstructured grids. The DGM combines the element-wise approach of FEM with the localized solution properties of the FVM [1, 4]. The DGM uses the integral form of the conservation equations. However, recently, the FR approach introduced by Huynh [6, 7], unifies several of these high-order unstructured methods into one framework [9, 10].

The chaotic behavior of turbulent flows remains poorly understood. There are a variety of approaches for modelling the turbulence. The Reynolds-averaged Navier-Stokes (RANS) equations are derived by decomposing the velocity into time-average and time-fluctuating components [11]. Another approach is large-eddy simulation (LES), which solves the spatially filtered Navier-Stokes equations [11]. A third approach to simulate turbulent flows is the direct numerical simulation (DNS), which solves the Navier-Stokes equations on a mesh that is fine enough to resolve all turbulent length scales and a time-step size small enough to capture all turbulent time scales. Unfortunately, DNS is typically limited to simple geometries and low Reynolds number flows, because of the limited computing capabilities of even the most potent modern supercomputers.

1.2 Turbulence Modeling

The majority of flows that we see in daily life are turbulent [12, 13]. The first step in modelling turbulent flows is understanding their fundamental nature. Turbulence is characterized by the chaotic, nonlinear behavior of fluid particles in space and time. Turbulent flows are very sensitive to initial conditions, such that even small perturbations to these initial conditions lead to significant changes to flow structures at a later time. Turbulence is also characterized by a wide range of length and time scales, varying from large to small vortices [14], and turbulent models use a description of turbulence based on these scales. The relationship between the size of the largest and the smallest eddies is called the Kolmogorov length scale and described by [15]

$$\eta = \left(\frac{\nu^3}{\epsilon}\right)^{\frac{1}{4}}, \tag{1.1}$$

where ϵ is the average rate of dissipation of turbulent kinetic energy per unit mass, and ν is the kinematic viscosity of the fluid. Similarly, the Kolmogorov time scale is defined as [15]

$$t_\eta = \left(\frac{\nu}{\epsilon}\right)^{\frac{1}{2}}. \tag{1.2}$$

DNS is a turbulence modelling method that captures all turbulent eddies down to the Kolmogorov length and time scales. In a three-dimensional domain, the number of degrees of freedom required to capture turbulence with DNS resolution is [16]

$$N_{DOF} = \left(\frac{L}{\eta}\right)^3 \approx \left(\frac{L}{\eta}\right) Re^{\frac{3}{4}}, \tag{1.3}$$

where L is a characteristic length scale and Re is the Reynolds number ($Re = uL/\nu$) of the flow. Moreover, according to Davidson [16], we can find a relationship between the ratio of these length scales and Reynolds number as

$$\frac{L}{\eta} \sim Re^{\frac{3}{4}}. \tag{1.4}$$

In addition, a similar relationship between the time scales and Reynolds number can be derived as [16]

$$\frac{t_L}{t_\eta} = Re^{\frac{1}{2}}, \tag{1.5}$$

where t_L is the characteristic time scale in the flow. This shows that a larger Reynolds number leads to a smaller Kolmogorov time scale, and consequently, a smaller required time-step size. This brings up the importance of efficient, reliable and stable temporal schemes for the simulation of turbulent flows. In fact, we are always looking for time marching methods that give the most accurate results at the lowest computational cost.

In an effort to avoid the computational cost of the DNS, scientists and engineers developed the RANS approach to model turbulent flows. The RANS consists of the time-averaged Navier-Stokes equations that decompose all the quantities of the flow into averaged and fluctuation terms as [14, 17]

$$\psi = \overline{\psi} + \psi', \tag{1.6}$$

where ψ is an arbitrary quantity of the flow, ψ' is fluctuating term and $\overline{\psi}$ is the time-averaged term that is defined as

$$\overline{\Psi} \equiv \frac{1}{\bar{t}} \int_{\bar{t}} \Psi(x,t)dt, \tag{1.7}$$

where $\bar{t}$ is the period of time over which averaging operations are evaluated.

By taking time-averaged of momentum equations we obtain

$$\rho\left(\frac{\partial \overline{u_i}}{\partial t} + \overline{u}_j \frac{\partial \overline{u_i}}{\partial x_j}\right) = -\frac{\partial \overline{P_0}}{\partial x_j} + \mu \frac{\partial^2 \overline{u_i}}{\partial x_j \partial x_j} - \frac{\partial}{\partial x_j}\left(\rho \overline{u_i' u_j'}\right), \tag{1.8}$$

where the new terms $\rho\overline{u_i' u_j'}$, are called the Reynolds stresses. Equation 1.8 cannot be solved without additional assumptions about these Reynolds stresses. For example, using the Boussinesq hypothesis

$$-\overline{u_i' u_j'} = \nu_t\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right) - \frac{1}{3}E_k\delta_{ij}, \tag{1.9}$$

where $E_k = \frac{1}{2}\overline{u_i'u_i'}$ is turbulent kinetic energy, $\delta_{i,j}$ is Kronecker delta, and ν_t is referred to as the eddy viscosity. A variety of models have been formulated, for the eddy viscosity and turbulent kinetic energy (ν_t and E_k), such as Spallart-Allmaras [18], $k - \epsilon$ [14], $k - \omega$ [14], etc.

While RANS models can be applied successfully to a wide range of problems, they often fail to capture pertinent physics of turbulence in many practical scenarios, including shock waves, turbulent transition and flow separation. Hence, an intermediate approach called LES [19] was developed. In LES, only the largest scale eddies are computed down to a cut-off scale, which is usually defined by the grid size, as shown in Figure 1.1, and the smaller scale eddies are modelled through a sub-grid scale (SGS) model. LES generally has higher accuracy compared to RANS and lower computational cost compared to DNS [20].

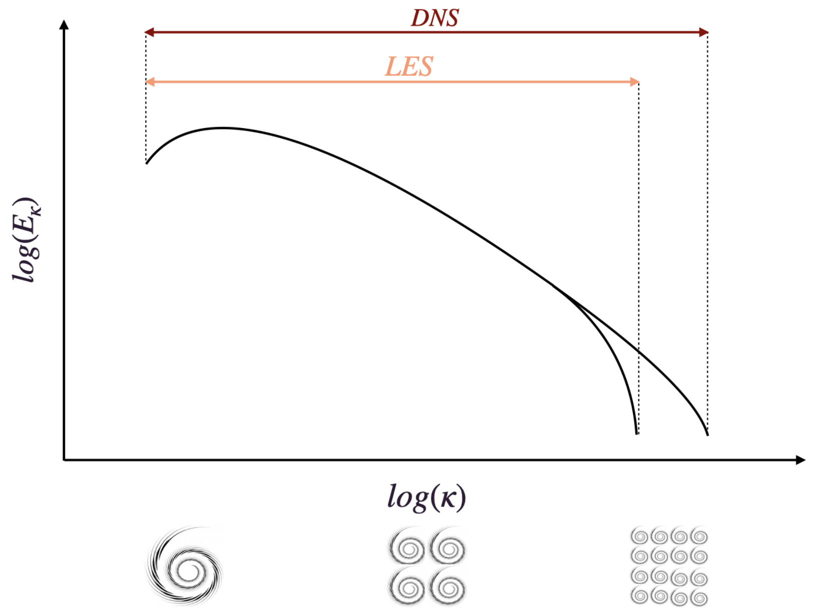


Figure 1.1. Turbulent kinetic energy cascade.

1.2.1 Implicit Large Eddy Simulation

LES models turbulent flows by approximating local, spatial averages of the flow. In SGS modeled LES methods, the large scale eddies are resolved, and smaller scale eddies are modeled. Classical (explicit) SGS LES models have been explored for many years [21]. However, numerical truncation error arising from discretization can be similar in form and magnitude to these SGS models. Hence, a natural alternative to SGS modeled LES is to use the numerical dissipation of the discretization to model the dissipation that takes place at the unresolved scales [22]. This new approach, referred to as implicit LES or ILES, was first introduced by Boris et al. [23] and has been recommended by several authors [24]. The application and performance of ILES have been tested for wall-bounded flows [25, 26, 27]. Furthermore, ILES has been applied to a variety of different spatial discretizations including the FDM [28, 29, 30], FVM [25, 31, 32], DGM [33, 34], SD [35], and FR approach [36] . In the scope of current work, since we are interested in utilizing high-order methods, it has been shown that numerical errors act as a dissipative model for LES [36, 37]. Therefore, we choose to use ILES for all of the turbulent simulations to avoid overdissipation.

1.3 Time Marching Schemes

Most physical phenomena are expressed in terms of time-dependent differential equations. This requires, the ability to reliably predict the state of a trajectory at later times from a given system of differential equations along with initial conditions. These differential equations rarely have analytical solutions. Hence, accurate and reliable numerical time integrators are a necessary part of any CFD solver. There are three attributes of an initial value problem that have to be taken into account, first the existence of the solution, then, if the solution exists, is it unique, and lastly, the sensitivity of the solution to the small perturbations to the initial condition [38]. Even though, mathematical proof of these three criteria is well beyond the scope of this study, it is important to address these criteria when a temporal scheme is being designed.

In CFD, a time-dependent conservation law can be expressed as

$$\frac{du}{dt} = R(u), \tag{1.10}$$

where $R(u)$ is the semi-discrete space operator, and u is the conserved quantity. The simplest approach to solving such an equation is the explicit Euler method, whereby we examine the state of the unknown at a later time by knowing the current state and its slope. Another necessary requirement to solve these types of equations is the choice of time-step size, often limited by a Courant-Friedrichs-Lewy (CFL) condition [39]. The CFL condition for a wave propagation problem is

$$CFL = \frac{U\Delta t}{\Delta x} \leq C_{max}, \tag{1.11}$$

where U is wave propagation velocity, Δt is the time-step size, Δx is the grid spacing, and C_{max} is the maximum allowable value of the CFL number that depends on the numerical methods used to discretize the governing equation.

1.3.1 Stiff Systems of Equations

Stiffness is a subtle but important concept in the numerical integration of systems of PDEs. A differential equation is called stiff if the solution varies slowly in time, but there are nearby solutions that vary rapidly. Therefore, the temporal scheme must take a large number of small time-steps to remain stable to reach the desired final time. For a standard CFD problem, in addition to differential equations and their initial conditions, stiffness can also depend on the geometry of the computational domain, the spatial discretization and grid size, and the physics of the problem. In fact, the numerical stiffness in CFD can be categorized in four major category [3]

- Geometry induced stiffness,
- Mesh induced stiffness,
- Flow induced stiffness,
- Discretization stiffness.

Geometry induced stiffness arises in the vicinity of complex geometries, such as the boundary layers of complex bodies. Discretizing the domain is another major source of stiffness. This is caused by small elements that often appear in local regions of the domain during grid generation. Stiffness also arises from the physics of a problem. Small time-step sizes are required for high Reynolds number flows, shock waves and reactions, amongst others. Lastly, some spatial discretizations are inherently stiffer. For instance, the second-order upwind FV scheme has stricter stability constraints when compared to the first-order upwind FV scheme.

1.3.2 Runge-Kutta Methods

The idea of generalizing the Euler method by allowing for a number of evaluations of the derivative to take place in a single step, leads to a family of the methods referred to as Runge-Kutta (RK) methods [38]. The development of RK methods can be traced back to the 1890s [38]. However, with exponential growth in computational resources, renewed attention has been paid to these methods over the past several decades [38]. Early studies on RK methods were limited to explicit schemes, but interest later moved to implicit methods, which are often more appropriate for stiff systems of equations. Several approaches have been developed for the analysis of RK methods. In this work, use the framework developed by John Butcher [38]. RK schemes are categorized into explicit, implicit and implicit-explicit (IMEX) schemes. Explicit schemes posses low cost per time-step and strict stability limits. However, implicit schemes have higher stability with a higher cost per time-step. RK methods will be discussed in detail in Chapter 3.

1.4 High-Order Unstructured Methods

To capture the pertinent physics of wall-bounded turbulent flows, in addition to efficient temporal schemes, we require to utilize an appropriate spatial discretization. Low-order spatial discretizations are methods with up to second-order accuracy and have been widely used in academia and industry. However, transitional flows share many features with wave propagation phenomena, for which high-order accuracy is known to be critical [22]. A small perturbation is amplified n transitional flows, becomes unstable, and causes chaotic behavior that manifests as turbulence. The amplitude of these instabilities is up to ten orders of magnitude smaller than the free stream velocity at some locations in the domain, such as in boundary layers [40]. Consequently, accurate numerical schemes with low dissipation and dispersion are required to capture this transition. This motivates the use of high-order methods for the simulation of transitional and turbulent flows.

In addition, the ultimate goal is the simulation of turbulent flows in the vicinity of complex geometries, such as the components of a turbine blade, or a jet engine. Hence, an unstructured method is required, allowing an irregular distribution of elements and volumes in the simulation domain [9]. The FEM is a classical unstructured scheme, in which the solution is globally continuous and represented using high-order polynomial degrees on an element-wise basis, while the elements share the same value at their interfaces with adjacent elements. The pitfall of the FEM is the requirement for a globally continuous solution, where a large stiffness matrix must be inverted, significantly increasing computational cost. Meanwhile, with the FVM, the solution is defined locally, and the stiffness matrix is single valued [41]. However, classical FVM is limited to the low-order accuracy. The combination of element-wise high-order accuracy of the FEM and the localized representation of solutions from FVM leads to new high-order approaches.

A summary of high-order unstructured methods has been provided by Ekaterinaris [8]. High-order methods, including DGM [4, 42, 43, 44], SV [5], SD [45], and FR approaches [6, 7], have the advantage of utilizing a local element-wise high-order representation of the solution rather than a piecewise constant solution along with the domain.

FR approach [6, 7] is particularly appealing since it combines several unstructured high-order schemes within a common framework. The FR scheme is an unstructured method that has the ability to recover other high-order methods by choosing an appropriate flux function.

Table 1.1 shows the advantages and disadvantages of the FR approach along with other spatial schemes. This shows that FR has higher benefits once compared to other methods. It is more accurate and more efficient in terms of utilizing computational resources. However, the FR approach needs to be paired with an efficient, robust and stable temporal scheme. Although current explicit schemes are simple to implement for the FR approach, they are inefficient and have prohibitive CFL restrictions. Furthermore, the cost of implicit schemes and their memory consumption increases significantly once paired with the high-order FR approach.

The formulation of FR the approach will be discussed in detail in Chapter 3.

Table 1.1. Comparison of spatial schemes, established by Hesthaven and Warburton [1].

Schemes	Unstructured Method	High-Order Accurate	Explicit Form	Quadrature Free	Unifying
Finite Difference Method	✗	✓	✓	✓	✗
Finite Volume Method	✓	✗	✓	✓	✗
Finite Element Method	✓	✓	✗	✓	✗
Discontinuous Galerkin	✓	✓	✗	✗	✗
Flux Reconstruction	✓	✓	✓	✓	✓

1.5 Thesis Objectives and Contributions

This work aims to develop robust, stable, efficient, and high-order accurate temporal schemes for high-order unstructured methods as well as other popular spatial discretization for computational aerodynamics applications.

To achieve this, we will develop optimized explicit Runge-Kutta methods for multi-dimensional high-order elements type using the flux reconstruction approach. We will then verify the convergence of these optimized methods and compare their performance with classical explicit RK schemes.

We will then develop a family of third-order accurate paired explicit RK schemes (P-ERK) suitable for locally stiff equations. We explore the order of accuracy of these new families of schemes using an isentropic vortex case. Then, we will investigate their utility with Euler and Navier-Stokes equations compared to independent numerical and experimental reference datasets.

Lastly, we generate optimal explicit Runge-Kutta methods for Ansys Fluent [46] density-based finite volume solver. We then verify the order of accuracy of these new schemes and validate them by running benchmark test cases, including the Taylor-Green vortex and flow over T106 turbine blade, using Fluent.

1.6 Thesis Outline

This Thesis is organized into a number of chapters. Chapter 2 presents the fundamental physical laws governing fluid flow, including conservation of mass, momentum, and energy. Chapter 3 outlines state of the art spatial and temporal discretizations in detail. In Chapter 4, we introduce the optimal explicit Runge-Kutta methods up to fourth-order of accuracy in time and we present the verification and validation test cases. In Chapter 5, we expand Paired Explicit Runge-Kutta schemes (PERK) to third-order accuracy. Chapter 6 contains stability analysis of Ansys Fluent density-based temporal schemes and presents optimal explicit schemes for finite volume solvers. Chapter 7 summarizes the results, and outlines recommendations for future work.

Chapter 2

Governing Equations

The fundamental laws governing fluid behavior are

- Conservation of mass; "mass cannot be created nor destroyed in a closed physical system."
- Conservation of momentum; "The momentum of an isolated physical system is conserved."
- Conservation of energy; "The total energy of an isolated physical system is conserved."

Conservation laws are governing equations of fluid flow and are used to solve fluid dynamics problems. They may be expressed in integral and differential forms. The integral forms are used to describe the rate of change in a control volume, whereas differential formulation applies the divergence theorem to apply these laws at points.

2.1 Conservation of Mass

The law of conservation of mass states that mass cannot be created nor destroyed. This requires the total derivative of the mass in a closed system to be zero as

$$\frac{dm}{dt} = 0, \tag{2.1}$$

where m is the mass in a closed system. Considering a moving control surface $\mathbb{S}$ that encloses a mass of fluid within a control volume Ω, using Reynold transport theorem

$$\frac{dm}{dt} = \int_{\Omega} \left[\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{u}) \right] d\Omega = 0, \tag{2.2}$$

where ρ is density and $\mathbf{u}$ is the velocity vector. Using divergence theorem, this can be written as

$$\frac{dm}{dt} = \frac{\partial}{\partial t} \int_{\Omega} \rho d\Omega + \oint_{\mathbb{S}} \rho(\mathbf{u}.d\mathbb{S}) = 0. \tag{2.3}$$

Moreover, conservation of mass in differential form can be derived from Equation 2.2 as

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{u}) = 0. \tag{2.4}$$

2.2 Conservation of Momentum

Newton's second law of motion states that the total external force applied to a closed system must be equal to the rate of change of momentum of that system. This can be mathematically expressed as

$$\Sigma \mathbf{f_{ext}} = \frac{d\mathbf{P}}{dt} = \frac{d}{dt}(m\mathbf{u}), \tag{2.5}$$

where $\mathbf{f_{ext}}$ is an external body force acting on the assumed control volume, and $\mathbf{P}$ is the linear momentum vector. Applying the Reynolds transport theorem leads to

$$\int_{\Omega} \left[\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla.(\rho \mathbf{u} \otimes \mathbf{u}) - \nabla.\boldsymbol{\sigma} \right] d\Omega = \int_{\Omega} \rho \mathbf{f_{ext}} d\Omega, \tag{2.6}$$

and using the divergence theorem

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \mathbf{u} d\Omega + \oint_{d\mathbb{S}} \left[(\rho \mathbf{u} \otimes \mathbf{u}) - \boldsymbol{\sigma} \right] .d\mathbb{S} = \int_{\Omega} \rho \mathbf{f_{ext}} d\Omega, \tag{2.7}$$

where $\boldsymbol{\sigma}$ is tensor of surface forces as

$$\boldsymbol{\sigma} = \begin{bmatrix} \tau_{x,x} & \tau_{x,y} & \tau_{x,z} \\ \tau_{y,x} & \tau_{y,y} & \tau_{y,z} \\ \tau_{z,x} & \tau_{z,y} & \tau_{z,z} \end{bmatrix} = -p_0\mathbf{I} + \boldsymbol{\tau}', \tag{2.8}$$

where $\boldsymbol{\tau}'$ is tensor of viscous forces as

$$\boldsymbol{\tau}' = -\frac{2}{3}(\nabla.\mathbf{uI}) + 2\mu\mathbf{D}, \tag{2.9}$$

where entries of the D tensor are $d_{i,j} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$. Similarly, conservation of momentum in the differential form can be expressed as

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla.\Big(\rho\mathbf{u}\otimes\mathbf{u} + p_0\mathbf{I} - \boldsymbol{\tau}'\Big) = \rho\mathbf{f_{ext}}. \tag{2.10}$$

2.3 Conservation of Energy

The first law of thermodynamics states that the amount of energy in an isolated system is constant. Mathematically, this can be expressed as

$$\frac{dE}{dt} = \delta Q - \delta W, \tag{2.11}$$

where $E = \rho\left(e + \frac{1}{2}\mathbf{u}.\mathbf{u}\right)$ is total energy of the system knowing $e = C_v.T$, where C_v is specific heat at constant volume, $W = \mathbf{f_{ext}}.\mathbf{u}$ is work done by or on the system as a result of the body force $\mathbf{f_{ext}}$, and $Q = \oint_{\mathbb{S}} q.d\mathbb{S}$ is the heat flux across the surface of control volume where q can be defined using the law of diffusion as

$$q = -\rho\kappa\nabla T, \tag{2.12}$$

where κ is heat conductivity constant and T is the temperature. By applying the Reynolds transport theorem

$$\int_{\Omega}\left[\frac{\partial E}{\partial t}+\nabla.(E\mathbf{u}+\rho\kappa\nabla T-\boldsymbol{\sigma}.\mathbf{u}-q_H)\right]d\Omega=\int_{\Omega}\rho\mathbf{f_{ext}}.\mathbf{u}d\Omega, \tag{2.13}$$

where $\boldsymbol{\sigma}.\mathbf{u}$ is the heat produced by the internal shear stress [47], and q_H is other heat sources. Applying the divergence theorem leads to

$$\frac{\partial}{\partial t}\int_{\Omega}Ed\Omega+\oint_{\mathbb{S}}(E\mathbf{u}+\rho\kappa\nabla T-\boldsymbol{\sigma}.\mathbf{u}-q_H).d\mathbb{S}=\int_{\Omega}\rho\mathbf{f_{ext}}\mathbf{u}d\Omega. \tag{2.14}$$

Alternatively, conservation of energy can be written in divergence form as

$$\frac{\partial E}{\partial t}+\nabla.(E\mathbf{u}+\rho\kappa\nabla T-\boldsymbol{\sigma}.\mathbf{u}-q_H)=\rho\mathbf{f_{ext}}\mathbf{u}. \tag{2.15}$$

2.4 Equation of State

In addition to above mentioned equations, we need another relationship between the flow variables to close the system of equations. That relationship is typically the ideal gas law which is shown to be a good approximation of the behavior of gases under a wide range of conditions [48]. The ideal gas law can be described as

$$P_0=\rho\overline{R}T, \tag{2.16}$$

where P_0 is pressure and $\overline{R}$ is the gas constant.

2.5 General Conservative Form

Conservation laws can be combined as parts of Euler [49] and Navier-Stokes [12] equations. The Navier-Stokes equations can be written as

$$\frac{\partial\mathbf{Q}}{\partial t}+\nabla.\left(\mathbf{F_i^c}(\mathbf{Q})-\mathbf{F_i^v}(\mathbf{Q},\nabla.\mathbf{Q})\right)=0, \tag{2.17}$$

where **Q** is the state vector and is defined as

$$Q = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ E \end{bmatrix}. \tag{2.18}$$

Moreover, the inviscid flux $\mathbf{F_i^c}(\mathbf{Q})$ is expressed as

$$\mathbf{F_i^c}(\mathbf{Q}) = \begin{bmatrix} \rho u_i \\ \rho u_i u_x + \delta_{i,1} P_0 \\ \rho u_i u_y + \delta_{i,2} P_0 \\ \rho u_i u_z + \delta_{i,3} P_0 \\ u_i E \end{bmatrix}, \tag{2.19}$$

where i represents a spatial coordinate $i = [x, y, z]$. and $\delta_{i,n}$ is column n of Kronecker delta. Similarly, viscous flux $\mathbf{F_i^v}$ can be defined as

$$\mathbf{F_i^v}(\mathbf{Q}, \nabla.\mathbf{Q}) = \begin{bmatrix} 0 \\ \tau_{x,i} \\ \tau_{y,i} \\ \tau_{z,i} \\ \tau_{i,j}.\mathbf{u} + \kappa \partial_i T \end{bmatrix}. \tag{2.20}$$

where, τ is viscous shear stress, and ∂_i is the partial derivative with respect to i, where $i = [x, y, z]$

We can also define a source vector which consists of external forces f and heat sources q_H as

$$\mathbf{S} = \begin{bmatrix} 0 \\ \rho f_x \\ \rho f_y \\ \rho f_z \\ \rho(\mathbf{f}.\mathbf{u}) + q_H \end{bmatrix}. \tag{2.21}$$

Furthermore, in the case of the Euler equations for inviscid flow, we can simply neglect the viscous fluxes. In this study we use ILES; hence, no additional SGS equation is solved.

Chapter 3

Numerical Discretizations

3.1 Flux Reconstruction as Spatial Discretization

In this study, we are primarily interested in the FR approach as a high-order spatial discretization since it has the unique property of recovering other high-order methods. In this section, we review the basic formulation of FR spatial discretizations. In Chapter 6, we develop optimized temporal schemes for Ansys Fluent [46] finite volume density-based solver. Hence, we review the FVM in Chapter 6.

Flux reconstruction was first developed for advection equations [6], extended to diffusion equations [7], and later expanded by Wang et al. [50], William et al. [51], and Vincent et al. [10, 37] for simplex elements. Here, we describe the original formulation for the FR approach.

3.1.1 One-Dimensional Formulation

We start with a one-dimensional conservation law in a domain Ω

$$\frac{\partial u}{\partial t} + \frac{\partial F}{\partial x} = 0, \tag{3.1}$$

where $u = u(x,t)$ is the solution, and $F = F(u)$ is the flux function. We split the domain Ω into N_e elements such that

$$\Omega = \bigcup_{n=1}^{N_e} \Omega_n, \qquad \bigcap_{n=1}^{N_e} \Omega_n = \emptyset. \tag{3.2}$$

For simplicity, we perform all operations in a reference space where each element can be mapped into $\xi \in [-1, 1]$

$$\xi = 2\left(x - \frac{x_L + x_R}{2}\right), \tag{3.3}$$

where x_L and x_R is the mesh node corresponding to the left and right face of Ω_n, and ξ is the location in reference space. This inverse form of this mapping is

$$x = \left(\frac{1-\xi}{2}\right)x_L + \left(\frac{1+\xi}{2}\right)x_R, \tag{3.4}$$

which is also linear. The system of equations in reference space can then be written as

$$\frac{\partial u_n^\delta}{\partial t} + \frac{\partial \xi}{\partial x}\frac{\partial F_n^\delta}{\partial \xi} = 0. \tag{3.5}$$

where the superscript δ denotes that this flux F, like the solution u, is allowed to be discontinuous at the interface between elements. According to Huynh [6], the solution within each element is represented by a degree k polynomial, which is allowed to be discontinuous at the interface between elements. This polynomial is supported by nodal basis functions generated at $k+1$ solution points. Therefore, the solution within each element in reference space can be approximated as

$$u_n^\delta = \sum_{l=0}^{k} u_{n,l}\phi_l, \tag{3.6}$$

where $u_n^\delta = u_n^\delta(\xi, t)$ is the polynomial representation of the solution within an element, $u_{n,l}^\delta = u_{n,l}^\delta(t)$ is the value of the solution at solution point l, and $\phi_l = \phi_l(\xi)$ is its corresponding nodal basis function in reference space. For the one-dimensional case, these basis functions are the Lagrange polynomials

$$\phi_l(\xi) = \prod_{m=0, m\neq l}^{k} \frac{\xi - \xi_m}{\xi_l - \xi_m}. \tag{3.7}$$

A polynomial representation of the discontinuous flux function $F_n^{\delta D} = F_n^{\delta D}(x,t)$ can be constructed using the same polynomial basis as the solution according

$$F_n^{\delta} = \sum_{l=0}^{k} F_{n,l}\phi_l, \tag{3.8}$$

and $F_{n,l}^{\delta} = F_{n,l}^{\delta}(t)$ is the flux evaluated at the solution points, as shown in Figure 3.1.

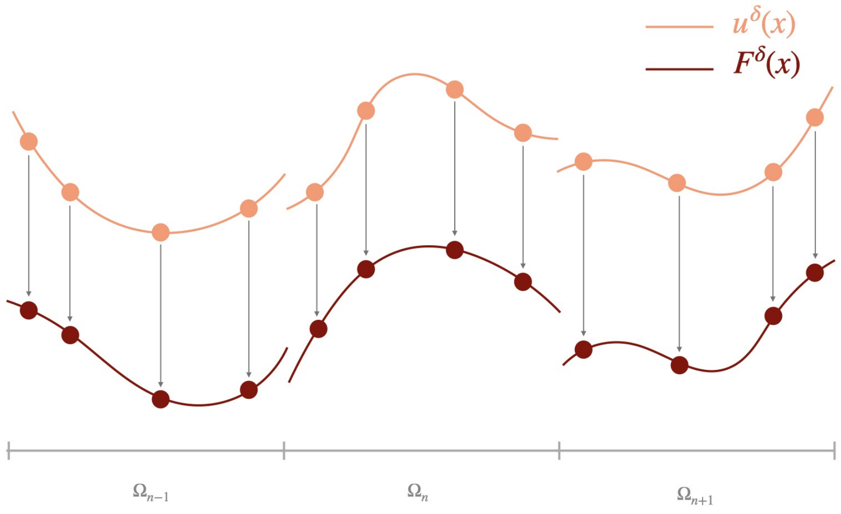


Figure 3.1. Schematic representation of solution and flux polynomials along with solution points.

This discontinuous flux is then corrected to be globally continuous, as shown in Figure 3.2, using a pair of correction functions

$$F_n^{\delta C} = \left(F_n^{CL} - F_{n,L}^{\delta}\right) g_L + \left(F_n^{CR} - F_{n,R}^{\delta}\right) g_R, \tag{3.9}$$

where $F_{n,L}^{\delta} = F_n^{\delta}(-1,t)$, and $F_{n,R}^{\delta} = F_n^{\delta}(1,t)$ are fluxes evaluated at the right and left interfaces. The terms $F^{CL} = F^{CL}(u_L^-, u_L^+)$ and $F^{CR} = F^{CR}(u_R^-, u_R^+)$ are common interface fluxes computed at the flux points between elements by an appropriate Riemann solver using the interpolated values u_L^-, u_L^+, u_R^-, and u_R^+ of the solution from the neighbouring elements at each edge. The functions $g_L = g_L(\xi)$ and $g_R = g_R(\xi)$ are correction functions, which are degree $k+1$ polynomials with the constraints

$$g_L(-1) = 1, \quad g_L(1) = 0, \tag{3.10}$$

$$g_R(-1) = 0, \quad g_R(1) = 1, \tag{3.11}$$

where g_L and g_R can be chosen such that we recover different high-order methods.

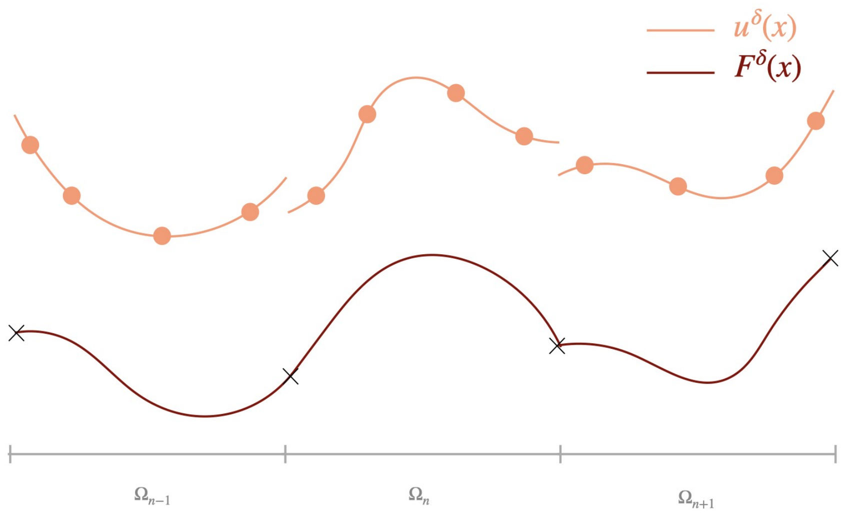


Figure 3.2. Schematic representation of continuous flux corrected at the interface using a correction function, where x represents the spatial coordinates.

3.1.2 Multi-Dimensional Formulation

We start by considering the conservation law of the form

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \boldsymbol{F} = 0, \tag{3.12}$$

where $\boldsymbol{F} = \boldsymbol{\alpha}\mathbf{u}$ is the flux vector, and $\boldsymbol{\alpha}$ governs the wave speed and direction. In the current study, we restrict the value of $\boldsymbol{\alpha}$ such that $|\boldsymbol{\alpha}| = 1$, meaning the wave speed is unity. The solution is represented by a discrete approximation on each element such that [6, 52]

$$\mathbf{u}(\boldsymbol{x}, t) \approx \mathbf{u}^h(\boldsymbol{x}, t) = \bigoplus_{i=1}^{N_e} \mathbf{u}_i^h(\boldsymbol{x}, t), \tag{3.13}$$

where $\mathbf{u}^h(\boldsymbol{x}, t)$ is the global piecewise continuous approximation of the solution and $\mathbf{u}_i^h(\boldsymbol{x}, t)$ is a continuous representation of the solution on one of N_e elements in the domain. Similar to one-dimensional case, we take the approximate solution on each element to be a polynomial nodal basis representation of degree p such that

$$\mathbf{u}_i^h(\boldsymbol{x}, t) = \sum_{j=1}^{N_s} \mathbf{u}_{i,j}(t)\phi_{s,i,j}(\boldsymbol{x}), \tag{3.14}$$

where $\mathbf{u}_{i,j}(t)$ is the solution's value at one of N_s solution nodal basis points on a given element and $\phi_{s,i,j}(\boldsymbol{x})$ is its corresponding multidimensional solution nodal basis function. This approach ensures the solution is continuous on each element but allows the solution to be discontinuous on the interfaces between elements [6]. Following the flux reconstruction approach [6] and its extension to simplex element types [52, 53], the physical conservation law that must be satisfied in the discrete sense on each element is

$$\frac{\partial \mathbf{u}_i^h}{\partial t} + \nabla \cdot \boldsymbol{F}_i^h + \mathcal{D}_i = 0, \tag{3.15}$$

where $\boldsymbol{F}_i^h = \boldsymbol{\alpha} \boldsymbol{u}_i^h$ and $\mathcal{D}_i$ is a correction field on the element in the same polynomial space as the solution. This correction field is analogous to the divergence of the correction functions introduced in the original FR scheme for tensor product elements [6]. Finally, applying the conservation law at each of the solution points, we obtain

$$\frac{du_{i,j}^h}{dt} + \left(\nabla \cdot \boldsymbol{F}_i^h\right)\Big|_j + \mathcal{D}_{i,j} = 0, \tag{3.16}$$

and following the FR formulation

$$\mathcal{D}_{i,j} = \frac{1}{|\Omega_i|} \sum_{f \in S_\Omega} \sum_{l} \kappa_{i,j,f,l} [\hat{F}]_{i,f,l} \mathbb{S}_f, \tag{3.17}$$

where $|\Omega_i|$ is the element volume, f is one of the number of faces on the element surface $\mathbb{S}_\Omega$, l is one of the flux points, $\kappa_{i,j,f,l}$ is a constant lifting coefficient, $[\hat{F}]_{i,f,l}$ is the difference between a common Riemann flux at the flux point and the value of the internal flux, and $\mathbb{S}_f$ is the area of the face. In the current study, we use an upwind flux at all interfaces. Depending on the specification of these lifting coefficients, a number of different energy stable schemes can be obtained for general element types. This study, uses lifting coefficients based on the nodal basis functions to recover the DG method [6, 52].

As mentioned before, for efficiency and simplicity, we map the solution and discrete system of governing equations into a reference element with coordinates $\boldsymbol{\xi}$ using a one-to-one mapping $\boldsymbol{x} = M(\boldsymbol{\xi})$ such that $\boldsymbol{\xi} = M^{-1}(\boldsymbol{x})$. The mapping points are used to define a nodal polynomial representation of the mapping function M such that

$$\boldsymbol{x}_i^h(\boldsymbol{\xi}) = \sum_{j=1}^{N_g} \boldsymbol{x}_{i,j} \phi_{g,i,j}(\boldsymbol{\xi}). \tag{3.18}$$

where $\boldsymbol{x}_i^h(\boldsymbol{\xi})$ is the interpolated physical location, N_g is the number of mapping points, $\boldsymbol{x}_{i,j}$ is the physical location of the mapping points, and $\phi_{g,i,j}(\boldsymbol{\xi})$ are the nodal basis functions associated with the mapping points. The Jacobian of this transformation can be found at any point from

$$J = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}. \tag{3.19}$$

This allows all operations to be performed on the reference element and mapped back to the physical element as required [6]. In the current study, all elements are taken to be straight-sided.

3.2 Temporal schemes

3.2.1 Introduction

Current temporal schemes, which are widely used in science and engineering, are categorized into three different groups [39]:

- Explicit schemes
- Implicit schemes
- Implicit-Explicit (IMEX) schemes

Explicit schemes are relatively inexpensive per time-step and easy to implement. However, they limit the time-step size, which makes the solution of systems of differential equations using an explicit approach often impractical since numerical stiffness necessitates that prohibitively small time-steps must be taken to maintain stability [39].

Implicit schemes are more expensive per time-step, but we can take relatively large time-steps. In fact, implicit schemes are often used to overcome explit schemes' time-step size and stability limits. However, implicit schemes require high memory storage and are computationally expensive per time-step for the solution of a large system of linear or non-linear differential equations [39].

Both of these schemes have their advantages and disadvantages. Meanwhile, in practical applications, we often come across double-scale equations, which, in some regions of the domain, show stiff behavior, and in other regions, they are non-stiff. Implicit-Explicit (IMEX) schemes are appealing in this case. These schemes work by pairing an implicit scheme in the stiff part with an explicit scheme in the non-stiff part. In other words, an IMEX scheme splits a system of double-scale equations into its stiff and non-stiff constituent parts [54]. The stiff part is solved implicitly, usually using an unconditionally stable scheme, while the non-stiff part is solved explicitly. This approach has the ability to greatly reduce the size of the implicit system of equations, avoid the prohibitive stability restrictions of the explicit scheme, and significantly reduce the computational cost of solving stiff systems of ODEs and PDEs.

In this section, we try to identify recent progress in each of these three approaches and recognize the computational cost of solving a system of differential equations. We also study the stability and optimization of these new schemes for high-order spatial discretizations such as DGM and FR.

3.2.2 Explicit Schemes

Many transient physical problems either can be described in the form of time-dependent ODEs or in the form of time-dependent conservation laws PDEs. These problems are often solved using a method-of-lines approach, where spatial discretization of the PDEs yields a set of time-dependent ODEs that must then be integrated in time [55]. It is well known that, when using explicit time discretization methods, the time-steps that are employed must satisfy Courant-Friedrichs-Lewy (CFL) stability constraints [39]. These constraints include conditions to maintain both linear stability, to ensure convergence of the solution, and some form of nonlinear stability (e.g, total variation “TVD” stability), to overcome oscillations of the numerical approximations in case of discontinuities [39].

First-Order Forward Euler Scheme

The first-order forward Euler scheme is the most straightforward scheme that can be used to solve differential equations. This scheme is usually used to solve linear initial value problems. The Euler method is a first-order method, which means that the local error (error per step) is proportional to the square of the step size, and the global error (error at a given time) is proportional to the step size. The general forward Euler method can be defined as

$$u_{n+1} = u_n + \Delta t R_n, \tag{3.20}$$

where u_{n+1} and u_n are the solutions of the differential equations at time-step t_{n+1}, and t_n and R_n is the value of residuals at t_n. However, it can be shown that a method-of-lines approach using FR spatial discretizations in conjunction with the first-order forward Euler method is linearly unstable for $k = 0$ under any constant time-step to mesh size ratio [44].

Explicit Runge-Kutta Schemes

Historically, Runge-Kutta schemes are a family of iterative methods, which use the Euler method with a set of stages [38]. It is usually convenient to represent a Runge-Kutta method by a partitioned tableau named after John Butcher, of the form [38]

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b^T} \end{array}, \tag{3.21}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1s} \\ a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \dots & a_{ss} \end{bmatrix}, \tag{3.22}$$

$$c_i = \sum_{j=1}^{s} a_{ij}, \tag{3.23}$$

where s is the number of stages in a general Runge-Kutta scheme (number of iterations). To have consistency (first-order accuracy), the following condition must be satisfied

$$\sum_{i=1}^{s} b_i = 1. \tag{3.24}$$

The vector **c** indicates the positions, within the step, of the stage values, the matrix **A** indicates the dependence of the stages on the derivatives found at other stages, and $\mathbf{b}^T$ is a vector of quadrature weights, showing how the final result depends on the derivatives computed at the various stages. In explicit schemes, since these derivatives are all associated with previous stages derivatives, the A matrix is a strictly lower triangular matrix. In other words, for explicit schemes, we require [38]

$$a_{ij} = 0, \forall \ \ j \geq i, \tag{3.25}$$

hence, an explicit Runge-Kutta scheme can be defined as

$$u_i = u_n + \Delta t \sum_{j=1}^{s} a_{ij} R(t_n + c_j \Delta t, u_j), \tag{3.26}$$

and then

$$u_{n+1} = u_n + \Delta t \sum_{i=1}^{s} b_i R(t_n + c_i \Delta t, u_i). \tag{3.27}$$

Popular $\mathrm{RK}_{3,3}$ and $\mathrm{RK}_{4,4}$ methods can be expressed with Butcher tableau as

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & 0 \\
1 & -1 & 2 & 0 \\
\hline
 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6}
\end{array}$$

Table 3.1. $\mathrm{RK}_{3,3}$ scheme

$$\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}$$

Table 3.2. $\mathrm{RK}_{4,4}$ scheme

Order Conditions of Runge-Kutta Schemes

As the order of accuracy of a Runge-Kutta scheme increases, the algebraic conditions on the coefficients of the method become increasingly complicated. The pattern behind these conditions is known, but in this brief introduction, we just state these conditions without any justification which can be obtained from Butcher's book [38]. In general, it can be said that to have a scheme of order n we must satisfy 2^{n-1} constraints. These constraints (order conditions) can be written down as polynomials of the coefficient matrices.

As mentioned, for a first-order schemes, we need to only satisfy the consistency condition in Equation 3.24 [38]. For second-order schemes, we need to satisfy two conditions. In addition to the consistency condition (Equation 3.24) we need to satisfy [38]

$$\sum_{i=1}^{s} b_i c_i = \frac{1}{2}, \tag{3.28}$$

and for third-order schemes, we need two more constraints [38]

$$\sum_{i=1}^{s} b_i c_i^2 = \frac{1}{3}, \tag{3.29}$$

$$\sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} c_j = \frac{1}{6}, \tag{3.30}$$

and for the fourth-order schemes, we must satisfy four more constraints as [38]

$$\sum_{i=1}^{s} b_i c_i^3 = \frac{1}{4}, \tag{3.31}$$

$$\sum_{i=1}^{s} \sum_{j=1}^{s} b_i c_i a_{ij} c_j = \frac{1}{8}, \tag{3.32}$$

$$\sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} c_j^2 = \frac{1}{12}, \tag{3.33}$$

$$\sum_{i=1}^{s} \sum_{j=1}^{s} \sum_{k=1}^{s} b_i a_{ij} a_{jk} c_k = \frac{1}{24}. \tag{3.34}$$

It is also worth mentioning that a scheme with stages s can have an order of p if and only if [38]

$$s \geq p. \tag{3.35}$$

Explicit Runge-Kutta schemes are widely used in academia, coupled with DG/FR spatial discretization to solve physical phenomena. For example, PyFR, an open source software for solving advection-diffusion type problems [56], or HiFiLES, a high-order LES unstructured solver [57], or Hindenlang et al. [58] for implementation of explicit schemes for DG/FR methods. Low order explicit schemes are also used in industry in combination with different spatial discretizations, including the FVM and FDM [41].

Stability of Explicit Runge-Kutta Schemes

Explicit Runge-Kutta methods are among the most widely used types of numerical integrators for solving initial value ordinary and partial differential equations. The time-step size should be taken as large as possible since the cost of solving up to a fixed final time is proportional to the number of steps that must be taken. In practical computation, the time-step is often limited by stability and accuracy constraints. Either accuracy, stability, or both may be limiting factors for a given problem. The linear stability and accuracy of an explicit Runge-Kutta method are characterized entirely by the so-called stability polynomial of the method, which in turn dictates the acceptable step size [38, 39]. An explicit Runge-Kutta method can always be expressed as [59]

$$u_{n+1} = P_{s,p}(z)u_n, \tag{3.36}$$

where $P_{s,p}(z)$ is called the schemes stability polynomial and $z = \omega\Delta t$, where ω is an eigenvalue of the system of equations being solved (eigenvalues based on discretization method in space). The resulting scheme will be linearly stable for this eigenvalue provided $|P_{s,p}| \leq 1$. The value of the stability polynomial can be determined directly from the Butcher tableau

$$P_{s,p}(z) = 1 + z\mathbf{b}^T\left(\mathbf{I} - z\mathbf{A}\right)^{-1}\mathbf{e} = \frac{|\mathbf{I} - z\mathbf{A} + z\mathbf{e}\mathbf{b}^T|}{|\mathbf{I} - z\mathbf{A}|}, \tag{3.37}$$

where $\mathbf{e}$ is a vector of ones. The region of absolute stability S of the explicit Runge-Kutta method is the set in the complex plane where this condition holds, that is [59],

$$S = \left\{z \in \mathbb{C} : |P_{s,p}| \leq 1\right\}, \tag{3.38}$$

and the linear stability condition is

$$\Delta t\omega \subseteq S\,. \tag{3.39}$$

Figure 3.3 shows the regions of absolute stability of classical Runge-Kutta schemes.

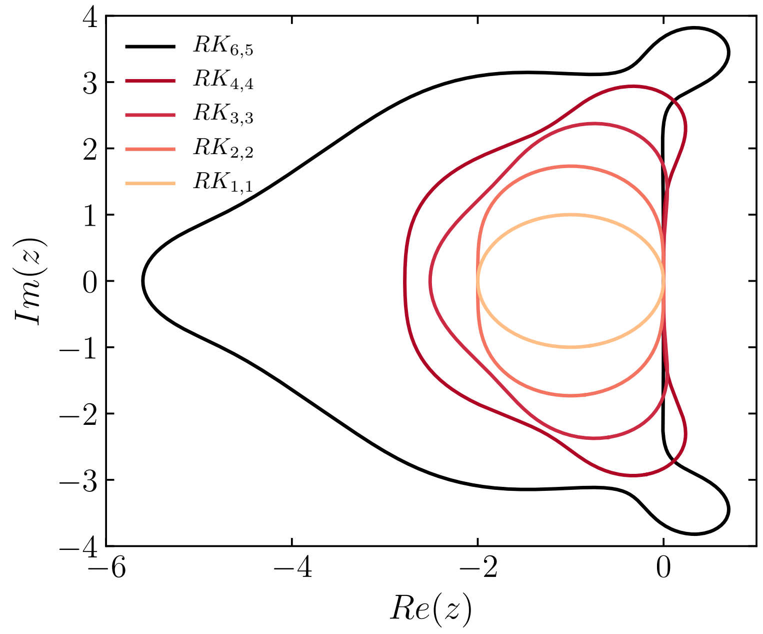


Figure 3.3. Region of absolute stability of classical Runge-Kutta methods.

For a given system of equations and corresponding values of ω, optimizing the region of absolute stability of the explicit Runge-Kutta method can enable a larger stable Δt.

Design of Optimal Stability Polynomials

Here, we would like to consider the problem of choosing a stability polynomial to maximize the step size under which stability constraints discussed earlier, are satisfied. In fact, having the stability constraint in addition to the stability polynomial provides us with a set of non-linear equations, which can be used to determine the unknowns of the Butcher tableau.

The stability conditions yield nonlinear inequality constraints. Typically one also wishes to impose a minimal order of accuracy. The monomial basis representation of $P_{s,p}(z)$ is then convenient because the first $p+1$ coefficients $(\gamma_0, \gamma_1, ..., \gamma_p)$ of the stability polynomial are simply taken to satisfy the order conditions [59]. As a result, the space of decision variables has dimension $s + 1 - p$, and consists of the coefficients $(\gamma_{p+1}, \gamma_{p+2}, ..., \gamma_s)$, as well as the step size Δt. Then the problem can be stated as [59]

Problem 1:

$$\begin{aligned} &\underset{\gamma_{p+1},\ldots,\gamma_s}{\text{maximize}} && \Delta t \\ &\text{subject to} && |P_{s,p}(\Delta t\omega^{\delta})|-1 \leq 0, \ \ \forall\omega^{\delta}, \end{aligned} \tag{3.40}$$

where $z = \omega\Delta t$, and $\omega \in S$, where $S \subset \mathbb{C}$. Therefore, we look for the optimal time-step size Δt_{opt} and corresponding optimal stability polynomial P_{opt}. From Von Neumann analysis of finite-dimensional PDEs spatial discretization, we know that, these set of eigenvalues (ω) can be a continuous function. In this case, we have to use a finite sample of ω [59].

It is evident that finding the global solution to this problem is in general quite challenging. Even though optimization techniques can give us stability polynomial solutions to convex problems, and convex problems are useful in case of seeking minima. However, in our case, we are dealing with a non-convex problem (for $s > 2$) and we are looking for maxima. Therefore, the first step in this optimization problem is reformulation in terms of the least deviation. In fact instead of asking for the maximum stable time-step size we now ask, for a given step size (Δt), and see how small the maximum modulus of $P(\Delta t)$ can be. This leads to a generalization of the classical least deviation problem as

Problem 2:

$$\underset{\gamma_{q+1},\gamma_{p+2},\ldots,\gamma_s}{\text{minimize}} \quad \underset{\omega^{\delta}\,\in\,\boldsymbol{\omega}^{\delta}}{max} \left(|P_{s,p}(\Delta t\omega^{\delta})|-1\right). \tag{3.41}$$

where $\omega \in S$, and $S \subset \mathbb{C}$. We denote the solution of Problem 2 by $r(\Delta t, \omega)$. If we notice, in Problem 2 we try to minimize the value of $a_{p+1}, a_p, \ldots, a_s$, since $P(z)$ is linear function of a_j. Therefore, it can be said that Problem 2 is a convex problem [59].

Here, we can reformulate our main optimization problem in terms of Problem 2 as [59]

Problem 3:

$$\begin{aligned} \underset{\gamma_{q+1},\gamma_{q+2},\dots,\gamma_s}{\text{maximize}} \quad & \Delta t \\ \text{subject to} \quad & r_{s,q}(\Delta t, \boldsymbol{\omega}^{\delta}) \leq 0. \end{aligned} \tag{3.42}$$

Although Problem 3 is not known to be convex, it is an optimization in a single variable. Therefore, the bisection approach can be used to find the solution.

Since, $r(0,\omega) = -1$ and $\lim_{\Delta t \to \infty} r(\Delta t, \omega) = +\infty$, then, we can say there is a $\Delta t_{max} > 0$ such that $r(\Delta t, \omega) = 0$ for some $\Delta t \in [\Delta t_{min}, \Delta t_{max}]$. Global convergence can be assured if and only if

$$r_{p,s}(\Delta t_0, \omega) = 0 \implies r_{p,s}(\Delta t, \omega) \leq 0 \quad \text{for all} \quad 0 \leq \Delta t \leq \Delta t_0 \tag{3.43}$$

Here, we have to consider the situations under which the above condition can be established. For first-order schemes ($p = 1$), and $s \geq 1$, we can prove that the maximum time-step is equal to

$$\lim_{\epsilon \to 0} \Delta t_{\epsilon} = \Delta t_{opt} \tag{3.44}$$

where ϵ is a small number and the value of Δt_{ϵ} can be found via bisection [59]

```
Select $\Delta t_{max}$
$\Delta t_{min} = 0$
**while** $\Delta t_{max} - \Delta t_{min} > \epsilon$ **do**
    $\Delta t = \frac{\Delta t_{min} + \Delta t_{max}}{2}$
    Solve Equation 3.42
    **if** $r_{s,q}(\Delta t, \boldsymbol{\omega}^{\delta}) \leq 0$ **then**
        $\Delta t_{min} = \Delta t$
    **else**
        $\Delta t_{max} = \Delta t$
    **end**
**end**
**return** $\Delta t_{opt} = \Delta t_{min}$
```

Algorithm 1: Finding Δt_{opt}.

Kubatko et al. [55] stated that the convexity of this optimization problem for high-order explicit Runge-Kutta schemes could be neither proven nor refused. However, it was still feasible in practice to use the same method to optimize the stability polynomial of high-order Runge-Kutta schemes. The bisection algorithm requires an initial Δt such that it satisfies $r(\Delta t, \omega) > 0$. Since the evaluation of $r(\Delta t, \omega) > 0$, is generally fast, one can start with a guess and successively double it until $r(\Delta t, \omega) > 0$ is satisfied.

3.2.3 Implicit Schemes

Explicit schemes, including the family of explicit Runge-Kutta schemes are conditionally stable. Because of this limitation, the maximum possible time-step to maintain stability is often small, especially in stiff equations. This increases the number of time-steps and total computational cost. Implicit schemes are advantageous from the perspective of stability [60]. However, they are more costly per time-step and usually more complicated to implement.

First-Order Backward Euler Scheme

The backward Euler method is an implicit scheme which means, contrary to explicit methods, finding the solution of an equation involves information from current and previous stages, and information from as yet unknown future stages. More precisely, if we assume a general differential equation with the first-order derivative with respect to time as

$$\frac{\partial u}{\partial t} + R = 0, \tag{3.45}$$

where R is the value residual which is a linear or non-linear function of u as $R = R(u(x^i, t))$). In order to get an approximate solution in time, we use the first-order expansion for the temporal derivative as

$$\frac{u_{n+1} - u_n}{\Delta t} + R = 0, \tag{3.46}$$

where n is a variable taken at the current time-step and $n+1$ at the next time-step. To have an implicit scheme (Backward Euler), we evaluate the value of residual R at next time-step rather than the current time-step as

$$\frac{u_{n+1} - u_n}{\Delta t} + R_{n+1} = 0, \tag{3.47}$$

which is a first-order accurate implicit Euler method. Noticing that the residual is a function of u_{n+1}, which is what we are trying to solve the equation for, making the numerical algorithm more complicated. To solve this, we define a new value R^* as

$$R^*_{n+1} = \frac{u_{n+1} - u_n}{\Delta t} + R_{n+1}, \tag{3.48}$$

which is called the unsteady residual. Then, we must satisfy the condition

$$R^*_{n+1} = 0. \tag{3.49}$$

To solve, we use the Newton-Raphson linearization to approximate the value of the unsteady residual at the next iteration as

$$R^*_{k+1} \approx R^*_k + \frac{\partial R^*_k}{\partial u_k} \Delta u, \tag{3.50}$$

where

$$\Delta u = u_{k+1} - u_k \tag{3.51}$$

which is the update for the solution vector at the end of each iteration k.

The second term on the right hand side of Equation 3.50 is the Jacobian matrix of the residual vector, and it can be determined simply by taking the derivative from Equation 3.48 as

$$\frac{\partial R^*_k}{\partial u_k} = \left[\frac{I}{\Delta t} + \frac{\partial R_k}{\partial u_k} \right]. \tag{3.52}$$

By combining Equation 3.48 and 3.52 with equation 3.50, we can get

$$\left[\frac{I}{\Delta t} + \frac{\partial R_k}{\partial u_k}\right] \Delta u = -R_k - \frac{u_k - u_n}{\Delta t} \tag{3.53}$$

which is a linear system of equations that must be solved at each iteration. These iterations must be continued until the value of the unsteady residual R_k^* become less than an implicit tolerance. The value of this tolerance depends on the desired accuracy. For linear system of equations, the unsteady residual must converge to a specified implicit tolerance in just one iteration, whereas for non-linear systems, it can require several outer iterations.

Implicit Runge-Kutta Schemes

Implicit Runge-Kutta schemes (IRK) use a multi-stage series of first-order accurate intermediate stages to form an overall high-order scheme [39, 60, 61]. These schemes are beneficial in the case of stiff systems of equations. However, they are often more expensive than the backward Euler scheme due to the additional expense of multiple implicit stages per time-step.

To study implicit Runge-Kutta schemes, we reconsider the general Butcher tableau with s stages as

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b^T} \end{array}, \tag{3.54}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,s} \\ a_{2,1} & a_{2,2} & \dots & a_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s,1} & a_{s,2} & \dots & a_{s,s} \end{bmatrix}. \tag{3.55}$$

We previously defined the explicit Butcher tableau as a strictly lower triangular A matrix. However, if non-zero values exist in the upper triangular or diagonal sections of the A matrix, the Butcher tableau yields an implicit scheme since the value of derivatives at each stage are computed with residuals from unknown later stages. There is also a special case of A, where if just the diagonal values are non-zero, then the scheme is called Diagonally Implicit Runge-Kutta or DIRK scheme. In DIRK schemes, the value of derivatives at each stage is a function of the unknown solution at the next stage.

Since the memory requirements for a general implicit Runge-Kutta scheme are very high, it is not often efficient to use them for the large scale system of equations. Hence, the main focus is on DIRK schemes.

For a DIRK scheme, the general form of the scheme at each stage by separating off-diagonal explicit terms from the implicit diagonal terms can be determined as

$$u_i = u_n + \Delta t \sum_{j=1}^{i-1} a_{i,j} R_j + \Delta t a_{i,i} R_i, \tag{3.56}$$

where, since this is a DIRK scheme, we know that

$$a_{i,j} = 0 \ \forall \ j > i.$$

Therefore, we can see that an implicit residual exists only in the third term on the right hand side of Equation 3.56. Hence, the other terms can be defined as

$$u_p = u_n + \Delta t \sum_{j=1}^{i-1} a_{i,j} R_j, \tag{3.57}$$

where u_p is the value of the unknown quantity with only explicit residuals. Then Equation 3.56 can be rewritten as

$$u_i = u_p + \Delta t a_{i,i} R_i, \tag{3.58}$$

if we rearrange the above equation

$$\frac{u_i - u_p}{\Delta t a_{i,i}} - R_i = 0, \tag{3.59}$$

and by using the same method we used in Section 3.2.3 we can find

$$\frac{\partial R_k^*}{\partial u_k} = \left[\frac{I}{\Delta t a_{ii}} + \frac{\partial R_k}{\partial u_k}\right] \tag{3.60}$$

then for each iteration in the Newton-Raphson method we have

$$\left[\frac{I}{\Delta t a_{ii}} - \frac{\partial R_k}{\partial u_k}\right]\Delta u = -R_k - \frac{u_k - u_n}{\Delta t}, \tag{3.61}$$

where $\Delta u = u_{k+1} - u_k$ again. As was the case for the backward Euler method, here, we iterate to approximate $R_k^* \approx 0$ to a specified implicit tolerance.

Calculation of the matrix $\frac{\partial R_k}{\partial u_k}$, which is called the Jacobian matrix, is computationally costly. Sometimes, It is even more costly than the actual solution of the linear system. Therefore, it is necessary to pay attention to the time-step size and implicit tolerance when dealing with non-linear equations. One performance improvement method could be the usage of one identical diagonal value ($a_{i,i} = \gamma$). This is called a Singly Diagonal Implicit Runge-Kutta (SDIRK) scheme, which needs to precondition the system only once for linear systems. The structure of this Jacobian matrix is associated with the spatial discretization of the problem. For general DG/FR schemes, the Jacobian matrix is a square matrix, where the number of rows and columns is equal to the number of elements multiplied by the degree of the polynomial plus one to the power of the number of spatial dimensions (($N_e \times (k+1)^3$) in case of a 3D spatial dimensions, where N_e is the number of elements. Hence, it can be said that the Jacobian matrix has non-zero entities in diagonal blocks, where the size of each block is $(k+1) \times (k+1)$ for one-dimensional problems. It also can be shown that if we use upwind flux, we will have non-zero $(k+1) \times (k+1)$ off-diagonal blocks on the upwind side (left or right side) of the main diagonal of the Jacobian Matrix. Note that, in the case of 2D and 3D PDEs, the size of the diagonal blocks in the Jacobian matrix is $(k+1)^2 \times (k+1)^2$ and $(k+1)^3 \times (k+1)^3$, respectively. This shows that the cost of implicit schemes increases exponentially as the mesh size or order of accuracy in space increases.

A general form of the Jacobian matrix with an upwind flux and periodic boundary conditions can be seen below, where diagonal and off-diagonal blocks are shown

$$\frac{\partial R_k}{\partial u_k} = \begin{bmatrix} \begin{bmatrix} \quad \end{bmatrix} & 0 & \dots & \dots & \begin{bmatrix} \quad \end{bmatrix} \\ \begin{bmatrix} \quad \end{bmatrix} & \begin{bmatrix} \quad \end{bmatrix} & 0 & \dots & 0 \\ 0 & \begin{bmatrix} \quad \end{bmatrix} & \begin{bmatrix} \quad \end{bmatrix} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \begin{bmatrix} \quad \end{bmatrix} & \begin{bmatrix} \quad \end{bmatrix} \end{bmatrix}. \tag{3.62}$$

By taking advantage of Butcher tableau we can express implicit schemes compactly. Some typical implicit schemes are shown below

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Table 3.3. Backward Euler scheme

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

Table 3.4. Crank-Nicholson scheme

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1 & 1-\gamma & \gamma \\ \hline & 1-\gamma & \gamma \end{array}$$

Table 3.5. SDIRK$_{2,2}$ scheme

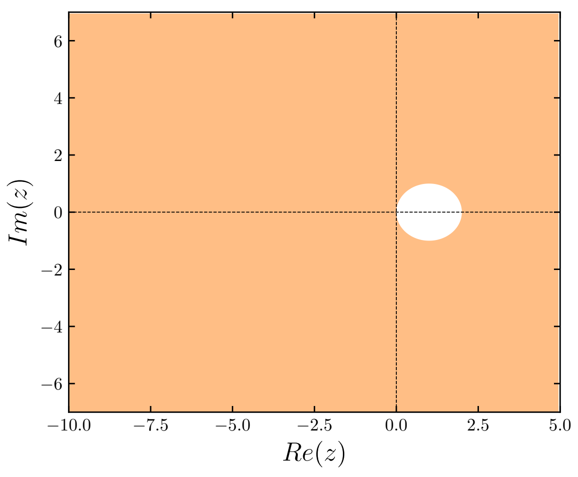
Using the order conditions for SDIRK$_{2,2}$, we can find $\gamma = 1 - \sqrt{2}/2$. The implementation of implicit schemes (DIRK schemes) and their applications with high-order methods can be found in the work of Persson et al. [62] and Roethe et al [63] and Hillewaert et al [64]. In addition, implicit schemes are widely used with classical FVM and FEM methods in industry.

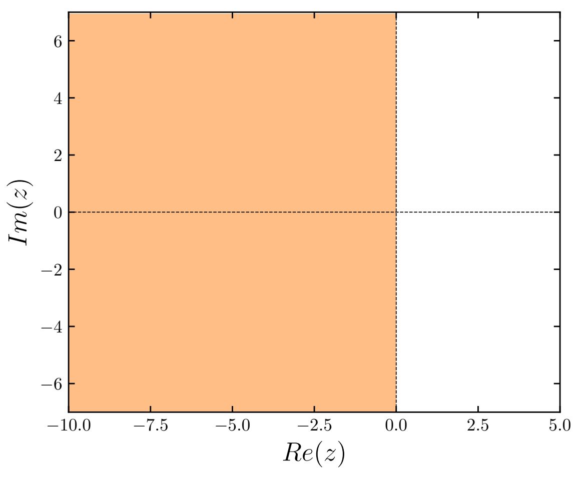
Stability of Implicit Runge-Kutta Schemes

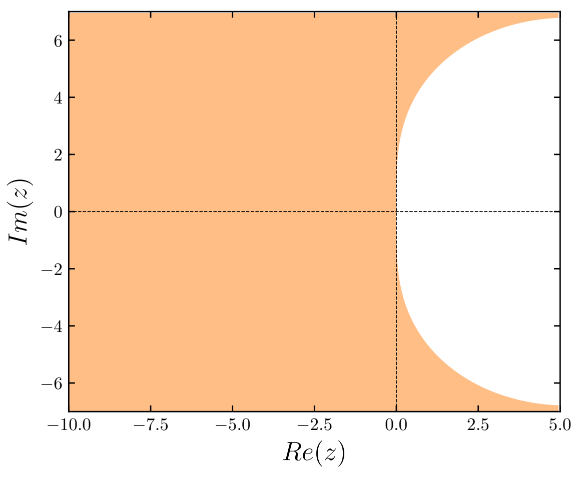
Since we are concerned with numerical stability; we recall the Runge-Kutta stability function [39] in Equation 3.37. A numerical Runge-Kutta scheme is stable for any region where $|P_{s,p}|\leq 1$ [39]. Also, an A-stable scheme is one for which $|P_{s,p}|\leq 1$ in the entire negative real plane [38, 39]. Some A-stable schemes also have the feature that $R(\infty) = 0$. These are called L-stable or stiffly accurate schemes. L-stable schemes remain strictly stable for very large time-step sizes, whereas A-stable schemes become marginally stable.

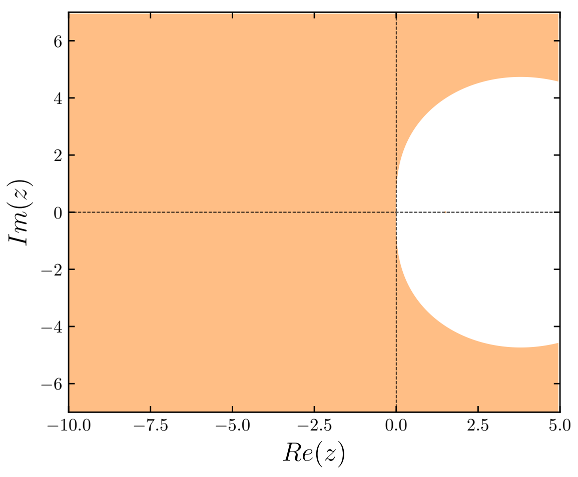
For example, the implicit Euler method, which is a first-order accurate scheme, is L-stable. On the other hand, the Crank-Nicholson scheme is a A-stable scheme, which means it is weakly stable for large time-steps. In fact, second-order Crank-Nicholson with only one implicit stage per time-step is only appropriate for equations with relatively small time-steps, which are still large for explicit schemes. The second-order SDIRK scheme with two stages (SDIRK_{22}), as well as the third-order SDIRK scheme with three stages (SDIRK_{33}), are also L-stable. They are beneficial for systems of equations with relatively large time-steps and those that are very stiff.

Figure 3.4 shows the region of absolute stability of classical implicit schemes.


(a) Backward Euler method


(b) Crank-Nicolson method


(c) $\text{SDIRK}_{2,2}$


(d) $\text{SDIRK}_{3,3}$

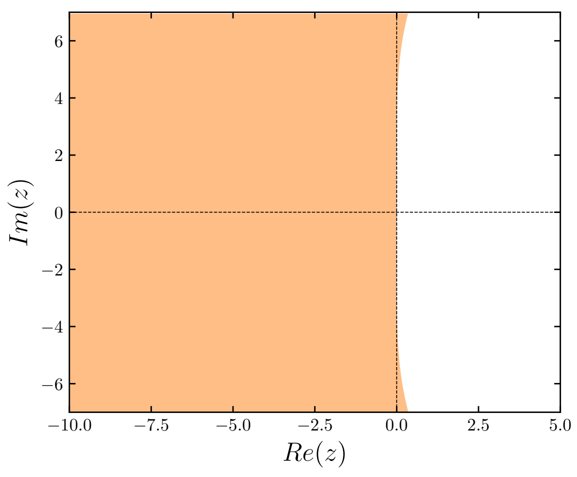

(e) $\text{SDIRK}_{5,4}$

Figure 3.4. Region of absolute stability (marigold color) of classical implicit schemes.

3.2.4 Implicit-Explicit Schemes

One is often confronted with a multi-scale system of equations. Using a fully implicit scheme in the whole domain of such a system, which is locally stiff, is not efficient in terms of cost and memory requirements. Hence, the most efficient approach is often a hybrid implicit-explicit approach [65]. This novel approach has shown the potential to reduce the cost of the solution relative to fully implicit schemes. IMEX schemes result from a combination of appropriate implicit and explicit schemes. The implicit scheme is used in stiff regions, and the explicit scheme is used in the rest of the domain. Then, the two schemes are paired coherently to maintain the order of accuracy of the total solution. The total order of accuracy of the IMEX scheme is equal to the smaller order of accuracy of the implicit and explicit schemes [54].

In general, there are two types of multi-scale systems of differential equations. Type one are those that can be split into the form [65]

$$\frac{\partial u}{\partial t} = f(u) + g(u), \tag{3.63}$$

where $f(u)$ is the non-stiff portion of the system and $g(u)$ is the stiff portion of the system of equations.

The second type are those PDEs that are stiff in some parts of their domain and non-stiff elsewhere [54]. In this case, some elements located in the stiff area will be solved using an implicit scheme, and the rest will be solved using an explicit scheme.

IMEX Runge-Kutta Schemes

The IMEX Runge-Kutta methods are, perhaps, the most popular of the available IMEX schemes. These IMEX schemes work by pairing an implicit Runge-Kutta scheme in the stiff part with a suitable explicit Runge-Kutta scheme in the non-stiff part. Popular methods use Singly-Diagonally Implicit Runge-Kutta (SDIRK) schemes for the implicit part, which are padded to have the same number of stages as the corresponding explicit Runge-Kutta scheme [54]. These SDIRK schemes are advantageous as they can use the same Jacobian matrix for each stage in Quasi Newton-Raphson iterations, reducing the number of expensive preconditioning operations. However, SDIRK schemes require a relatively large number of implicit solves, one per implicit stage, which can be expensive as the number of stages increases. Hence, the difficulty of generating SDIRK schemes with a large number of stages typically limits the maximum number of stages that can be used for any particular IMEX scheme.

A general IMEX Runge-Kutta scheme with s-stages and of order p for the explicit part and $\hat{s}$-stages and order $\hat{p}$ for the implicit part can be denoted by IMEX$_{s,p^*}$ where p^* is the smaller of p and $\hat{p}$. Such a scheme can be represented using the Butcher tableaus [38] as

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} \qquad \begin{array}{c|c} \hat{\mathbf{c}} & \hat{\mathbf{A}} \\ \hline & \hat{\mathbf{b}^T} \end{array} \tag{3.64}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ a_{2,1} & 0 & 0 & \dots & 0 \\ a_{3,1} & a_{3,2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{s,1} & a_{s,2} & a_{s,3} & \dots & 0 \end{bmatrix}, \qquad \hat{\mathbf{A}} = \begin{bmatrix} \gamma & 0 & 0 & \dots & 0 \\ \hat{a}_{2,1} & \gamma & 0 & \dots & 0 \\ \hat{a}_{3,1} & \hat{a}_{3,2} & \gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{a}_{\hat{s},1} & \hat{a}_{\hat{s},2} & \hat{a}_{\hat{s},3} & \dots & \gamma \end{bmatrix}, \tag{3.65}$$

$$c_i = \sum_{j=1}^{s} a_{i,j}, \;\; \hat{c}_i = \sum_{j=1}^{\hat{s}} \hat{a}_{i,j}, \tag{3.66}$$

and the consistency constraint

$$\sum_{i=1}^{s} b_i = 1, \;\; \sum_{i=1}^{\hat{s}} \hat{b}_i = 1, \tag{3.67}$$

must be satisfied. One additional constraint in IMEX Runge-Kutta schemes is that the quadrature weight vectors must be identical

$$b_i = \hat{b}_i. \tag{3.68}$$

For explicit schemes we also require

$$a_{i,j} = 0, \;\; j \geq i, \tag{3.69}$$

which ensures $\mathbf{A}$ is strictly lower triangular. Furthermore, for SDIRK schemes require

$$\hat{a}_{i,j} = 0, \quad j > i, \tag{3.70}$$

ensuring $\hat{\mathbf{A}}$ is lower triangular and

$$\hat{a}_{i,i} = \gamma, \tag{3.71}$$

ensuring that all diagonal coefficients for the implicit scheme are identical and equal γ.
An algorithm of an IMEX schemes with s-stages for a general differential equation that is first-order in time

$$\frac{\partial u}{\partial t} + R = 0, \tag{3.72}$$

can be expressed as

$$u_i^{[ex]} = u_n^{[ex]} + \Delta t \sum_{j=1}^{s} a_{i,j}^{[ex]} R(t_n + c_j \Delta t, u_j^{[ex]}), \tag{3.73}$$

and

$$u_i^{[im]} = u_n^{[im]} + \Delta t \sum_{j=1}^{s} a_{i,j}^{[im]} R(t_n + c_j \Delta t, u_j^{[im]}), \tag{3.74}$$

and since the weight vectors are identical (Equation 3.68)

$$u_{n+1} = u_n + \Delta t \sum_{i=1}^{s} b_i R(t_n + c_i \Delta t, u_i). \tag{3.75}$$

There are many IMEX Runge-Kutta schemes with various orders of accuracy and numbers of stages. The computational cost per time-step of these schemes is associated with the number of stages in the scheme. The accuracy is proportional to the time-step size and the order of accuracy. When the time-step is relatively small, a lower-order scheme may be appropriate since it has fewer stages and, therefore, it has a lower cost per time-step. However, a higher-order scheme may be more appropriate to reduce numerical error, with larger time-steps, particularly for the stiff regions. It should be noted that IMEX Runge-Kutta schemes generally work best for problems with a large range of stiffness. Otherwise, either a fully explicit or implicit scheme should be chosen.

The computational cost per time-step of an IMEX Runge-Kutta scheme is associated with the number of stages [66, 67], a model for the computational cost C, per time-step of an IMEX scheme, is [54]

$$C = enc + \hat{e}\hat{n}\hat{c}, \tag{3.76}$$

where e denotes the number of explicit stage derivative evaluations per step, c denotes the wall-clock time per explicit stage derivative evaluation per element, and n represents the total number of explicit elements. Similarly, $\hat{e}$ denotes the number of implicit stage derivative evaluations per step, $\hat{c}$ denotes the wall-clock time per implicit stage derivative evaluation per element, and $\hat{n}$ represents the total number of implicit elements. For conventional IMEX schemes, $e = s$ and $\hat{e} = \hat{s} - 1$, where s is the number of stages in the explicit scheme and $\hat{s}$ is the number of stages [54, 66] in the padded implicit scheme. If we define the implicit fraction as $\eta = \hat{n}/(n + \hat{n})$ and normalize by the time-step size Δt and number of elements, we can write [54]

$$\tilde{C} = \frac{(1 - \eta)ec + \eta\hat{e}\hat{c}}{\Delta t}, \tag{3.77}$$

where $\tilde{C}$ is the total computational cost per unit time per element. The value Δt is typically chosen to be as large as possible without exceeding the stability limit of the explicit scheme. Furthermore, it is often observed that $\hat{c} \gg c$, as the implicit solver requires the solution of a large system of equations. Therefore, provided η is sufficiently large, the computational cost of an IMEX scheme can be estimated as

$$\tilde{C} \approx \frac{\eta\hat{s}\hat{c}}{\Delta t}. \tag{3.78}$$

From Equation 3.78 there are four possible approaches to reducing the cost of a simulation using an IMEX Runge-Kutta scheme when it is dominated by the implicit part [66]. Reducing η by moving elements from the implicit part to the explicit part, reducing the number of implicit stage derivative evaluations $\hat{e}$, reducing the computational cost per derivative evaluation $\hat{c}$, or increasing the time-step size Δt. Reducing η is often not advantageous as it requires a reduction in Δt as stiffer elements are moved to the explicit part. Reducing the number of stage derivatives $\hat{e}$ is not desirable as it

requires a less-accurate implicit RK scheme. Reducing $\hat{c}$ has been the focus of significant previous research on general linear and non-linear solvers. Noting that the numerator of $\tilde{C}$ is dominated by the implicit solver, but the maximum value of Δt is limited by the explicit solver's stability, increasing the maximum allowable Δt by optimizing its stability polynomial [65, 66].
The implementation of IMEX scheme can be found in the work of Pareschi et al. [68], Vermeire et al. [66] and Persson et al. [60].

Stability and Optimization of IMEX Runge-Kutta Schemes

One advantage of using an SDIRK scheme for the implicit part of the IMEX scheme is that SDIRK schemes are often L-stable [66]. Hence, this makes the stiff part of the solution unconditionally stable. Accordingly, the stability of the IMEX scheme is determined by the explicit part. However, since the explicit scheme is only used in the non-stiff regions, its stability polynomial can be optimized for relatively large time-steps.

Accelerated IMEX Schemes

Based on the previous section, the total computational cost of an IMEX approach can be reduced by increasing the time-step size, provided the cost of the implicit solver does not increase significantly. To achieve this, a novel family of second-order Accelerated IMEX (AIMEX) schemes was introduced by Vermeire et al. [67]. To generate this family, we modify the combination of a three-stage explicit scheme of the form [65]

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ c_2 & a_{2,1} & 0 & 0 \\ c_3 & a_{3,1} & a_{3,2} & 0 \\ \hline & b_1 & b_2 & b_3 \end{array} \tag{3.79}$$

with a padded two-stage implicit scheme of the form [65]

$$\begin{array}{c|c} \hat{\mathbf{c}} & \hat{\mathbf{A}} \\ \hline & \hat{\mathbf{b}} \end{array} = \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \hat{c}_2 & 0 & \hat{a}_{2,2} & 0 \\ \hat{c}_3 & 0 & \hat{a}_{3,2} & \hat{a}_{3,3} \\ \hline & 0 & \hat{b}_2 & \hat{b}_3 \end{array} \tag{3.80}$$

which yields the well-known family of IMEX$_{3,2}$ schemes [65]. An L-stable second-order SDIRK scheme of this form is

$$\begin{array}{c|c} \hat{\mathbf{c}} & \hat{\mathbf{A}} \\ \hline & \hat{\mathbf{b}} \end{array} = \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \gamma & 0 & \gamma & 0 \\ 1 & 0 & 1-\gamma & \gamma \\ \hline & 0 & 1-\gamma & \gamma \end{array} \tag{3.81}$$

where $\gamma = \left(2 - \sqrt{2}\right)/2$ [65]. Applying the constraints for an IMEX scheme

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \gamma & \gamma & 0 & 0 \\ 1 & 1-a_{3,2} & a_{3,2} & 0 \\ \hline & 0 & 1-\gamma & \gamma \end{array} \tag{3.82}$$

where the commonly used IMEX$_{3,2}$ scheme is obtained by setting $a_{3,2} = 1 + 2\sqrt{2}/3$ [67]. To create our new family of AIMEX schemes, we first modify the SDIRK scheme by adding additional explicit stages between the two implicit stages

$$\begin{array}{c|c} \hat{\mathbf{c}} & \hat{\mathbf{A}} \\ \hline & \hat{\mathbf{b}} \end{array} = \begin{array}{c|cccccc} 0 & 0 & & & & & \\ \gamma & 0 & \gamma & & & & \\ \hat{c}_3 & 0 & \hat{a}_{3,2} & 0 & & & \\ \hat{c}_4 & 0 & \hat{a}_{4,2} & \hat{a}_{4,3} & \ddots & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\ 1 & 0 & 1-\gamma & \hat{a}_{s,3} & \dots & \hat{a}_{s,s-1} & \gamma \\ \hline & 0 & 1-\gamma & 0 & \dots & 0 & \gamma \end{array} \tag{3.83}$$

noting that these additional stages are not used directly in updating the solution to the next time-step based on the location of the zero components of $\hat{\mathbf{b}}$ [67].

$$\hat{a}_{ij} = 0, \;\; 1 \leq i \leq s, \;\; 3 \leq j \leq s-1. \tag{3.84}$$

This yields the following Butcher tableau

$$\begin{array}{c|c} \hat{\mathbf{c}} & \hat{\mathbf{A}} \\ \hline & \hat{\mathbf{b}} \end{array} = \begin{array}{c|cccccc} 0 & 0 & & & & & \\ \gamma & 0 & \gamma & & & & \\ \hat{c}_3 & 0 & \hat{a}_{3,2} & 0 & & & \\ \hat{c}_4 & 0 & \hat{a}_{4,2} & 0 & \ddots & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\ 1 & 0 & 1-\gamma & 0 & \dots & 0 & \gamma \\ \hline & 0 & 1-\gamma & 0 & \dots & 0 & \gamma \end{array} \tag{3.85}$$

This enables us to evaluate the solution at additional intermediate stages without computing any additional stage derivatives due to the non-zero structure of the $\hat{\mathbf{b}}$ vector. We also take $\hat{c}_s = 1$, and

$\hat{c}_2 = \gamma$ and the other components of c vector as

$$\hat{c}_i = \frac{(1-\gamma)(i-2)}{s-2} + \gamma, \;\; i \le 3 \le s-1. \tag{3.86}$$

Similar to the implicit part, we add additional stages to the explicit scheme between the second and last stages to interface with the implicit scheme.

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|cccccc} 0 & 0 & & & & & \\ \gamma & \gamma & 0 & & & & \\ c_3 & a_{3,1} & a_{3,2} & 0 & & & \\ c_4 & a_{4,1} & a_{4,2} & a_{4,3} & \ddots & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\ 1 & a_{s,1} & a_{s,2} & a_{s,3} & \dots & a_{s,s-1} & 0 \\ \hline & 0 & 1-\gamma & 0 & \dots & 0 & \gamma \end{array} \tag{3.87}$$

This leads to a large number of stage derivatives in the explicit part that requires a significant amount of memory to be stored. To reduce memory consumption, another constraint is used as

$$a_{i,j} = 0, \;\; 4 \le i \le s, \;\; 2 \le j \le i-2. \tag{3.88}$$

Including the constraint $\mathbf{c} = \hat{\mathbf{c}}$ produces the following Butcher tableau for the explicit scheme

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|cccccc} 0 & 0 & & & & & \\ \gamma & \gamma & 0 & & & & \\ \hat{c}_3 & a_{3,1} & a_{3,2} & 0 & & & \\ \hat{c}_4 & a_{4,1} & 0 & a_{4,3} & \ddots & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 & \\ 1 & a_{s,1} & 0 & \dots & 0 & a_{s,s-1} & 0 \\ \hline & 0 & 1-\gamma & 0 & \dots & 0 & \gamma \end{array} \tag{3.89}$$

With AIMEX schemes, for each additional stage added to the implicit scheme, an extra coefficient will be added to the stability polynomials of the explicit scheme. These extra coefficients can be optimized for a given spatial discretization to improve the region of the absolute stability of the explicit scheme. In general, second-order AIMEX schemes designed with a special s number of stages have an implicit scheme with two stage derivatives and an explicit scheme with s stages that can be optimized for a given spatial discretization to take the maximum time-step size.

3.2.5 Heterogeneous Computing

Modern hardware architectures have become sufficiently powerful to solve systems of differential equations. This has been demonstrated on massively parallel systems using explicit schemes [69]. However, this approach becomes rapidly impractical in the case of stiff partial differential equations due to mesh resolution requirements in space. Consequently, explicit solvers are hindered as they require prohibitively small time-steps to maintain numerical stability. In contrast, implicit solvers require full global coupling of the entire domain and suffer from poor parallel performance.

It is expected that next-generation computer hardware architectures will consist of massively parallel heterogeneous combinations of conventional Central Processing Units (CPUs) and accelerator hardware architectures, such as Graphical Processing Units (GPUs). For example, the leadership-class Compute Canada Cedar system, the forthcoming Summit and Sierra supercomputers run by the United States Department of Energy, and the Piz Daint supercomputer at the Swiss National Computing Centre. CPUs typically contain between one to ten individual cores and are well-suited for low-latency, low-throughput calculations. This means CPUs are particularly appealing for unstructured data formats and numerical methods relying on low-latency for individual instructions rather than the ability to execute a large number of instructions per clock cycle. In contrast, accelerators such as GPUs are well suited for high-latency, high throughput operations, as they have a relatively high FLOP/s (Floating Point Operations per Second) rate that is achieved via massive parallelism across several thousand compute cores, but they execute each instruction in a high-latency pipelined fashion.

Solvers have historically made use of algorithms specifically well-suited for CPUs or well-suited for GPUs. In this homogeneous computing paradigm, these solvers have typically been run using only the hardware architecture for which they were designed. This means that implicit schemes have been used on CPUs, due to their indirect unstructured memory access patterns. In contrast, explicit schemes have been used on GPUs, due to their low parallelism overhead and the ability to cast many mathematical expressions as level 3 BLAS (Basic Linear Algebra Subroutines) calls exploiting the high FLOP/s to memory bandwidth ratios of modern GPU hardware. Based on previous work, it is expected that implicit schemes will continue to be well suited for CPUs, whereas explicit schemes will continue to be well suited for GPUs.

3.2.6 Discussion

This Section lays the groundwork for an understanding of temporal schemes for numerical solution of differential equations with regards to new developments in algorithms and high-performance computing. Generally, temporal schemes can be scrutinized in terms of accuracy, efficiency, stability and simplicity of implementation. Table 3.6 gives an overview of Runge-Kutta schemes in terms of these criteria. Explicit Runge-Kutta schemes are accurate, they are relatively easy to implement, and they are compatible with novel heterogeneous hardware. However, they are inefficient because of limited stability. This thesis addresses the efficiency and stability of explicit Runge-Kutta schemes and proposes novel optimal explicit RK methods that have improved stability and are efficient once paired with high-order spatial discretizations.

Table 3.6. Comparison of Runge-Kutta schemes in terms of accuracy, efficiency, stability and simplicity of implementation.

Temporal Schemes	Accuracy	Efficiency	Stability	Simplicity	Heterogeneous Computing
Explicit Runge-Kutta Methods	✓	✗	✗	✓	✓
Implicit Runge-Kutta Methods	✓	✗	✓	✗	✗
IMEX Runge-Kutta Methods	✓	✓	●	✗	✓

Chapter 4

Optimal Runge-Kutta Stability Polynomials for Multidimensional Flux Reconstruction

4.1 Authorship Statement

The multi-dimensional Von-Neumann Analysis of the FR approach was undertaken by Carlos Pereira. Apart from that, all of the contributions in this section were produced by Siavash Hedayati Nasab.

4.2 Background

Explicit Runge-Kutta methods are widely used for the solution of non-stiff systems of equations, such as the advection, Euler, Navier-Stokes, shallow water wave, and Maxwell's equations, amongst many others. They are often utilized to advance high-order unstructured spatial discretizations of these physical problems in time via the method of lines, such as the Discontinuous Galerkin (DG), Spectral Difference (SD), Spectral Volume (SV), and Flux Reconstruction (FR) approaches [6]. Explicit schemes are often appealing due to their ease of implementation and scalability [38]. However, their stability constraints, particularly with stiff systems of equations, limit their maximum permissible time-step size [39, 70]. Hence, explicit schemes are typically limited to non-stiff problems.

The aforementioned linear stability and accuracy of explicit Runge-Kutta methods are determined by their so-called stability polynomial [59]. Previous research has focused on optimizing these stability polynomials for a given spatial discretization and system of equations. These stability polynomials are typically optimized to enable the largest possible time-step size for a given number of Runge-Kutta stages, reducing computational cost [55] and enabling the use of explicit methods for problems of increasing stiffness. For example, we have seen the application of optimization of Runge-Kutta stability polynomials in the works of Van der Houwen et al. [71], Ruuth et al. [72, 73], Ketcheson et al. [59], Parsani et al. [74] Kubatko et al. [55], and Vermeire et al. [75]. Recently, novel temporal schemes that leverage these stability polynomials for locally stiff systems of equations have been proposed such Accelerated Implicit Explicit (AIMEX) methods [67], which are an extension of classical IMEX schemes [54, 65, 66, 76, 77]. However, in the context of the Discontinuous Galerkin (DG) and Flux reconstruction (FR) approaches, and the P-ERK and AIMEX temporal schemes, these stability polynomials have as yet only been generated for one-dimensional elements. The stability of these polynomials, optimized for one-dimensional elements, for multidimensional problems has yet to be explored. Furthermore, optimal stability polynomials for multidimensional problems using DG and FR have not yet been found.

Hence, in this chapter, the overall objective is optimization of stability polynomials for different orders of accuracy in time, different numbers of Runge-Kutta stages, and different multidimensional element types, including hexahedral, tetrahedral, prismatic, quadrilateral and triangular, using FR spatial discretizations with various orders of accuracy in space. These will be compared to the one-dimensional optimal polynomials, as well as classical Runge-Kutta methods, to determine what performance improvements can be obtained relative to these previous methods. Finally, the utility of these schemes will be explored for the compressible Navier-Stokes equations.

4.3 Multidimensional Von Neumann Analysis of FR

The spectral properties of high-order semi discretizations have been thoroughly studied in the context of linear advection via Von Neumann analysis in one [10, 78] and higher dimensions [79]. Fully-discrete analyses have shown the effects of the temporal schemes on the spatial discretizations. Yang et al. [80] analyzed different first and second-order RK methods on DG discretizations. Vermeire et al. [37] investigated the properties of conventional explicit and implicit RK methods in the context of ILES for FRDG schemes and ESFR schemes [81]. Pereira et al. [82] showed that the spectral properties of optimal RK schemes are similar to those of conventional RK methods away from the stability limits.

In order to generate the aforementioned optimal stability polynomials of n-dimensional spatial FR schemes, we perform Von Neumann analysis to obtain the eigenvalues ω^δ of the semi discretization. The analysis consists of evaluating a scheme's wave propagation properties by considering the n-dimensional linear advection equation

$$\frac{\partial \mathbf{u}}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{\alpha}\mathbf{u}) = 0, \tag{4.1}$$

where $u = u(\boldsymbol{x}, t)$ is the scalar solution variable, $\boldsymbol{\alpha} u$ is the linear advection flux with $\boldsymbol{\alpha}$ the advection velocity, t is time and $\boldsymbol{x}$ is the spatial coordinate. The advection direction is defined by the angles θ_1, θ_2, with $\theta_2 = 0$ in the case of $n = 2$ (see Figure 4.1).

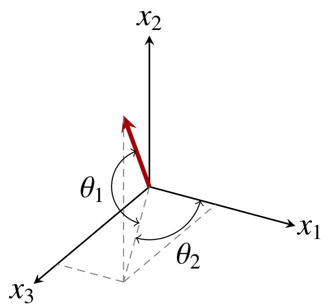


Figure 4.1. Vector decomposition reference

We divide a computational domain Ω into N_e elements. We consider triangular and quadrilateral element types for $n = 2$ and hexahedral, prismatic and tetrahedral elements for $n = 3$. This division must be performed ensuring that Ω is periodic in all directions. Hence, N_d non-periodic elements are agglomerated into a referential periodical element with edge length h, as shown in Figure 4.2. We consider h to be unity for all edges.

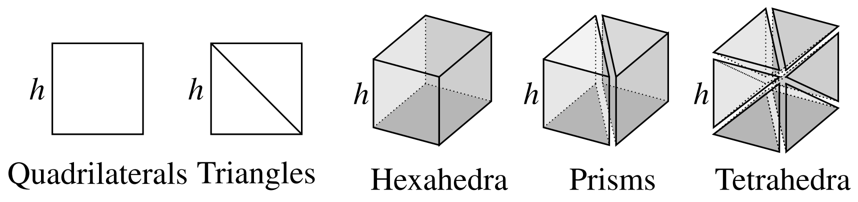


Figure 4.2. Subdivision of element types for $n = 2$ and $n = 3$

Next, we seek plane wave solutions of the form

$$u(\boldsymbol{x}, t) = e^{I(\boldsymbol{\kappa}\cdot\boldsymbol{x} - \nu t)}, \tag{4.2}$$

where $\boldsymbol{\kappa} = |\boldsymbol{\kappa}|\,(\beta_1, \beta_2)$ defines the wavenumber, ν is the frequency of the wave and $I = \sqrt{-1}$ is the imaginary number. We consider $|\boldsymbol{\alpha}| = 1$ and hence the exact dispersion relation is simply $\nu = |\boldsymbol{\kappa}|$. After a simple projection, we may write for any given element

$$\boldsymbol{u}^{\delta}(\boldsymbol{x}, t) \equiv \boldsymbol{u}_i^{\delta}(\boldsymbol{x}, t) = e^{I(\boldsymbol{\kappa}\cdot\boldsymbol{x}_i - \nu^{\delta} t)}\boldsymbol{U}, \tag{4.3}$$

where $\boldsymbol{U}$ is an unknown vector that contains the amplitudes of the wave in numerical space, and $\boldsymbol{x}_i$ is the center coordinate of the element. Note that the solution within any given element is the same and hence the choice of the subscript $i \in [1, N_e]$ is arbitrary. After discretizing the linear advection equation using FR with an upwind Riemann flux and the δ correction field that yields a DG scheme, we can rewrite Equation 4.3 at the discrete level

$$\frac{d\boldsymbol{u}^{\delta}}{dt} = \boldsymbol{L}\boldsymbol{u}^{\delta}, \tag{4.4}$$

where L is the semidiscrete matrix of dimension $N_d N_s \times N_d N_s$, which depends on the wavevector $\boldsymbol{\kappa}$, advection direction, δ coefficients, Riemann flux choice, and element type. The matrix L contains the contributions from the neighboring elements, and can be written in general for any n number of dimensions

$$L = \left[C^0 + \sum_{i=1}^{n} \left(C_u^i e^{-I\boldsymbol{\kappa}\cdot\hat{\mathbf{x}}^{\mathbf{i}}} + C_d^i e^{+I\boldsymbol{\kappa}\cdot\hat{\mathbf{x}}^{\mathbf{i}}} \right) \right], \tag{4.5}$$

where $\hat{x}^i,\ i = 1, \ldots, n$ is a unit vector, C^0 is a matrix associated with the element in study, and $C_u^i,\ C_d^i$ are respectively the upstream element and the downstream element Jacobian matrices along the i-th direction. Note that the second term inside the summation in Equation 4.5 vanishes when an upwind Riemann flux is implemented. After substituting Equation 6.24 into 4.4, we obtain

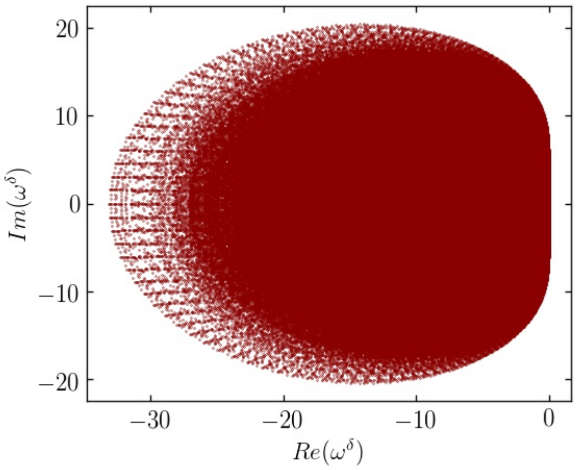
$$-I\nu^{\delta}\boldsymbol{U} = L\boldsymbol{U}, \tag{4.6}$$

which is clearly a classical eigenvalue problem with $N_d N_s$ eigenvalues $\boldsymbol{\omega}^{\delta} \in \mathbb{C}$. Then these eigenvalues are related to numerical frequencies by

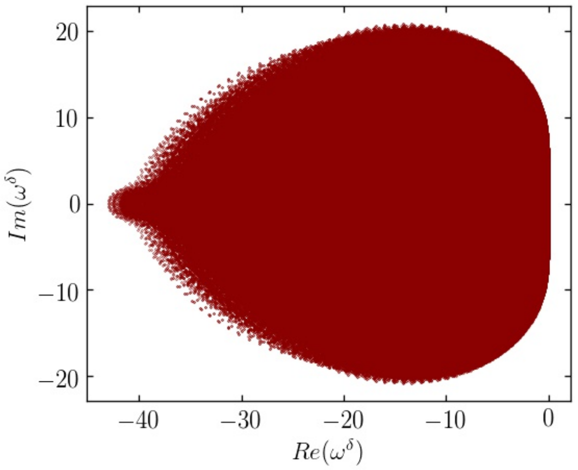
$$\nu^{\delta} = I\omega^{\delta}. \tag{4.7}$$

The spectrum of the eigenvalues considering all wavenumbers and wavevector orientations define the stability properties of the spatial discretization and are scaled by the time-step size to fit within the stability regions of the explicit temporal schemes. In addition, the imaginary part of the numerical frequencies must be nonpositive to ensure boundedness [10, 79, 83]. Element types with larger eigenspectra typically require smaller time-step sizes, similar to the effects of increasing the polynomial degree [84]. However, this condition is also influenced by the number of solution points $N_d N_s$ within the referential element.

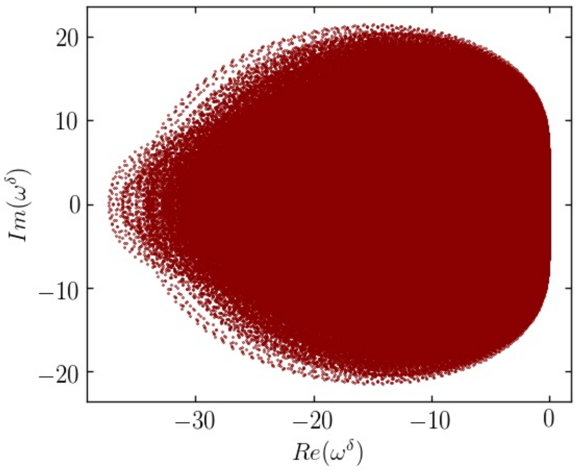
Figure 6.2 displays the collection of eigenvalues for all considered element types using a solution polynomial degree $k = 4$ spatial discretization for resolvable wavenumbers and orientations. We note that using even double precision in the computation of eigenvalues may yield numerical frequencies with positive spurious real components on the order of machine precision. As a consensus, these values were removed from the results for the computation of the optimal stability polynomials, discussed in the following sections.



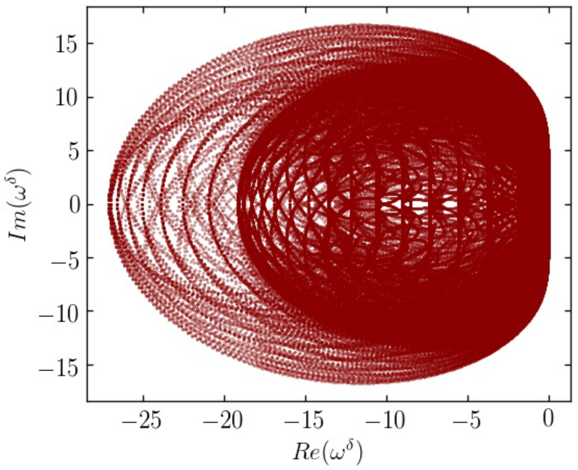
(a) Hexahedral element.



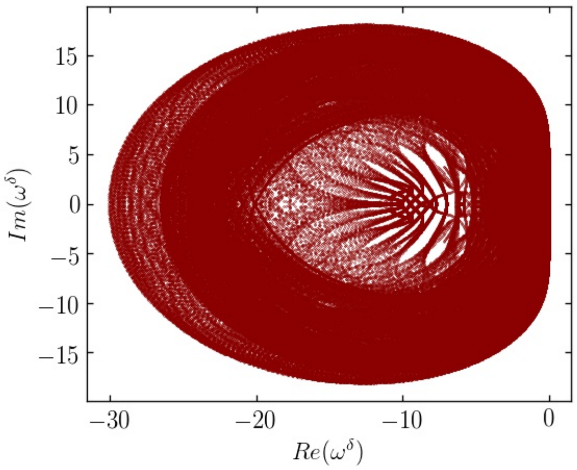
(b) Tetrahedral element.



(c) Prismatic element.



(d) Quadrilateral element.



(e) Triangular element.

Figure 4.3. Collection of eigenvalues for a fourth-order ($k = 3$) spatial discretization for all considered element types

After determining the eigenspectra for each element type, we use Algorithm 1 to generate stability polynomials for different solution polynomial degree $k = 0$ to $k = 6$, and temporal order of accuracy $p = 1$ to $p = 4$, and up to $s = 16$ stages. In the current study, we converge the bisection method in Algorithm 1 to 10^{-12} using Matlab 2016a [85]. To solve the convex minimization problem in Equation 3.42, we used CVX [86], which is a Matlab-based package for specifying and solving convex programs, using the "best" precision setting [86]. Following this approach, we can generate an optimal stability polynomial for a given system of equations.

4.4 Optimized Stability Polynomials

4.4.1 One-Dimensional Schemes

Using the one-dimensional Von Neumann analysis proposed by Huynh [6], and the stability polynomial optimization procedure in Section 1, optimal stability polynomials have been generated for one-dimensional elements using solution polynomials of degree $k = 1, 2, 3, 4, 5$, and 6 and temporal orders of accuracy $p = 1, 2, 3$, and 4 with up to $s = 16$ stages. Figure 4.4a and 4.4b show plots of the optimal time step size for $k = 0$ and $k = 6$ for all orders of accuracy in time as a function of the number of stages. An approximately linear increase in time-step size is observed as the number of stages increases, which is consistent with previous studies [55]. For $k = 0$ there is an approximately constant penalty, in terms of the magnitude of Δt_{opt}, when using higher-order temporal schemes, regardless of the number of stages. However, we note that for higher-order spatial discretizations the second-order accurate temporal scheme achieves nearly the exact same Δt_{opt} as the first-order scheme when using large stage counts. Hence, this implies that second-order accuracy can be obtained with minimal additional cost relative to a first-order temporal scheme, when using a high-order spatial discretization. Figure 4.16a shows the relative performance of each scheme, measured as $\frac{\Delta t_{opt}}{s}$, for each order in time. These results show that increasing the number of stages, and resulting terms in the stability polynomial, steadily increase the maximum stable time step size relative to the number of stages. Also, whereas the lower-order temporal schemes with $p = 1$ and 2 do not benefit significantly from large stage counts, the higher-order temporal schemes continue to benefit beyond $s = 16$ stages.

Figure 4.5 shows the eigenspectra obtained from Von Neumann analysis for $k = 4$ scaled by Δt_{opt}, and the region of absolute stability of their corresponding optimal stability polynomials for $p = 1$ and 4 with $s = 5$ and 16 stages. As required for stability, all of the eigenvalues are contained within their corresponding regions of absolute stability. As shown in Figures 4.5a and 4.5b, even for a relatively small number of stages, the region of absolute stability closely approximates the shape of the eigenspectra. However, when the temporal order of accuracy is increased, some regions of the stability polynomial protrude significantly beyond the scaled eigenspectra. This is due, in part, to the higher-order temporal schemes having fewer degrees of freedom for optimization than the lower-order schemes. However, as the number of stages is increased the region of absolute stability also eventually approximates the shape of the eigenspectra from the spatial discretization.

4.4.2 Two-Dimensional Schemes

Using the multi-dimensional Von Neumann analysis presented in Section 4.3, and the stability polynomial optimization procedure in Section 1, optimal stability polynomials have been generated for quadrilateral and triangular elements using $k = 1, 2, 3, 4, 5,$ and 6 and $p = 1, 2, 3,$ and 4 with up to $s = 16$ stages. Rather than use the entire eigenspectra, their concave hull was first extracted to reduce the number of points evaluated in the optimization procedure [59]. Figures 4.6a, 4.6b, 4.8a, and 4.8b show plots of the optimal time step size for $k = 0$ and $k = 6$ for all orders of accuracy in time as a function of the number of stages for both the quadrilateral and triangular elements, respectively. Similar to the one-dimensional elements, an approximately linear increase in time-step size is observed as the number of stages increases. For $k = 0$ there is still an approximately constant penalty, in terms of the magnitude of Δt_{opt}, when using higher-order temporal schemes for both element types. Also, similar to the one-dimensional elements, both quadrilateral and triangular elements have relatively small reductions in Δt_{opt} for high-orders in space and time, relative to their respective first-order accurate temporal schemes. Figures 4.16b and 4.16c show the relative performance of each temporal scheme for $k = 4$ with both quadrilateral and triangular elements, respectively. Similar to the one-dimensional element type, increasing the number of stages continues to improve the performance of the higher-order temporal schemes with $p = 3$ and 4, even beyond $s = 16$ stages.

Figures 4.7 and 4.9 show the concave hulls of the eigenspectra obtained from Von Neumann analysis for $k = 4$ scaled by Δt_{opt}, and the region of absolute stability of their corresponding optimal stability polynomials for $p = 1$ and 4 with $s = 5$ and 16 stages for quadrilateral and triangular elements, respectively. The concave hull and regions of absolute stability of the quadrilateral elements are visually indistinguishable from those of the one-dimensional element results. However, the results for the triangular elements is significantly different. While the general trend is the same, in that the schemes with more stages conform generally to the shape of the eigenspectra, they resulting stability regions are visually different. This is due to the protrusion of the eigenspectra, and their resulting convex hull, towards the negative real axis. Nevertheless, for all polynomial degrees these eigenspectra are contained within their respective regions of absolute stability, and this region conforms to the concave hull of the eigenspectra as the number of stages increases.

4.4.3 Three-Dimensional Schemes

Using the multi-dimensional Von Neumann analysis presented in Section 4.3, and the stability polynomial optimization procedure in Section 1, optimal stability polynomials have been generated for hexahedral, prismatic and tetrahedral elements using $k = 1, 2, 3, 4, 5$, and 6 and $p = 1, 2, 3$, and 4 with up to $s = 16$ stages. Rather than use the entire eigenspectra, their concave hull was first extracted to reduce the number of points evaluated in the optimization procedure [59]. Figures 4.10a, 4.10b, 4.12a, 4.12b, 4.14a, and 4.14b show plots of the optimal time step size for $k = 0$ and $k = 6$ for all orders of accuracy in time as a function of the number of stages for hexahedral, prismatic and tetrahedral elements, respectively. Similar to the one-dimensional and two dimensional elements, an approximately linear increase in time-step size is observed as the number of stages increases. For $k = 0$ there is still an approximately constant penalty, in terms of the magnitude of

Δt_{opt}, when using higher-order temporal schemes for all three types of elements. Also, similar to the one-dimensional and two dimensional elements, all three hexahedral, prismatic and tetrahedral elements have relatively small reductions in Δt_{opt} for high-orders in space and time, relative to their respective first-order accurate temporal schemes. Figures 4.16d, 4.16e and 4.16f show the relative performance of each temporal scheme for $k = 4$ with hexahedral, prismatic and tetrahedral elements, respectively. Similar to the one and two dimensional element type, increasing the number of stages continues to improve the performance of the higher-order temporal schemes with $p = 3$ and 4, even beyond $s = 16$ stages.

Figures 4.11, 4.13 and 4.15 show the concave hulls of the eigenspectra obtained from Von Neumann analysis for $k = 4$ scaled by Δt_{opt}, and the region of absolute stability of their corresponding optimal stability polynomials for $p = 1$ and 4 with $s = 5$ and 16 stages for hexahedral, prismatic and tetrahedral elements, respectively. The concave hull and regions of absolute stability of the hexahedral elements are visually indistinguishable from those of the one-dimensional and quadrilateral element results. However, the results for the prismatic and tetrahedral elements are significantly different. While the general trend is the same, in that the schemes with more stages generally conform to the shape of the eigenspectra, they resulting stability regions are visually different. This is due to the protrusion of the eigenspectra, and their resulting convex hull, towards the negative real axis. Nevertheless, for all polynomial degrees, these eigenspectra are contained within their respective regions of absolute stability, and this region conforms to the concave hull of the eigenspectra as the number of stages increases.

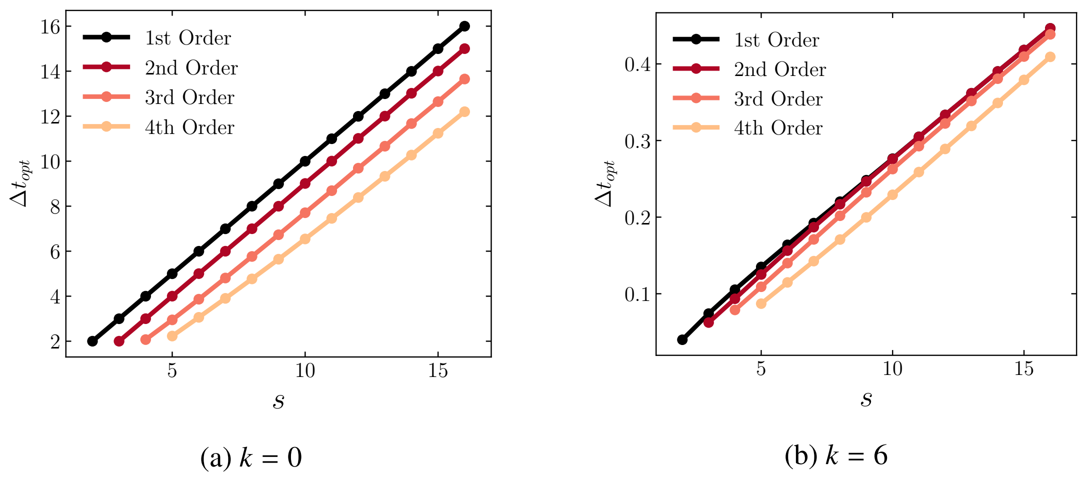


Figure 4.4. Plots of the Δt_{opt} as a function of the number of stages for line elements with $k = 0$ and $k = 6$

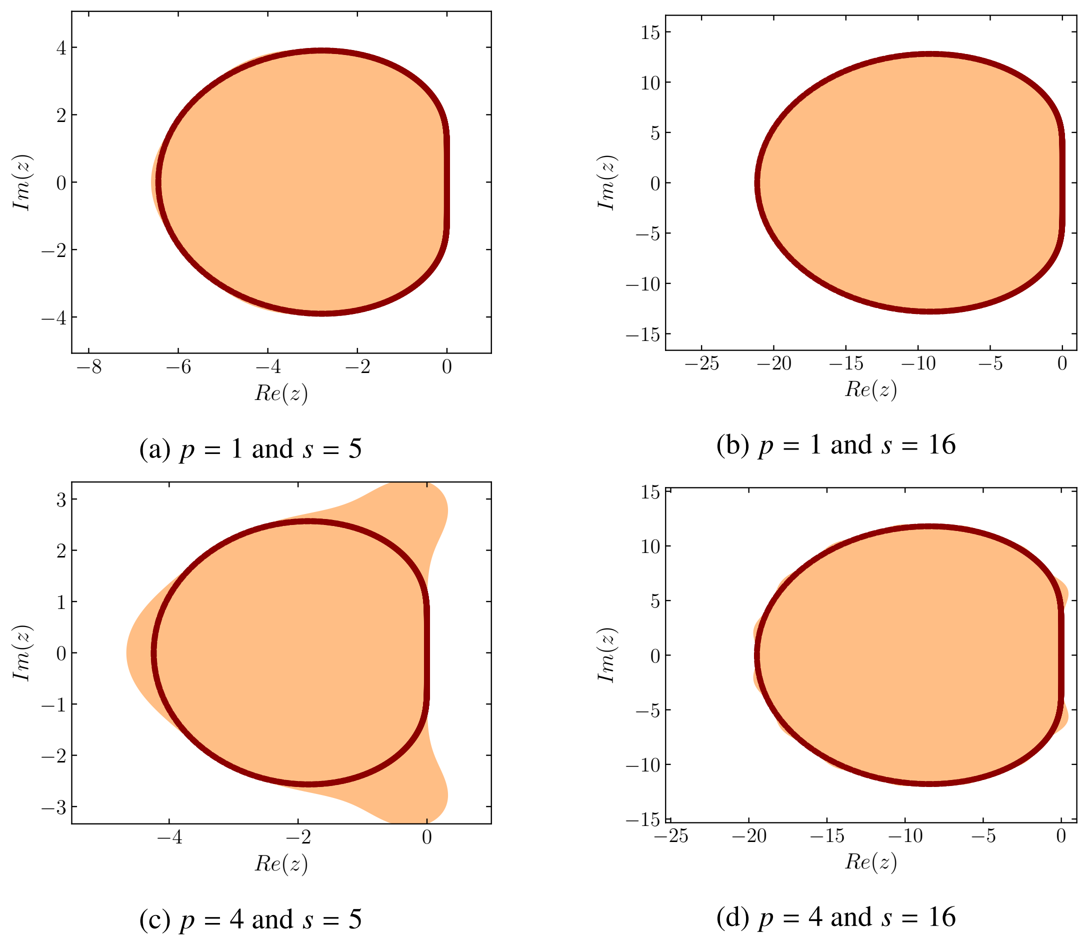
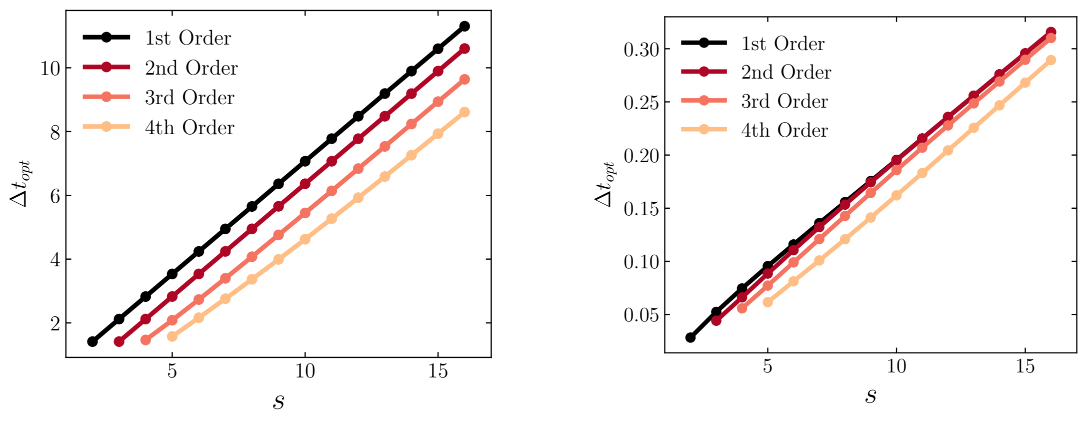
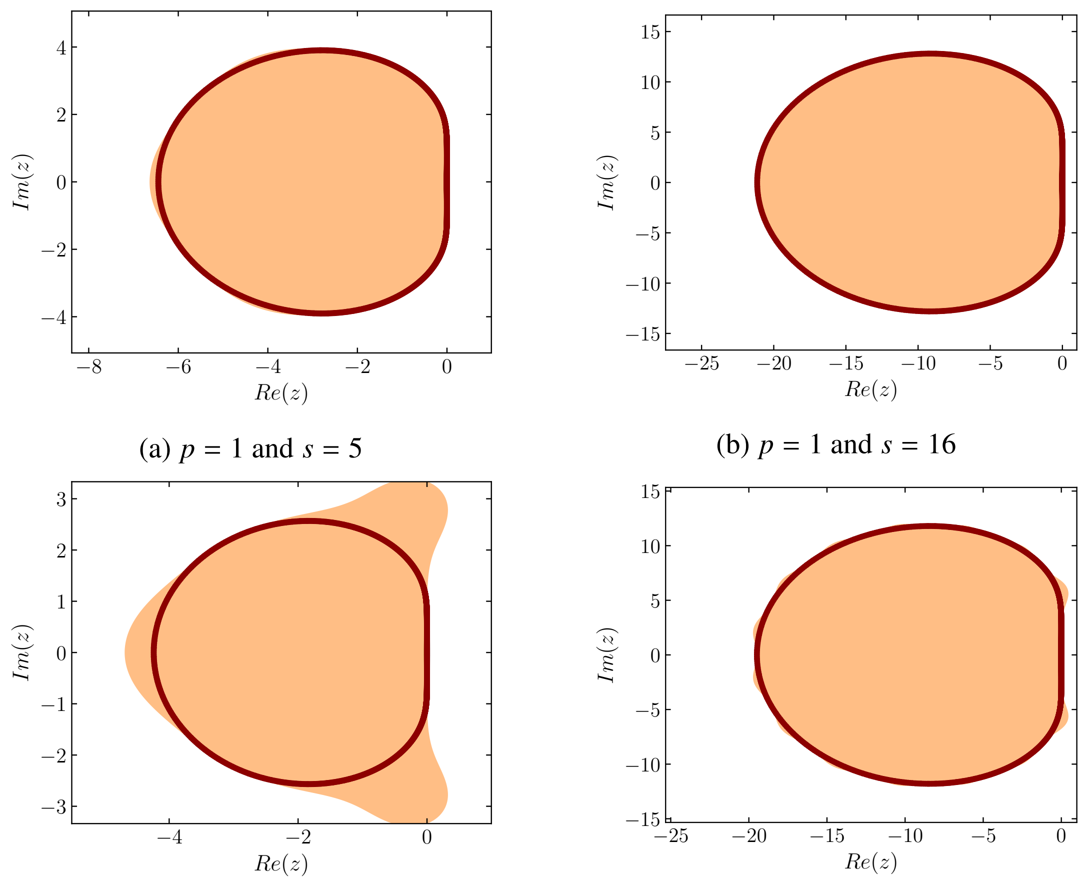


Figure 4.5. Semidiscrete eigenvalues scaled by Δt_{opt} (red) shown within the region of absolute stability of their corresponding stability polynomials for line elements with $k = 4$



(a) $k = 0$ (b) $k = 6$

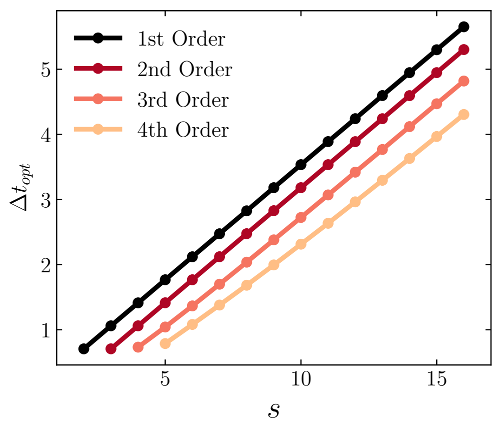
Figure 4.6. Plots of the Δt_{opt} as a function of the number of stages for quadrilateral elements with $k = 0$ and $k = 6$



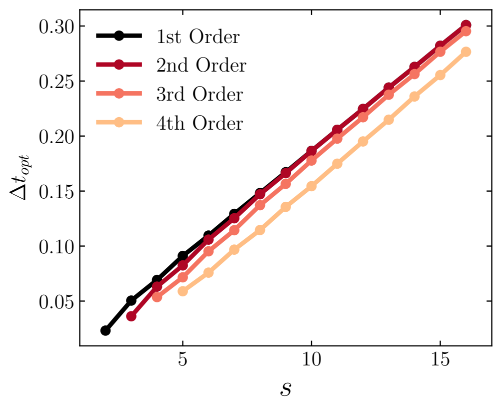
(a) $p = 1$ and $s = 5$ (b) $p = 1$ and $s = 16$

(c) $p = 4$ and $s = 5$ (d) $p = 4$ and $s = 16$

Figure 4.7. Concave hull of semidiscrete eigenvalues scaled by Δt_{opt} (red) shown within the region of absolute stability of their corresponding stability polynomials for quadrilateral elements with $k = 4$



(a) $k = 0$



(b) $k = 6$

Figure 4.8. Plots of the Δt_{opt} as a function of the number of stages for triangular elements with $k = 0$ and $k = 6$

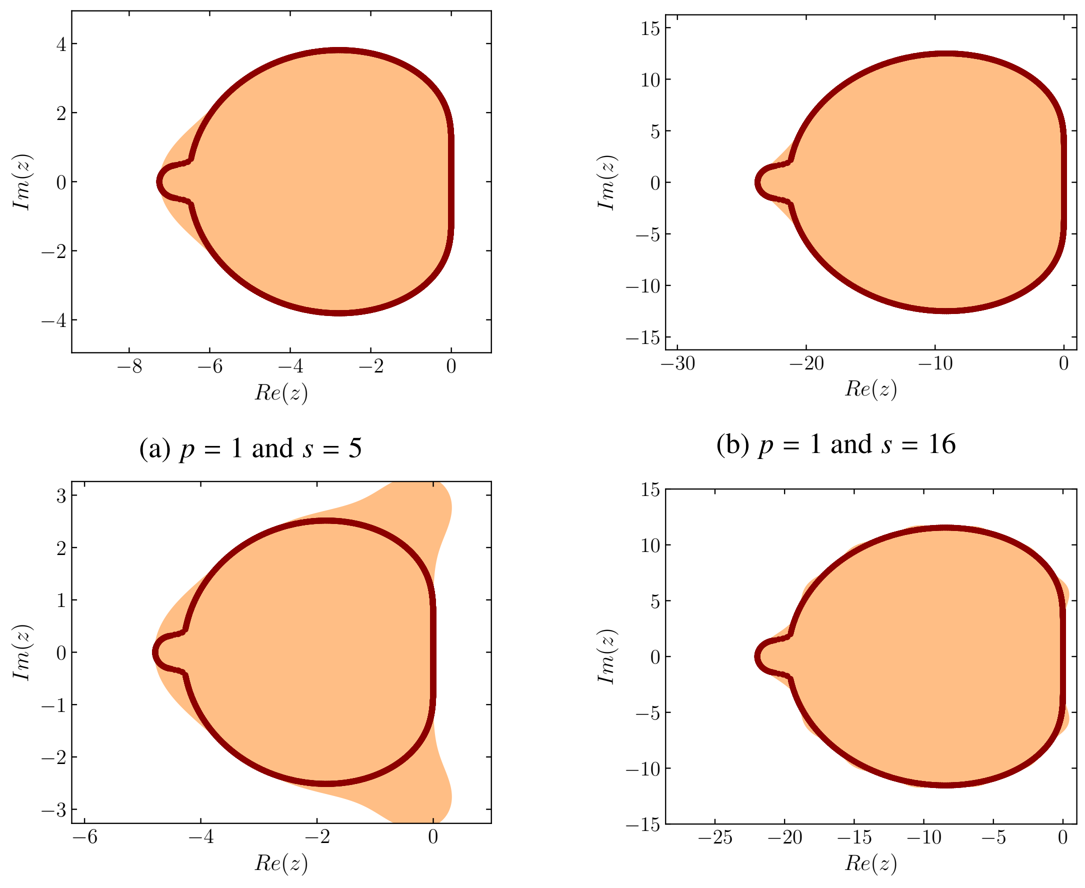
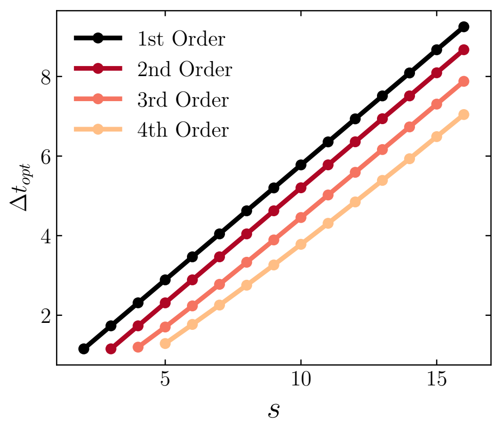
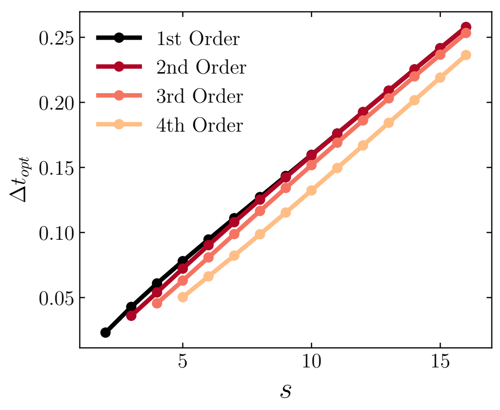


Figure 4.9. Concave hull of semidiscrete eigenvalues scaled by Δt_{opt}(red) shown within the region of absolute stability of their corresponding stability polynomials for triangular elements with $k = 4$

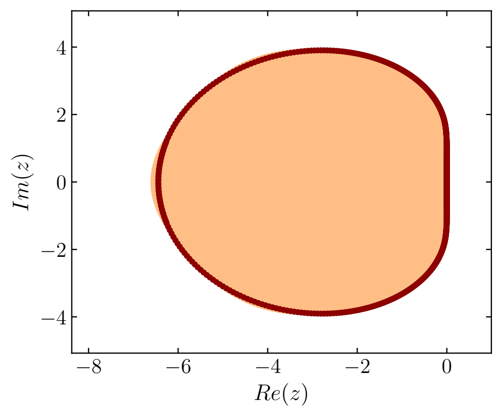


(a) $k = 0$

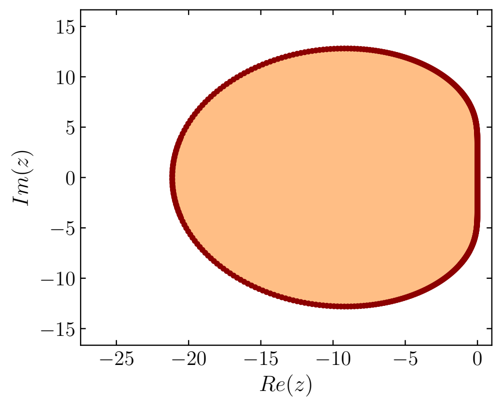


(b) $k = 6$

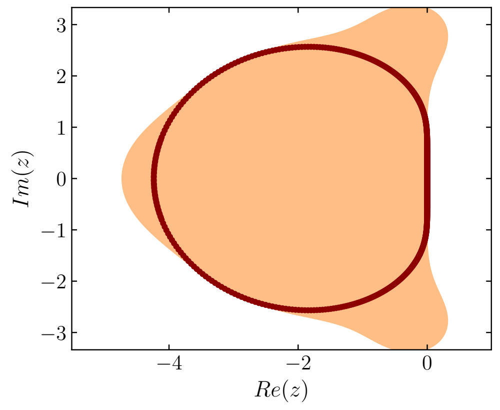
Figure 4.10. Plots of the Δt_{opt} as a function of the number of stages for hexahedral elements with $k = 0$ and $k = 6$



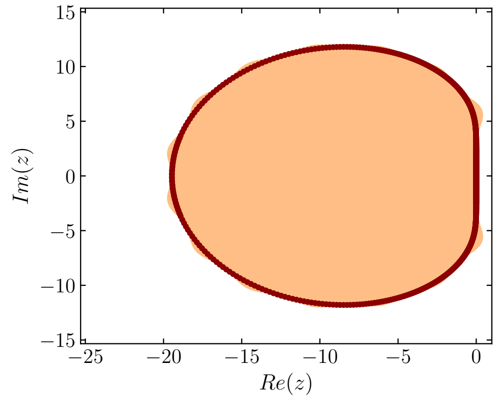
(a) $p = 1$ and $s = 5$



(b) $p = 1$ and $s = 16$

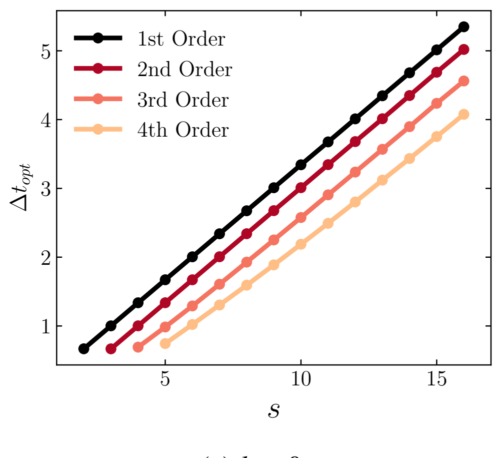


(c) $p = 4$ and $s = 5$

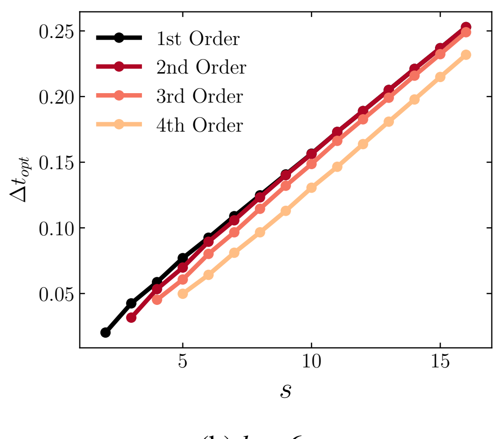


(d) $p = 4$ and $s = 16$

Figure 4.11. Concave hull of semidiscrete eigenvalues scaled by Δt_{opt}(red) shown within the region of absolute stability of their corresponding stability polynomials for hexahedral elements with $k = 4$

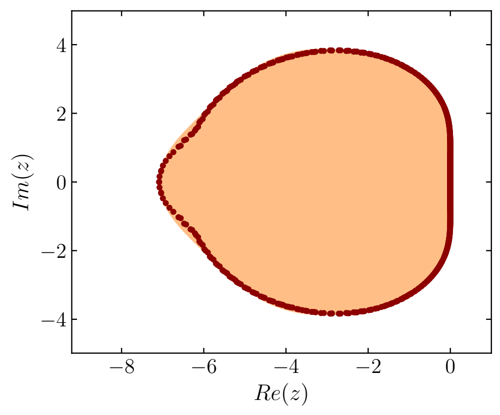


(a) $k = 0$

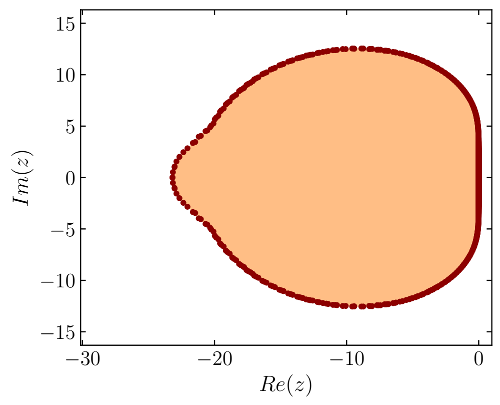


(b) $k = 6$

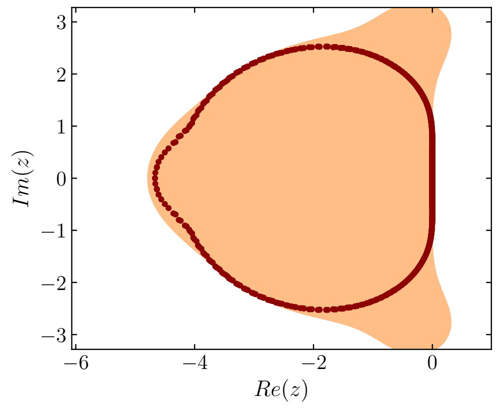
Figure 4.12. Plots of the Δt_{opt} as a function of the number of stages for prismatic elements with $k = 0$ and $k = 6$



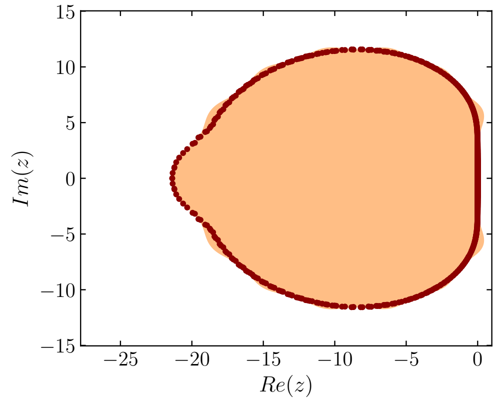
(a) $p = 1$ and $s = 5$



(b) $p = 1$ and $s = 16$

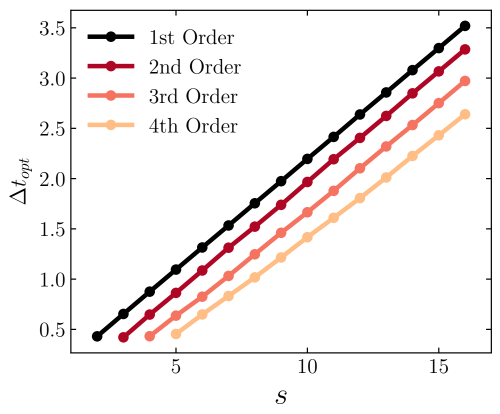


(c) $p = 4$ and $s = 5$

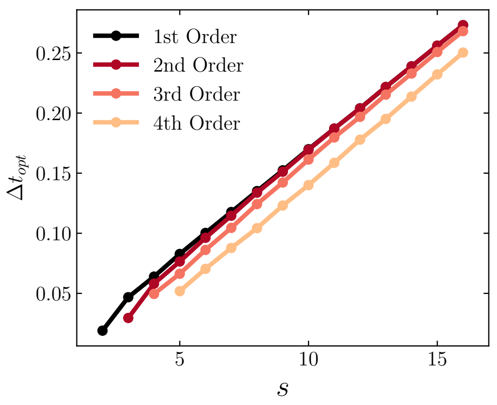


(d) $p = 4$ and $s = 16$

Figure 4.13. Concave hull of semidiscrete eigenvalues scaled by Δt_{opt}(red) shown within the region of absolute stability of their corresponding stability polynomials for prismatic elements with $k = 4$

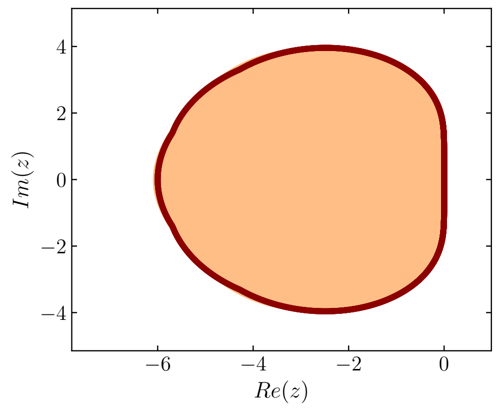


(a) $k = 0$

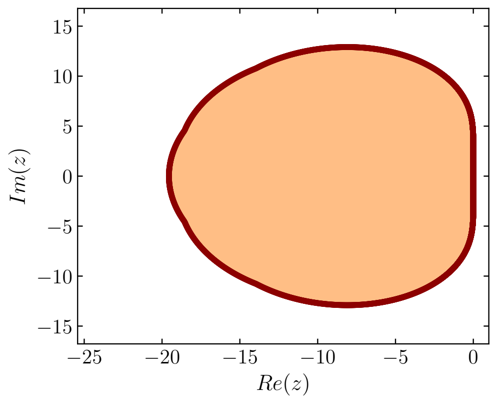


(b) $k = 6$

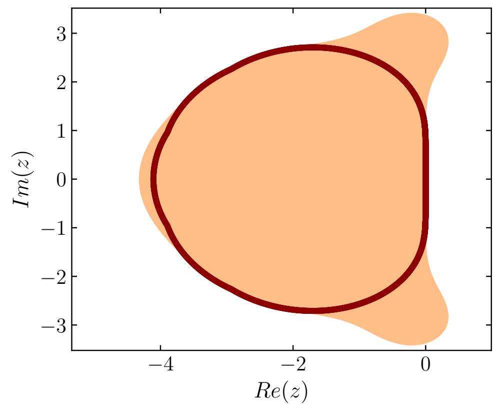
Figure 4.14. Plots of the Δt_{opt} as a function of the number of stages for tetrahedral elements with $k = 0$ and $k = 6$



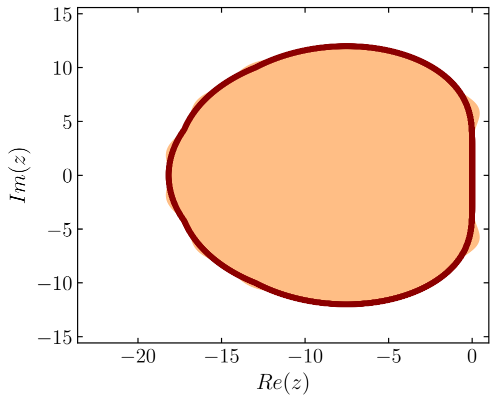
(a) $p = 1$ and $s = 5$



(b) $p = 1$ and $s = 16$

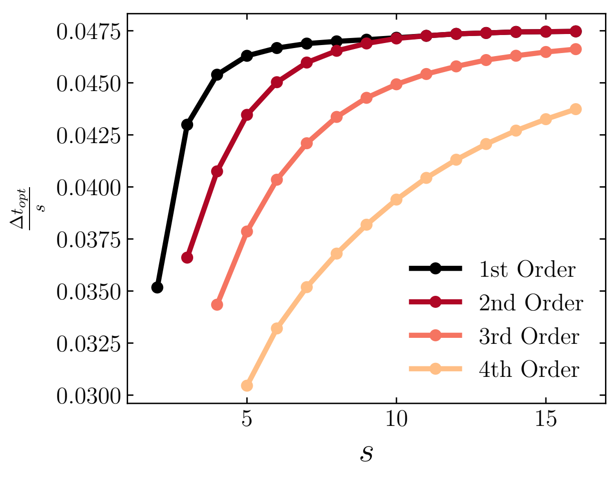


(c) $p = 4$ and $s = 5$

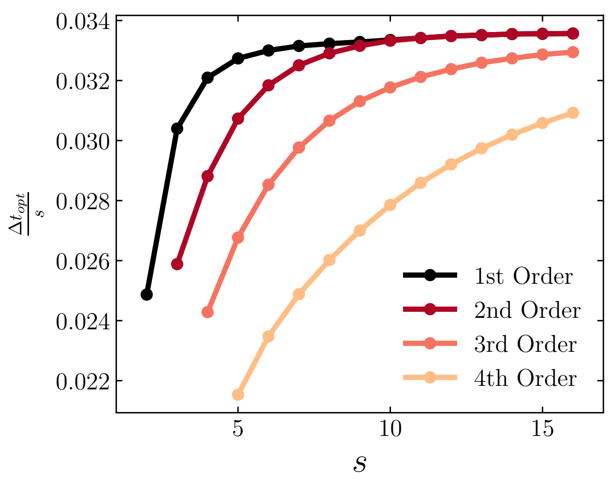


(d) $p = 4$ and $s = 16$

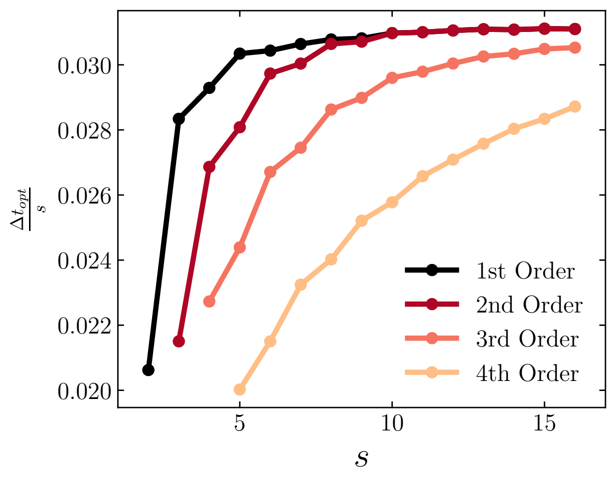
Figure 4.15. Concave hull of semidiscrete eigenvalues scaled by Δt_{opt}(red) shown within the region of absolute stability of their corresponding stability polynomials for tetrahedral elements with $k = 4$



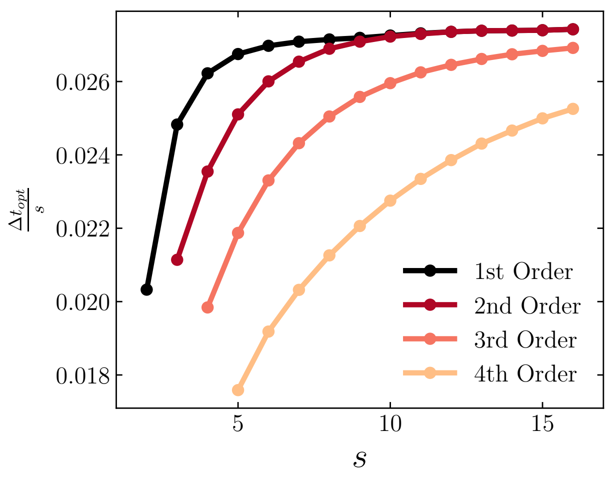
(a) Line elements



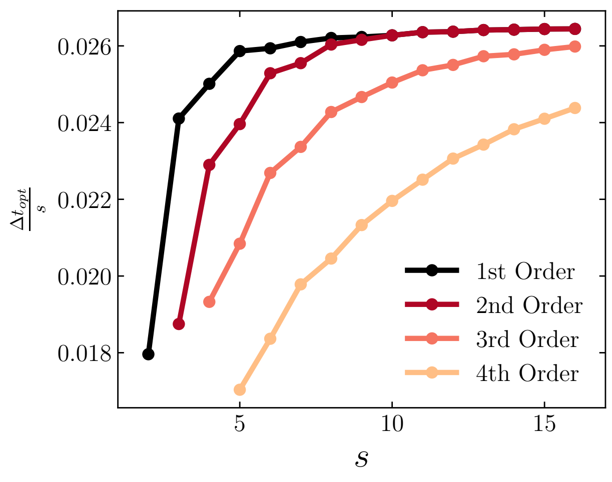
(b) Quadrilateral elements



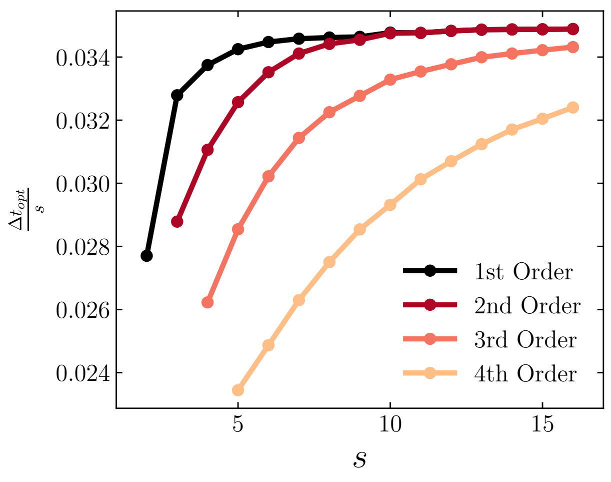
(c) Triangular elements



(d) Hexahedral elements



(e) Prismatic elements



(f) Tetrahedral elements

Figure 4.16. Plots of the $\frac{\Delta t_{opt}}{s}$ as a function of s for different types of elements ($k = 4$)

4.4.4 Comparison with Optimal One-Dimensional Schemes

In this section, we explore the utility of optimizing the stability polynomial for each multidimensional element type, rather than just using the scheme optimized for one-dimensional elements. Tables 4.1, 4.2, 4.3, 4.4 and 4.5 show the ratio of Δt_{opt} using the optimal stability polynomial of each multidimensional element type, relative to using the optimal one-dimensional stability polynomial with the multidimensional element. This is shown for all considered orders of accuracy in time and for all stage counts.

Table 4.1. Ratio of the maximum time step for $k = 4$ quadrilateral elements using their optimal stability polynomial, relative to the maximum time step obtained using a stability polynomial optimized for a one-dimensional element.

Number of Stages	1st order	2nd order	3rd order	4th order
s = 2	1.00			
s = 3	1.00	1.00		
s = 4	1.00	1.00	1.00	
s = 5	1.00	1.00	1.00	1.00
s = 6	1.00	1.00	1.00	1.00
s = 7	1.00	1.00	1.00	1.00
s = 8	1.00	1.00	1.00	1.00
s = 9	1.00	1.00	1.00	1.00
s = 10	1.00	1.00	1.00	1.00
s = 11	1.00	1.00	1.00	1.00
s = 12	1.00	1.00	1.00	1.00
s = 13	1.00	1.00	1.00	1.00
s = 14	1.00	1.00	1.00	1.00
s = 15	1.00	1.00	1.00	1.00
s = 16	1.00	1.00	1.00	1.00

Table 4.2. Ratio of the maximum time step for $k = 4$ triangular elements using their optimal stability polynomial, relative to the maximum time step obtained using a stability polynomial optimized for a one-dimensional element.

Number of Stages	1st order	2nd order	3rd order	4th order
s = 2	1.00			
s = 3	1.01	1.00		
s = 4	1.10	1.06	1.01	
s = 5	1.10	1.10	1.10	1.02
s = 6	1.11	1.10	1.05	1.11
s = 7	1.11	1.12	1.11	1.08
s = 8	1.12	1.10	1.08	1.12
s = 9	1.12	1.12	1.12	1.08
s = 10	1.12	1.12	1.09	1.12
s = 11	1.12	1.12	1.12	1.08
s = 12	1.12	1.12	1.10	1.12
s = 13	1.12	1.12	1.12	1.09
s = 14	1.12	1.12	1.11	1.12
s = 15	1.12	1.12	1.12	1.09
s = 16	1.12	1.12	1.11	1.12

Table 4.3. Ratio of the maximum time step for $k = 4$ hexahedral elements using their optimal stability polynomial, relative to the maximum time step using a stability polynomial optimized for a one-dimensional element.

Number of Stages	1st order	2nd order	3rd order	4th order
s = 2	1.00			
s = 3	1.00	1.00		
s = 4	1.00	1.00	1.00	
s = 5	1.00	1.00	1.00	1.00
s = 6	1.00	1.00	1.00	1.00
s = 7	1.00	1.00	1.00	1.00
s = 8	1.00	1.00	1.00	1.00
s = 9	1.00	1.00	1.00	1.00
s = 10	1.00	1.00	1.00	1.00
s = 11	1.00	1.00	1.00	1.00
s = 12	1.00	1.00	1.00	1.00
s = 13	1.00	1.00	1.00	1.00
s = 14	1.00	1.00	1.00	1.00
s = 15	1.00	1.00	1.00	1.00
s = 16	1.00	1.00	1.00	1.00

Table 4.4. Ratio of the maximum time step for $k = 4$ prismatic elements using their optimal stability polynomial, relative to the maximum time step obtained using a stability polynomial optimized for a one-dimensional element.

Number of Stages	1st order	2nd order	3rd order	4th order
s = 2	1.00			
s = 3	1.00	1.00		
s = 4	1.08	1.04	1.00	
s = 5	1.07	1.08	1.08	1.04
s = 6	1.09	1.06	1.04	1.08
s = 7	1.09	1.09	1.09	1.05
s = 8	1.09	1.08	1.05	1.09
s = 9	1.09	1.09	1.09	1.05
s = 10	1.09	1.08	1.06	1.09
s = 11	1.09	1.09	1.09	1.05
s = 12	1.09	1.09	1.07	1.09
s = 13	1.09	1.09	1.09	1.06
s = 14	1.09	1.09	1.08	1.09
s = 15	1.09	1.09	1.09	1.07
s = 16	1.09	1.09	1.08	1.09

From these results, it is clear that optimizing the stability polynomial for the quadrilateral and hexahedral elements provides no additional benefit relative to using the optimal one-dimensional stability polynomial. However, minor performance improvements can be obtained for triangular, tetrahedral, and prismatic elements with additional speedup factors of up to 1.12 observed.

Table 4.5. Ratio of the maximum time step for $k = 4$ tetrahedral elements using their optimal stability polynomial, relative to the maximum time step obtained using a stability polynomial optimized for a one-dimensional element.

Number of Stages	1st order	2nd order	3rd order	4th order
s = 2	1.09			
s = 3	1.06	1.00		
s = 4	1.04	1.02	1.02	
s = 5	1.04	1.03	1.02	1.00
s = 6	1.05	1.03	1.03	1.02
s = 7	1.05	1.05	1.05	1.03
s = 8	1.05	1.06	1.06	1.05
s = 9	1.06	1.06	1.06	1.06
s = 10	1.06	1.06	1.07	1.06
s = 11	1.06	1.06	1.07	1.07
s = 12	1.06	1.06	1.07	1.07
s = 13	1.07	1.07	1.07	1.07
s = 14	1.06	1.07	1.07	1.07
s = 15	1.06	1.06	1.06	1.07
s = 16	1.06	1.06	1.06	1.07

4.4.5 Comparison with Classical Runge-Kutta Schemes

To investigate the utility of the stability polynomials optimized for each multidimensional element type, we can compare them to the classical $RK_{3,3}$ and $RK_{4,4}$ methods. Table 4.6 shows the ratio of Δt_{opt} relative to the maximum stable time step size using $RK_{3,3}$. This shows that a larger time step relative to $RK_{3,3}$ can be taken for all elements as the number of stages increases, which is to be expected. More importantly, Table 4.7 shows that the speedup factor relative to $RK_{3,3}$ also increases with the number of stages, which is measured by the ratio of optimal schemes $\frac{\Delta t_{opt}}{s}$ relative to that of

$RK_{3,3}$. By 16 stages, speedup factors of 1.46 to 1.74 are observed, depending on the element type.

Table 4.6. Time step ratio of $p = 3$ optimal RK schemes for different element types with $k = 4$ relative to $RK_{3,3}$, $(\Delta t_{opt}/\Delta t_{RK_{3,3}})$

Number of Stages	Hexahedral	Prismatic	Tetrahedral	Quadrilateral	Triangular	Line element
s = 3	1.00	1.00	1.00	1.00	1.00	1.00
s = 4	1.53	1.69	1.48	1.53	1.73	1.53
s = 5	2.11	2.27	2.02	2.11	2.32	2.11
s = 6	2.70	2.97	2.57	2.70	3.05	2.70
s = 7	3.29	3.57	3.11	3.29	3.65	3.29
s = 8	3.87	4.24	3.65	3.87	4.36	3.87
s = 9	4.44	4.84	4.17	4.44	4.96	4.44
s = 10	5.01	5.46	4.71	5.01	5.63	5.01
s = 11	5.57	6.08	5.22	5.57	6.23	5.57
s = 12	6.13	6.67	5.73	6.13	6.86	6.13
s = 13	6.68	7.29	6.25	6.68	7.48	6.68
s = 14	7.23	7.87	6.76	7.23	8.08	7.23
s = 15	7.77	8.47	7.26	7.77	8.70	7.77
s = 16	8.31	9.07	7.77	8.31	9.29	8.32

Table 4.7. Speedup factor of $p = 3$ optimal RK schemes for different element types with $k = 4$ relative to $RK_{3,3}$, $(\frac{\Delta t_{opt}}{s} / \frac{\Delta t_{RK_{3,3}}}{3})$

Number of Stages	Hexahedral	Prismatic	Tetrahedral	Quadrilateral	Triangular	Line
s=3	1.00	1.00	1.00	1.00	1.00	1.00
s=4	1.15	1.26	1.11	1.15	1.30	1.15
s=5	1.27	1.36	1.21	1.27	1.39	1.27
s=6	1.35	1.48	1.28	1.35	1.52	1.35
s=7	1.41	1.53	1.33	1.41	1.57	1.41
s=8	1.45	1.59	1.37	1.45	1.63	1.45
s=9	1.48	1.61	1.39	1.48	1.65	1.48
s=10	1.50	1.64	1.41	1.50	1.69	1.50
s=11	1.52	1.66	1.42	1.52	1.70	1.52
s=12	1.53	1.67	1.43	1.53	1.71	1.53
s=13	1.54	1.68	1.44	1.54	1.73	1.54
s=14	1.55	1.69	1.45	1.55	1.73	1.55
s=15	1.55	1.69	1.45	1.55	1.74	1.55
s=16	1.56	1.70	1.46	1.56	1.74	1.56

Similar to Table 4.6, Table 4.8 shows the ratio of Δt_{opt} relative to the maximum stable time step size using $RK_{4,4}$. Again this shows that a larger time step relative to $RK_{4,4}$ can be taken for all elements as the number of stages increases. In addition, similar to Table 4.7, Table 4.9 demonstrates that the speedup factor relative to classical $RK_{4,4}$ also increases with the number of stages, which is again measured by the ratio of the optimal schemes $\frac{\Delta t_{max}}{s}$ relative to that of $RK_{4,4}$. By 16 stages, speedup factors of 1.63 to 1.97 are observed, depending on the element type.

Table 4.8. Time step ratio of $p = 4$ optimal RK schemes for different element types with $k = 4$ relative to $RK_{4,4}$, $(\Delta t_{opt}/\Delta t_{RK_{4,4}})$

Number of Stages	Hexahedral	Prismatic	Tetrahedral	Quadrilateral	Triangular	Line element
s=4	1.00	1.00	1.00	1.00	1.00	1.00
s=5	1.52	1.68	1.47	1.52	1.72	1.52
s=6	1.99	2.17	1.88	1.99	2.21	1.99
s=7	2.46	2.72	2.32	2.46	2.79	2.46
s=8	2.94	3.22	2.77	2.94	3.30	2.94
s=9	3.43	3.78	3.23	3.44	3.89	3.44
s=10	3.93	4.32	3.69	3.94	4.42	3.94
s=11	4.44	4.87	4.17	4.45	5.02	4.45
s=12	4.95	5.44	4.63	4.96	5.58	4.95
s=13	5.47	5.99	5.11	5.47	6.15	5.47
s=14	5.97	6.56	5.58	5.98	6.73	5.98
s=15	6.48	7.11	6.05	6.49	7.29	6.49
s=16	6.99	7.67	6.52	7.00	7.88	7.00

Table 4.9. Speedup factor of $p = 4$ optimal RK schemes for different element types with $k = 4$ relative to $RK_{4,4}$, $(\frac{\Delta t_{opt}}{s} / \frac{\Delta t_{RK_{4,4}}}{4})$

Number of Stages	Hexahedral	Prismatic	Tetrahedral	Quadrilateral	Triangular	Line
s=4	1.00	1.00	1.00	1.00	1.00	1.00
s=5	1.22	1.34	1.18	1.22	1.37	1.22
s=6	1.33	1.45	1.25	1.33	1.48	1.33
s=7	1.41	1.56	1.32	1.41	1.60	1.41
s=8	1.47	1.61	1.38	1.47	1.65	1.47
s=9	1.53	1.68	1.44	1.53	1.73	1.53
s=10	1.57	1.73	1.48	1.58	1.77	1.58
s=11	1.61	1.77	1.52	1.62	1.82	1.62
s=12	1.65	1.81	1.54	1.65	1.86	1.65
s=13	1.68	1.84	1.57	1.68	1.89	1.68
s=14	1.71	1.87	1.60	1.71	1.92	1.71
s=15	1.73	1.90	1.61	1.73	1.95	1.73
s=16	1.75	1.92	1.63	1.75	1.97	1.75

4.5 Numerical Results

4.5.1 Verification

For verification, a Butcher tableau was generated for each stability polynomial using the least truncation method [59] (all corresponding Butcher tableaus can be found in the supplementary materials), and was used to solve a linear advection test case. To amplify the temporal error beyond machine precision, a prescribed source term was added using the method of manufactured solutions.

The linear advection equation with this added source term is

$$\frac{\partial u}{\partial t} + \boldsymbol{\alpha}\frac{\partial u}{\partial \boldsymbol{x}} = S(t), \tag{4.8}$$

where $u(\boldsymbol{x}, t)$ is a scalar, $\boldsymbol{\alpha}$ is the advection velocity, and $S(t)$ is the source term. Assuming periodic boundaries and an initial condition $u(\boldsymbol{x}, 0) = e^{-\kappa|\boldsymbol{x}|^2}$, where $\kappa = 0.4$. The exact solution is

$$u_{exact} = e^{\kappa|\boldsymbol{x}-\boldsymbol{\alpha}t|^2} + \int S(t)dt. \tag{4.9}$$

Taking the exact solution to be

$$u(\boldsymbol{x}, t) = e^{\kappa|\boldsymbol{x}-\boldsymbol{\alpha}t|^2} + 100\sin(10\pi t), \tag{4.10}$$

the resulting source term is

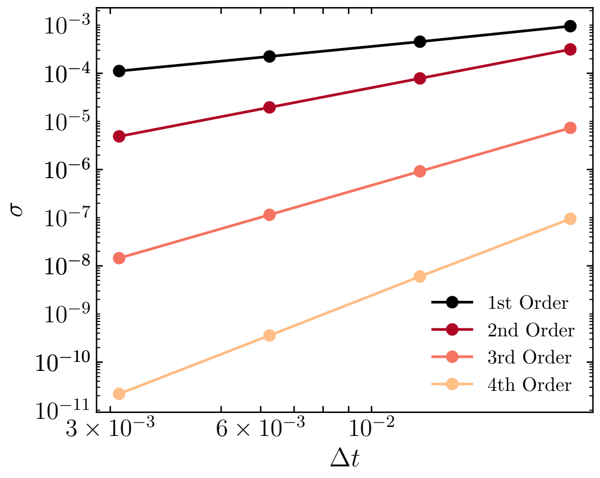
$$S(t) = 1000\pi \times \cos(10\pi t). \tag{4.11}$$

We used a 20×20 two-dimensional and $20 \times 20 \times 20$ three-dimensional domain Ω, with periodic boundary conditions in all directions. Two sets of two-dimensional simulations, one with 80×80 quadrilateral elements, and another with 160×160 triangular elements and, three sets of three-dimensional simulations, one with $80\times80\times80$ hexahedral elements, and two sets with $160\times160\times160$ tetrahedral and prismatic elements were run to evaluate order of accuracy of optimal schemes. The solution was represented using $k = 6$ degree polynomial on each element to minimize spatial error. A set of simulations was run using different optimal stability polynomials with $s = 16$ and $q = 1, 2, 3, 4$. Each simulation was run with unit advection velocity in all directions to a final simulation time of $t = 20$ to allow the flow to complete a full cycle through the periodic domain.

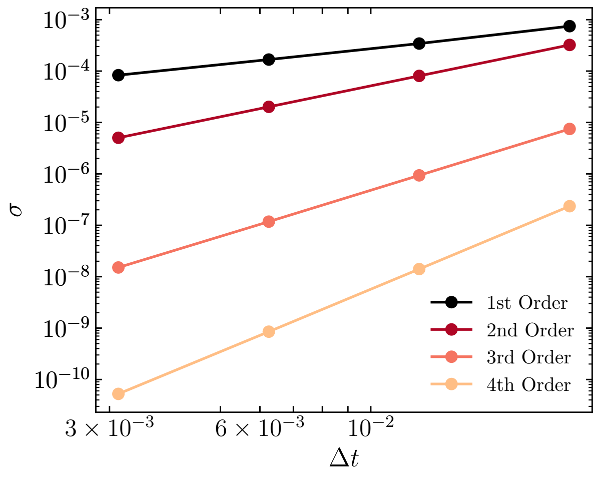
To evaluate the accuracy of each simulation we considered the L_2 norm of the error at the end of each simulation, defined as

$$L_2 = \sqrt{\int_\Omega (u_n(\boldsymbol{x}) - u_{exact}(\boldsymbol{x}))^2 d\Omega}. \tag{4.12}$$

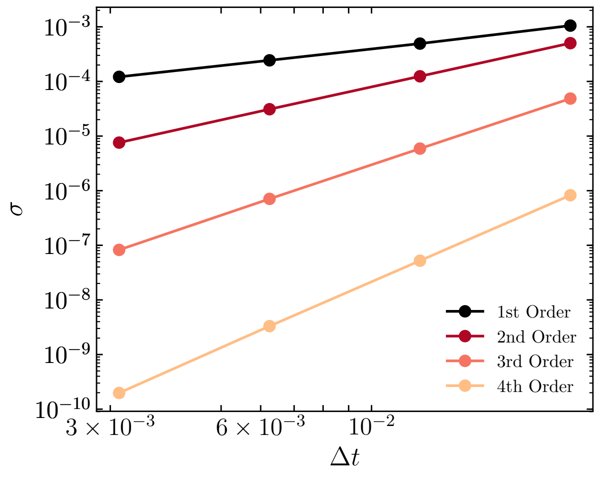
A summary of the error for each temporal scheme and time step size including observed orders of accuracy are plotted in Figure 4.17 for all element types. Importantly, we observe that all schemes achieve their designed order of accuracy.



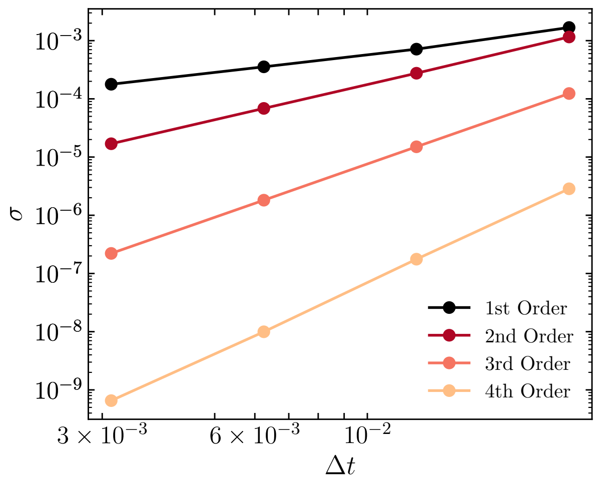
(a) Quadrilateral element



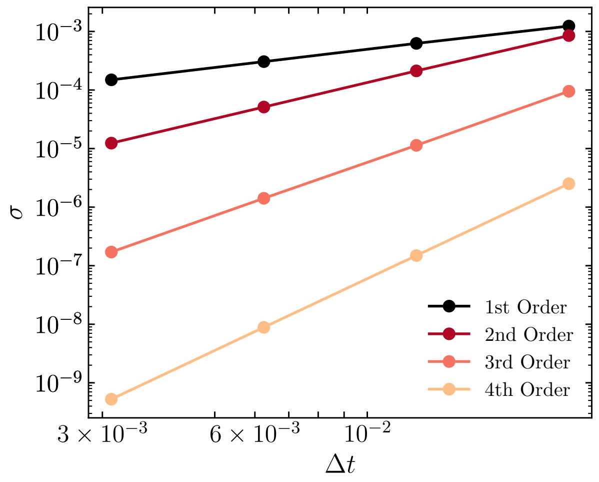
(b) Triangular element



(c) Hexahedral element



(d) Tetrahedral element



(e) Prismatic element

Figure 4.17. Convergence plots for linear advection using the method of manufactured solutions

4.5.2 Navier-Stokes Equations

To explore the utility of the optimized stability polynomials for multidimensional non-linear problems, we consider Direct Numerical Simulation (DNS) of the Taylor-Green vortex. The initial flow field for this test case is specified as [87]

$$
\begin{aligned}
u_x &= +U_0 \sin(x/L)\cos(y/L)\cos(z/L),\\
u_y &= -U_0 \cos(x/L)\sin(y/L)\cos(z/L),\\
u_z &= 0,\\
P_0 &= \frac{\rho_o U_0^2}{16}\left(\cos(2x/L)+\cos(2y/L)\right)\left(\cos(2z/L)+2\right),\\
\rho &= \frac{\mathcal{P}}{RT_0},
\end{aligned}
\tag{4.13}
$$

where, u, v, and w are the velocity components, $\mathcal{P}$ is the pressure, $\mathcal{P}_0$ is the background pressure, ρ is the density, and T_0 and U_0 are constants specified such that the flow Mach number based on U_0 is $Ma = 0.1$, effectively incompressible. The domain is a periodic cube with the dimensions $-\pi L \le x, y, z \le +\pi L$. For the current study we consider a Reynolds number $Re = 1600$ based on the length scale L and velocity scale U_0. The test case is run to a final non-dimensional time of $t = 20t_c$ where $t_c = L/U_0$, and reference spectral DNS data is available from van Rees et al. [88]. Of primary interest is the temporal evolution of the total kinetic energy, which can be found via and, more specifically, its dissipation rate $\epsilon = -dE_k/dt$. Furthermore, the temporal evolution of enstrophy can be computed via

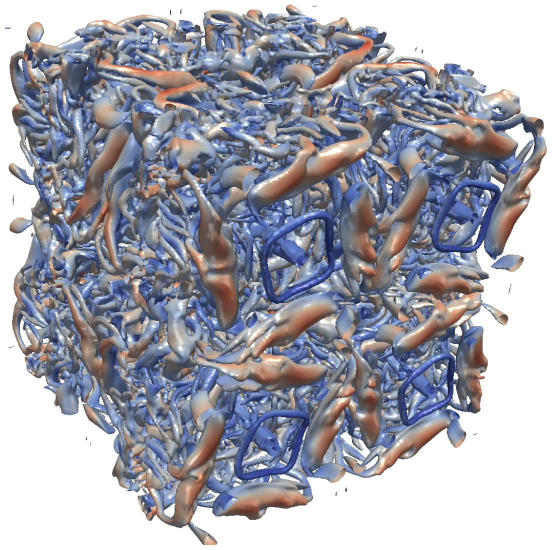
$$\varepsilon = \frac{1}{\rho_0 \Omega}\int_\Omega \rho \frac{\boldsymbol{\omega}\cdot\boldsymbol{\omega}}{2} d\Omega, \tag{4.14}$$

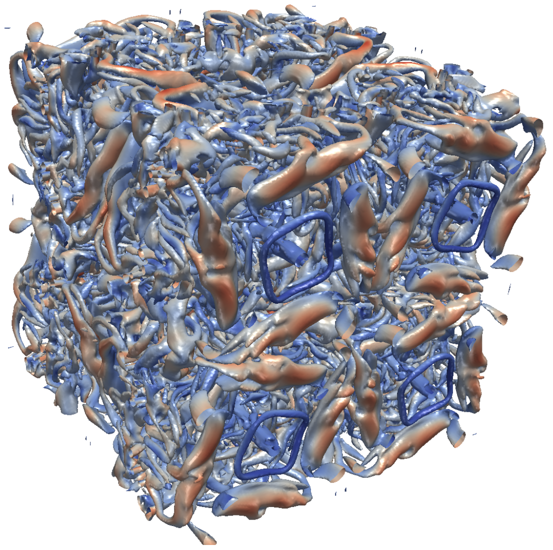
where $\boldsymbol{\omega}$ is the vorticity. For incompressible flows the dissipation rate can be related to the enstrophy by $\epsilon = 2\frac{\mu}{\rho_o}\varepsilon$ [87, 88]. Hence, the enstropy is a direct measure of the expected physical dissipation, and the dissipation rate computed from the kinetic energy is typically higher due to the additional numerical dissipation of the scheme.

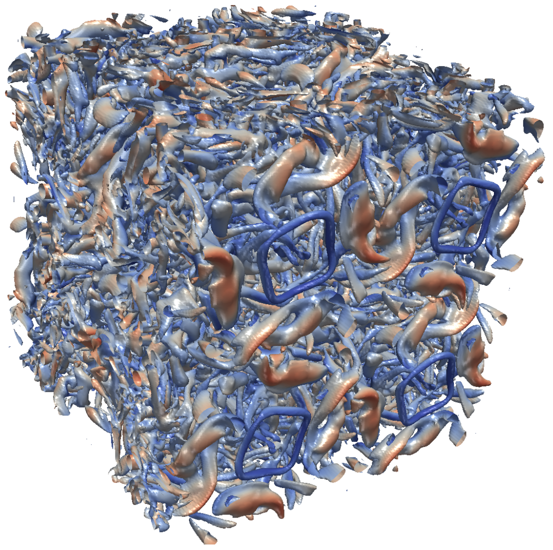
A total of six simulations were run using $k = 5$ and a nominal 256^3 solution points using hexahedral, prismatic and tetrahedral elements with the classical $\mathrm{RK}_{4,4}$ scheme and each element types corresponding optimal fourth-order temporal scheme with $p = 4$ and $s = 16$. The maximum stable time step size for each element type and temporal scheme were determined via bisection, and each simulation was run to completion using a time step 80% of this maximum size. This resulted in speedup factors of 1.68, 1.64, and 1.57 for hexahedral, prismatic and tetrahedral elements, respectively when using the optimal stability polynomials. Importantly, these speedup factors were obtained with negligible code modification, simply the substitution of the coefficients in the Butcher tableau. Qualitative results in terms of isosurfaces of Q-criterion colored by velocity magnitude are shown in Figure 4.18 for all element types and both sets of temporal schemes at a non-dimensional simulation time of $t_c = 15$, which is beyond the turbulent transition time. From these images, it is apparent that the flow is qualitatively indistinguishable between the two temporal schemes for all element types. Furthermore, quantitative results in terms of the total kinetic energy dissipation rate and enstrophy are shown in Figure 4.19 and Figure 4.20, respectively. These results demonstrate that there is negligible difference in terms of accuracy for both quantitative measures throughout each simulation, where the plots for the classical and optimized Runge-Kutta methods are visually indistinguishable. Hence, we observe that the optimized stability polynomials are able to yield significant reductions in simulation time with negligible influence on both qualitative and quantitative results.

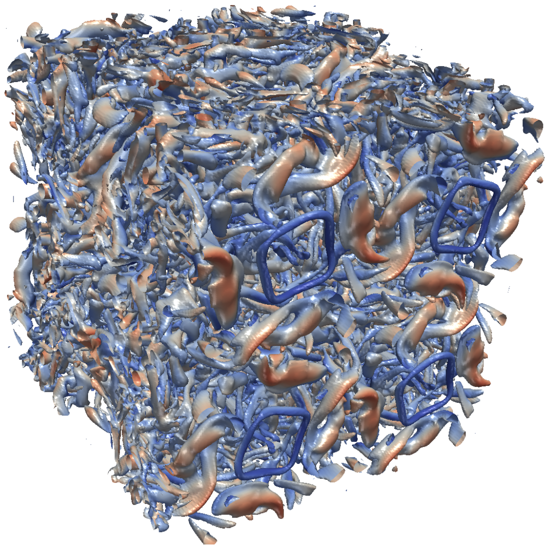
4.6 Discussion

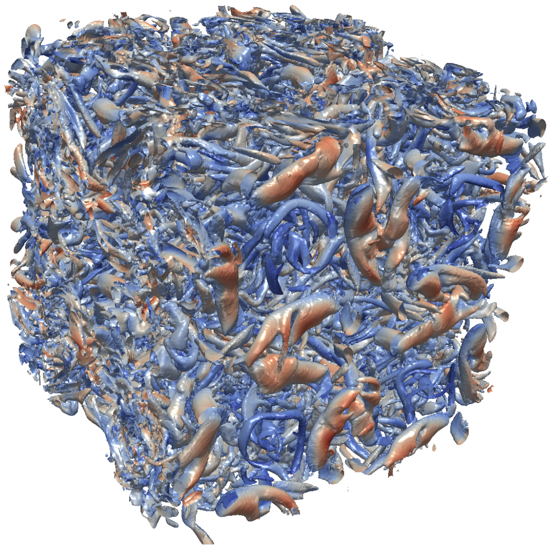
In this chapter, we have generated optimal stability polynomials for high-order discontinuous Galerkin schemes, recovered using the flux reconstruction approach. It has been shown that, by adding more Runge-Kutta stages, and terms to the corresponding stability polynomial, larger optimized time steps can be obtained. Furthermore, the relative efficiency $\frac{\Delta t_{opt}}{s}$ increases with the number of stages for all element types. In the case of quadrilateral and hexahedral elements, there was no added benefit relative to schemes optimized with multidimensional eigenspectra, when compared to a stability polynomials generated for line elements. However, for triangular, prismatic, and tetrahedral elements, there is a modest performance improvement. Compared to classical Runge-Kutta methods, the optimal stability polynomials yield speedup factors up to 1.97. Results for linear advection demonstrated that these schemes achieve their designed order of accuracy. Results from a Taylor-Green vortex demonstrated that the optimal schemes allow significant reductions in computational cost, with negligible impact on accuracy. Hence, Runge-Kutta stability polynomials optimized for multidimensional element types are a viable solution for accelerating solutions using high-order methods, including the discontinuous Galerkin method recovered via the flux reconstruction approach.


(a) Hexahedral elements with $\mathrm{RK}_{4,4}$


(b) Hexahedral elements with $p = 4$ and $s = 16$


(c) Prismatic elements with $\mathrm{RK}_{4,4}$


(d) Prismatic elements with $p = 4$ and $s = 16$


(e) Tetrahedral elements with $\mathrm{RK}_{4,4}$

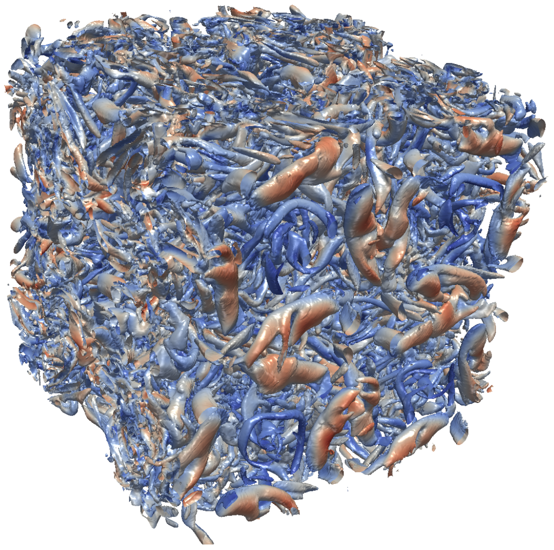
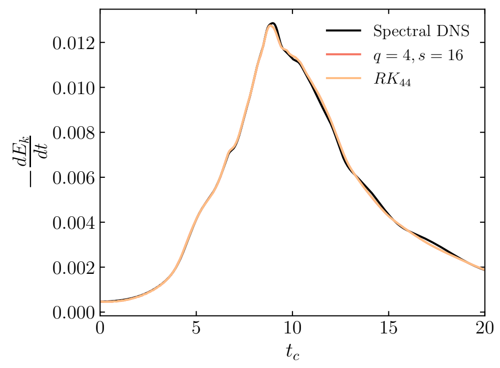
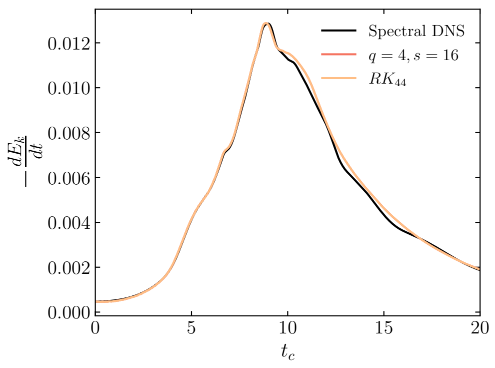

(f) Tetrahedral elements with $p = 4$ and $s = 16$

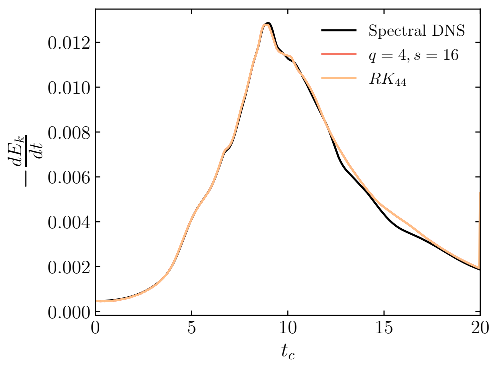
Figure 4.18. Isosurfaces of Q-criterion coloured by velocity magnitude for the Taylor-Green vortex.



(a) Hexahedral element

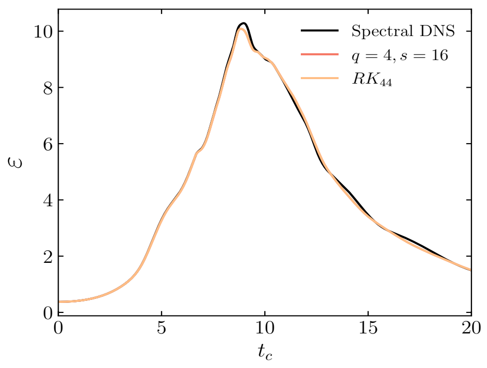


(b) Tetrahedral element

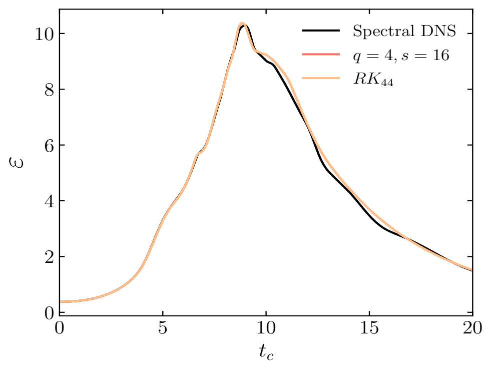


(c) Prismatic element

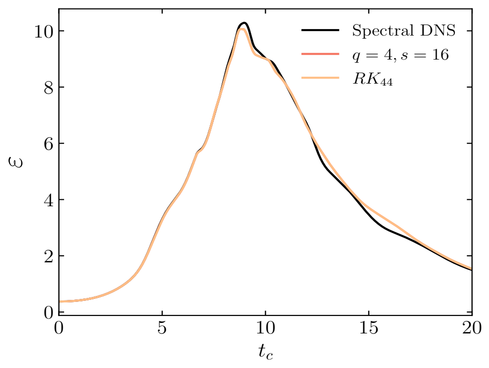
Figure 4.19. Energy decay rate versus dimensionless time



(a) Hexahedral element



(b) Tetrahedral element



(c) Prismatic element

Figure 4.20. Enstrophy versus dimensionless time

Chapter 5

Third-Order Paired Explicit Runge-Kutta Schemes

5.1 Authorship Statement

The mesh for tandem spheres case was provided via the high-order workshop (HiOCFD5) by Samuel James from GridPro. Apart from that, all of the contributions in this section were produced by Siavash Hedayati Nasab.

5.2 Background

Explicit methods are widely used for the solution of non-stiff systems of equations. However, a wide range of physical applications require the solution of multi-scale locally stiff systems, such as the Navier-Stokes equations, convection-diffusion equation, shallow water equations, and the Euler equations. Numerical stiffness requires prohibitively small time-steps to maintain stability for these types of locally stiff systems, even though this stiffness may arise in only a limited region of the computational domain, such as boundary layers or reaction zones [39, 70, 89]. Implicit schemes are often used to overcome these limitations. However, implicit approaches are often more expensive in terms of computational cost per time-step and memory requirements. Classical

Implicit-Explicit (IMEX) Runge-Kutta methods [54, 65, 66, 76, 77], and particularly the recently developed Accelerated-IMEX schemes [67], have demonstrated particular promise in the simulation of such systems, taking advantage of an implicit method for the stiff parts and an explicit method for the non-stiff parts of a system, respectively. Nevertheless, they are relatively complex to implement, and solution of the implicit region can still be expensive. Hence, although explicit methods have the disadvantage of being conditionally stable, they are often preferred due to their simplicity and low cost per time-step [38]. Therefore, ongoing research has been dedicated to alleviating the stability constraints of explicit methods when applied to stiff systems of equations. The primary focus of these studies has been optimizing a scheme's region of absolute stability, a property of the schemes' stability polynomial, enabling the largest possible time step size per stage [55]. For example, we have seen the application of optimization of Runge-Kutta stability polynomials in the works of Van der Houwen et al. [71], Ruuth et al. [72, 73], Ketcheson et al. [59], Parsani et al. [74] Kubatko et al. [55], and Vermeire et al. [75]. In addition, efforts have been made to optimize stability polynomials for multidimensional element types for high-order unstructured methods [74, 90].

Another approach for alleviating the stability of explicit schemes is multirate time-stepping methods. These methods allow different schemes with different time step sizes to be used in different regions of the domain. For example, Constantinescu and Sandu introduced two second-order multirate methods [91], Seny et al. developed a multirate method [92], and Schlegel et al. introduced a recursive multirate scheme for advection equations [93].

Recently, a new method for locally-stiff systems was proposed, referred to as Paired Explicit Runge-Kutta schemes [84]. The P-ERK formulation consists of a family of schemes whose stability polynomials are optimized for a given spatial discretization. With P-ERK different schemes with different stability properties are used in different regions of the domain to increase the global time-step size and accelerate the solution of the locally stiff systems. Members of a P-ERK family have arbitrarily large numbers of stages, but each can possess a different number of active stages. Only active stages require a right hand side derivative evaluation, and the number of active stages is determined based on local stiffness. Hence, P-ERK schemes require significantly fewer derivative evaluations in non-stiff parts of the domain, while retaining the stability benefits of a high-stage count scheme in stiff region [84]. In this study, we expand P-ERK schemes, originally formulated with second-order accuracy, to third-order accuracy. We then demonstrate the utility of third-order P-ERK schemes when paired with Flux Reconstruction (FR) spatial discretization applied to the solution of locally-stiff problems with the Navier-Stokes and Euler equations [94].

5.3 Third-Order Paired Explicit Runge-Kutta Schemes

5.3.1 Formulation of Third-Order Paired Explicit Runge-Kutta Schemes

In this section, we introduce a formulation for third-order accurate P-ERK schemes. This formulation is developed for an arbitrarily large number of stages, such that multiple families of schemes can be defined. The general form of a family of third-order P-ERK scheme possesses s stages ($s > 3$) and e active stages ($3 \leq e \leq s$) for each member of the family, denoted by P-ERK$_{s,e,3}$. Members with relatively high values of e are employed in stiff regions to enhance stability, and members with relatively small values of e are used in non-stiff regions to minimize computational cost. The free coefficients in the stability polynomial of each member ($e - 3$ coefficients) can then be optimized for a given spatial discretization to improve their region of absolute stability.

The Butcher tableau for a family of P-ERK$_{s,e,3}$ schemes is created from a general explicit RK scheme with general explicit constraints given in Section 3.2.2.

$$a_{ij} = 0, \quad j \geq i, \tag{5.1}$$

Since, $a_{11} = 0$, it follows that $c_1 = 0$. This gives the following Butcher tableau

$$
\begin{array}{c|c}
\mathbf{c} & \mathbf{A} \\
\hline
 & \mathbf{b}
\end{array}
=
\begin{array}{c|cccccc}
0 & 0 & & & & & \\
c_2 & a_{2,1} & 0 & & & & \\
c_3 & a_{3,1} & a_{3,2} & 0 & & & \\
c_4 & a_{4,1} & a_{4,2} & a_{4,3} & \ddots & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\
c_s & a_{s,1} & a_{s,2} & a_{s,3} & \dots & a_{s,(s-1)} & 0 \\
\hline
 & b_1 & b_2 & b_3 & \dots & b_{s-1} & b_s
\end{array}
\tag{5.2}
$$

This general form of an explicit schemes requires evaluation and storage of s stage derivatives. However, we can significantly reduce memory consumption by requiring

$$
a_{ij} = 0, \qquad \forall \quad j > 1, \qquad i > j + 1, \tag{5.3}
$$

and

$$
b_i = 0, \qquad \forall \quad i < (s - 1). \tag{5.4}
$$

This leads to the following Butcher tableau

$$
\begin{array}{c|c}
\mathbf{c} & \mathbf{A} \\
\hline
 & \mathbf{b}
\end{array}
=
\begin{array}{c|cccccc}
0 & 0 & & & & & \\
c_2 & a_{2,1} & 0 & & & & \\
c_3 & a_{3,1} & a_{3,2} & 0 & & & \\
c_4 & a_{4,1} & 0 & a_{4,3} & \ddots & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\
c_s & a_{s,1} & 0 & 0 & \dots & a_{s,(s-1)} & 0 \\
\hline
 & 0 & 0 & 0 & \dots & b_{s-1} & b_s
\end{array}
\tag{5.5}
$$

This tableau requires only three stage derivatives be stored simultaneously, regardless of the total number of stages, significantly reduces memory requirements.

To reduce the number of active stages to e, we require

$$a_{i,(i-1)} = 0, \qquad \forall \qquad 2 \leq i \leq s-e+1, \tag{5.6}$$

which yields to following Butcher tableau

$$\begin{array}{c|c}\mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}\end{array} = \begin{array}{c|cccccccc}
0 & 0 & & & & & & & \\
c_2 & a_{2,1} & 0 & & & & & & \\
c_3 & a_{3,1} & 0 & 0 & & & & & \\
c_4 & a_{4,1} & 0 & 0 & \ddots & & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & 0 & & & \\
c_{(s-e+1)} & a_{(s-e+1),1} & 0 & 0 & \dots & a_{(s-e+1),(s-e)} & 0 & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\
c_s & a_{s,1} & 0 & 0 & 0 & \dots & \dots & a_{s,(s-1)} & 0 \\
\hline
& 0 & 0 & 0 & 0 & \dots \quad \dots & \dots & b_{s-1} & b_s
\end{array} \tag{5.7}$$

This can be further simplified by recalling that

$$c_i = \sum_{j=1}^{s} a_{ij}, \tag{5.8}$$

which leads to

$$
\begin{array}{c|c}
\mathbf{c} & \mathbf{A} \\
\hline
 & \mathbf{b}
\end{array}
=
\begin{array}{c|cccccccc}
0 & 0 & & & & & & & \\
c_2 & c_2 & 0 & & & & & & \\
c_3 & c_3 & 0 & 0 & & & & & \\
c_4 & c_4 & 0 & 0 & \ddots & & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & 0 & & & \\
c_{(s-e+1)} & c_{(s-e+1)} - a_{(s-e+1),(s-e)} & 0 & 0 & \dots & a_{(s-e+1),(s-e)} & 0 & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\
c_s & c_s - a_{s,(s-1)} & 0 & 0 & 0 & \dots & \dots & a_{s,(s-1)} & 0 \\
\hline
 & 0 & 0 & 0 & 0 & \dots & \dots & \dots & b_{s-1} \quad b_s
\end{array}
\tag{5.9}
$$

Furthermore, we need to satisfy the order conditions for a third-order accuracy given in Equations 3.24, 3.28, 3.29, and 3.30 [38]. These four conditions yield to a system of four non-linear equations to determine the unknowns in Equation 5.9. However, after expanding these conditions, it is apparent that only c_s, $c_{(s-1)}$, $c_{(s-2)}$, $a_{s,(s-1)}$, $a_{(s-1),(s-2)}$, b_s and b_{s-1} will be constrained, since we have only two non-zero components in $\mathbf{b}$. Moreover, Equations 3.24, 3.28 and 3.29 involve only four unknowns, b_s, b_{s-1}, c_s and c_{s-1}. Hence, if we assume one of these unknowns, we can find the three other. In this study, We take $c_s = 1$, and subsequently obtained $b_{(s-1)} = \frac{3}{4}$, $b_s = \frac{1}{4}$ and $c_{(s-1)} = \frac{1}{3}$.
Finally, using the last order condition Equation 3.30, we obtain a relationship between $a_{s,(s-1)}$,$a_{(s-1),(s-2)}$ and $c_{(s-2)}$ such that

$$
\sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} c_j = \frac{1}{6} \quad \Rightarrow \quad b_s a_{s,(s-1)} c_{(s-1)} + b_{(s-1)} a_{(s-1),(s-2)} c_{(s-2)} = \frac{1}{6}, \tag{5.10}
$$

substituting the values of b_s, b_{s-1}, and c_{s-1} leads to

$$
\frac{1}{12} a_{s,(s-1)} + \frac{3}{4} a_{(s-1),(s-2)} c_{(s-2)} = \frac{1}{6}, \tag{5.11}
$$

which yields to

$$c_{(s-2)} = \frac{2 - a_{s,(s-1)}}{9a_{(s-1),(s-2)}}, \tag{5.12}$$

finally, by using the Equation 5.8 we obtain a general P-ERK$_{s,e,3}$ Butcher tableau of the form

$$\begin{array}{c|cccccccccc}
0 & 0 & & & & & & & & & \\
c_2 & c_2 & 0 & & & & & & & & \\
c_3 & c_3 & 0 & 0 & & & & & & & \\
c_4 & c_4 & 0 & 0 & \ddots & & & & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & 0 & & & & & \\
c_{(s-e+1)} & c_{(s-e+1)} - a_{(s-e+1),(s-e)} & 0 & 0 & \dots & a_{(s-e+1),(s-e)} & 0 & & & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & & & \\
c_{(s-2)} & c_{(s-2)} - a_{(s-2),(s-3)} & 0 & 0 & 0 & \dots & \dots & a_{(s-2),(s-3)} & 0 & & \\
\frac{1}{3} & \frac{1}{3} - a_{(s-1),(s-2)} & 0 & 0 & 0 & \dots & \dots & 0 & a_{(s-1),(s-2)} & 0 & \\
1 & 1 - a_{s,(s-1)} & 0 & 0 & 0 & \dots & \dots & 0 & 0 & a_{s,(s-1)} & 0 \\
\hline
 & & 0 & 0 & 0 & 0 & \dots & \dots & \dots & 0 & \frac{3}{4} \quad \frac{1}{4}
\end{array} . \tag{5.13}$$

The stability polynomial associated with this scheme is of the form:

$$\begin{aligned}
P_{e,3} = 1 + z + \frac{1}{2}z^2 + \left(\frac{3}{4}a_{(s-1),(s-2)}c_{(s-2)} + \frac{1}{12}a_{s,(s-1)}\right)z^3 + \Bigg(\frac{3}{4}(a_{(s-1),(s-2)}a_{(s-2),(s-3)}c_{(s-3)}) \\
+ \frac{1}{4}(a_{s,(s-1)}a_{(s-1),(s-2)}c_{(s-2)})\Bigg)z^4 + \Bigg(\frac{3}{4}(a_{(s-1),(s-2)}a_{(s-2),(s-3)}a_{(s-3),(s-4)}c_{(s-4)}) \\
+ \frac{1}{4}(a_{s,(s-1)}a_{(s-1),(s-2)}a_{(s-2),(s-3)}c_{(s-3)})\Bigg)z^5 + \left(\frac{1}{4}(a_{s,(s-1)}a_{(s-1),(s-2)}a_{(s-2),(s-3)}a_{(s-3),(s-4)}c_{(s-4)})\right)z^6 + ...
\end{aligned} \tag{5.14}$$

The fourth term in this polynomial must be equal to $\frac{1}{6}$, which is ensured by Equation 5.12.

5.3.2 Stability Polynomial Optimization

In the previous section, we introduced the third-order P-ERK$_{s,e,3}$ family of schemes that have $e - 3$ remaining unknown terms in their stability polynomials. As discussed in Chapter 3, when there are more RK stages than required to achieve a desired order of accuracy in time, higher degree terms in the stability polynomial can be optimized for a given spatial discretization to improve other desirable properties. In other words, in the monomial basis representation of $P_{e,p}$ of a P-ERK scheme, the first $p+1$ coefficients $(\gamma_0, \gamma_1, ..., \gamma_p)$ are simply taken to satisfy the order conditions [59], and the remaining coefficients $(\gamma_{p+1}, \gamma_{p+2}, ..., \gamma_e)$, can be optimized for a chosen spatial discretization. In general, we optimize these unknown coefficients to increase the maximum stable time-step size, Δt_{max} for a given number of active stages. For a member with e active stages, the stability polynomial $P_{e,p}$ can be defined as [10, 59, 79, 83]

$$P_{e,3} = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \sum_{j=3}^{e} \gamma_j z^j, \tag{5.15}$$

which has e-3 free parameters $(\gamma_4, \gamma_5, \ldots, \gamma_e)$.

Using 1, we can optimize each family member of P-ERK$_{16,e,3}$ and Von Neumann analysis of flux reconstruction approach described in Section 4.3 for FR schemes. Similarly, to solve the convex minimization problem in Equation 3.42 we used CVX [86], which is a Matlab-based package [85] for specifying and solving convex problems. Following this approach, we can generate an optimal stability polynomial for a given system of equations.

Figure 6.2 displays the collection of eigenvalues for all considered polynomial degrees for resolvable wavenumbers. In addition, third-order Δt_{opt} as well as $\Delta t_{opt}/e$ versus number of active stages (from $e = 3$ through $e = 16$) for different degrees of solution polynomials are plotted in Figure 5.2a and Figure 5.2b, respectively. An approximately linear increase in time-step size is observed in Figure 5.2a as the number of stages increases, which is consistent with previous studies [55]. The relative performance of third-order scheme, measured as $\Delta t_{opt}/e$, for each order in space, is shown in Figure 5.2b. These results show that increasing the number of stages, and resulting terms in the stability polynomial, steadily increase the maximum stable time step size relative to the number of stages for a third-order accurate scheme in time. It is observed that third-order accurate temporal scheme benefits from increasing the number of stages up to $e = 16$.

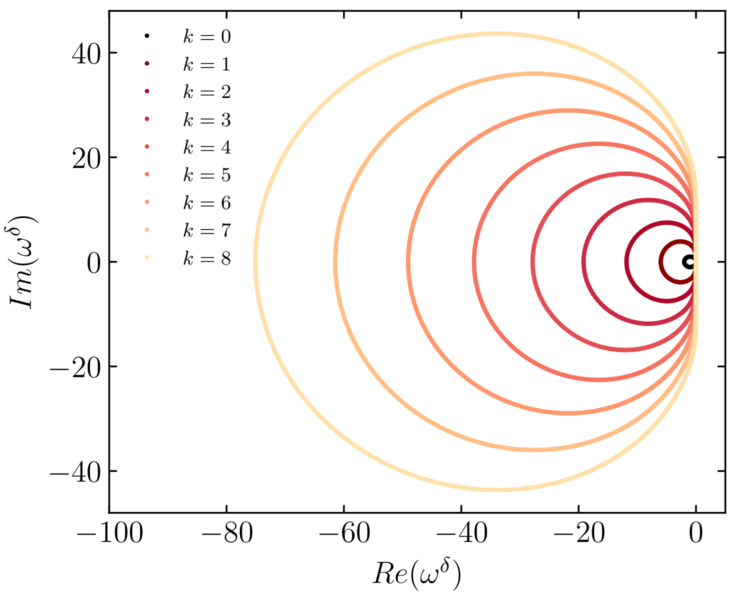
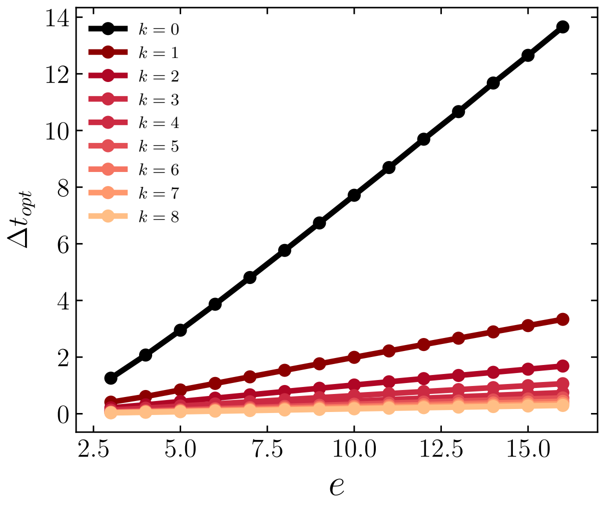
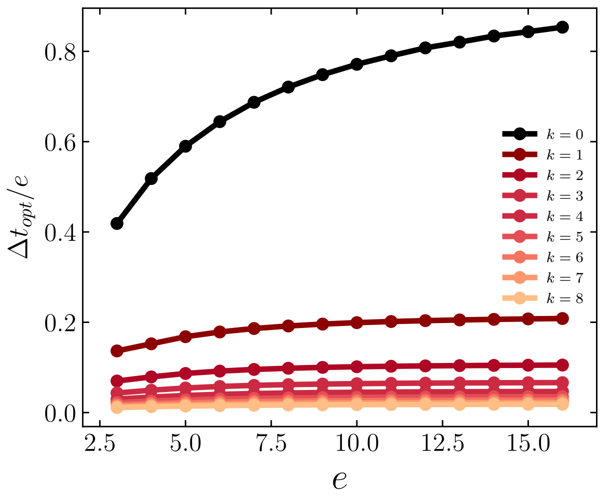


Figure 5.1. Collection of eigenvalues for different spatial order of accuracy.



(a) Values of Δt_{opt}



(b) Values of Δt_{opt} normalized by e

Figure 5.2. Values of Δt_{opt} for solution polynomials of $k = 0$ to $k = 8$ with between $e = 3$ and $e = 16$ derivative evaluations

These optimal stability polynomials along with the relationship in Equation 5.12, form a system of nonlinear equations for each member of third-order P-ERK family with 16 number of stages and e active stages, whose solution determines the unknown coefficients in corresponding Butcher tableau.

5.4 Numerical Results

5.4.1 Verification

To verify the accuracy of the Paired Explicit Runge-Kutta schemes (P-ERK) family of schemes, we consider an isentropic vortex using the Euler equations. This problem has an exact analytical solution, making it useful for verification. This exact solution is simply propagation of the isentropic vortex with the flow. The initial flow field is specified as

$$
\begin{aligned}
\rho &= \left(1 - \frac{S^2M^2(\gamma-1)e^{2f}}{8\pi^2}\right)^{\frac{1}{\gamma-1}}, \\
u_x &= \frac{Sye^f}{2\pi r_c}, \\
u_y &= 1 - \frac{Sxe^f}{2\pi r_c}, \\
P_0 &= \frac{\rho^\gamma}{\gamma Ma^2},
\end{aligned}
\tag{5.16}
$$

where ρ is the density, v_x and v_y are the velocity components, P_0 is the pressure, $f = \frac{(1-x^2-y^2)}{2r_c^2}$, S=13.5 is the vortex strength, $Ma = 0.4$ is the free stream Mach number, and $r_c = 1.5$ is vortex radius, and $\gamma = 1.4$ is the specific heat ratio . Figure 5.3a displays the initial density contours for the isentropic vortex.

We defined a $[0, 20] \times [0, 20]$ two-dimensional domain with 20×20 quadrilateral elements and periodic boundary conditions in the top and bottom boundaries and Riemann invariant boundary conditions on the side boundaries. The solution polynomials degree was taken to be $k = 6$ to minimize the spatial error. Fourteen simulations with each member of a third-order P-ERK family with $s = 16$ (from P-ERK$_{16,3,3}$ through P_ERK$_{16,16,3}$) and one simulation with a random distribution of all third-order P-ERK schemes, shown in Figure 5.3b, were run. Different time-step sizes up to the stability limit of each scheme were used for forty cycles of the vortex through the domain. To evaluate the accuracy of each scheme we consider the L_2 norm of the density error at the end of each simulation, defined as

$$\sigma = \sqrt{\int_0^{20} \int_0^{20} \Big(\rho_h(x,y) - \rho_e(x,y)\Big)^2 dxdy}, \tag{5.17}$$

where $\rho_h(x,y)$ is the final numerical solution and $\rho_e(x,y)$ is the exact analytical solution, which is identical to the initial condition. Figure 5.4 shows a logarithmic plot of the density error for each simulation versus the corresponding time-step size. It is apparent that all members achieved their designed third-order of accuracy. Members with a relatively high number of derivative evaluations have lower error for a particular time-step size compared to members with a relatively low number of derivative evaluations. Moreover, the simulation with a random distribution of all P-ERK$_{16,e,3}$ schemes also achieved third-order accuracy. A summary of density error for each simulation is also provided in Table 5.1. This shows that the P-ERK schemes proposed, have achieved their designed third-order accuracy.

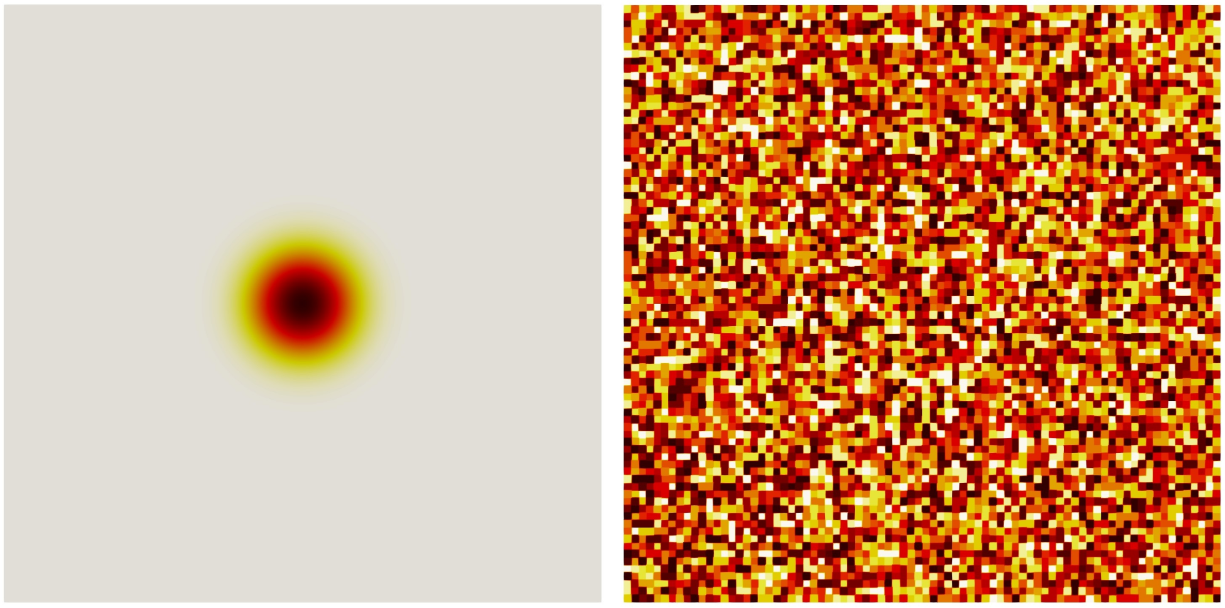

(a) Initial density distribution (b) Random third-order P-ERK time levels

Figure 5.3. Initial density distribution and random third-order PERK time levels for the isentropic vortex.

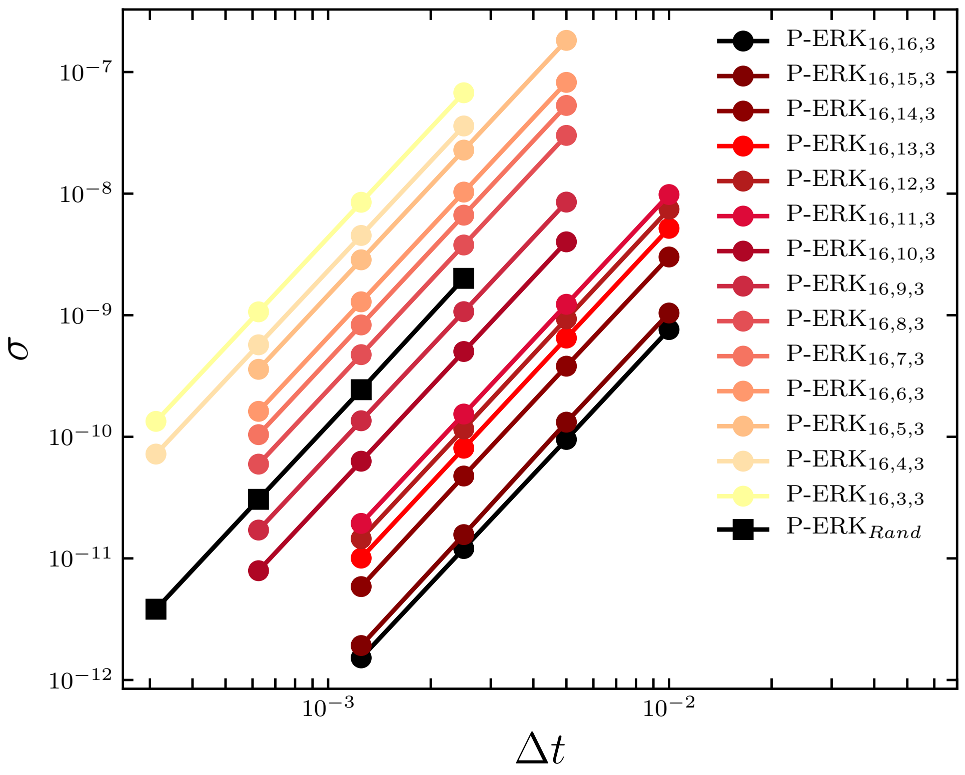


Figure 5.4. Convergence plots for the isentropic vortex case with different members of P-ERK$_{16,e,3}$.

Table 5.1. Density error and order of accuracy for the isentropic vortex case.

Scheme	Time-Step Size	Error	Order of Accuracy
P-ERK$_{16,3,3}$	2.500×10^{-3}	6.7526×10^{-08}	
	1.250×10^{-3}	8.4895×10^{-9}	2.99
	6.250×10^{-4}	1.0659×10^{-9}	2.99
	3.120×10^{-4}	1.3395×10^{-10}	2.99
P-ERK$_{16,4,3}$	2.500×10^{-3}	3.6012×10^{-08}	
	1.250×10^{-3}	4.5295×10^{-9}	2.99
	6.250×10^{-4}	5.7089×10^{-10}	2.98
	3.120×10^{-4}	7.2095×10^{-11}	2.98
P-ERK$_{16,5,3}$	5.000×10^{-2}	1.8217×10^{-07}	
	2.500×10^{-3}	2.2795×10^{-08}	2.99
	1.250×10^{-3}	2.8551×10^{-09}	2.99
	6.250×10^{-4}	3.5895×10^{-10}	2.99
P-ERK$_{16,6,3}$	5.000×10^{-2}	8.2215×10^{-08}	
	2.500×10^{-3}	1.0295×10^{-08}	2.99
	1.250×10^{-3}	1.2889×10^{-09}	2.99
	6.250×10^{-4}	1.6195×10^{-10}	2.98
P-ERK$_{16,7,3}$	5.000×10^{-2}	5.3426×10^{-08}	
	2.500×10^{-3}	6.6495×10^{-09}	2.99
	1.250×10^{-3}	8.3189×10^{-10}	2.99
	6.250×10^{-4}	1.0415×10^{-10}	2.99
P-ERK$_{16,8,3}$	5.000×10^{-2}	3.0125×10^{-08}	
	2.500×10^{-3}	3.7805×10^{-09}	2.99
	1.250×10^{-3}	4.7389×10^{-10}	2.99
	6.250×10^{-4}	5.9595×10^{-11}	2.98
P-ERK$_{16,9,3}$	5.000×10^{-2}	8.5178×10^{-09}	
	2.500×10^{-3}	1.0695×10^{-09}	2.99
	1.250×10^{-3}	1.3551×10^{-10}	2.98
	6.250×10^{-4}	1.7095×10^{-11}	2.98

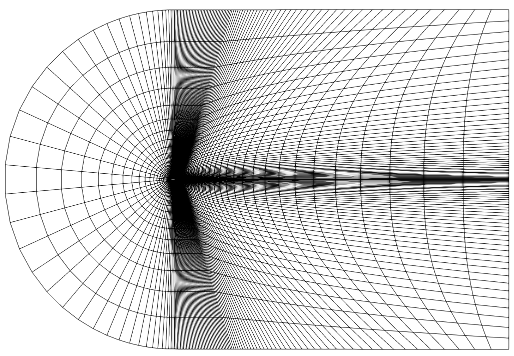
$\text{P-ERK}_{16,10,3}$	5.000×10^{-2}	4.0125×10^{-09}	
	2.500×10^{-3}	5.0295×10^{-10}	2.99
	1.250×10^{-3}	6.2992×10^{-11}	2.99
	6.250×10^{-4}	7.9195×10^{-12}	2.99
$\text{P-ERK}_{16,11,3}$	1.000×10^{-2}	9.8257×10^{-09}	
	5.000×10^{-2}	1.2295×10^{-09}	2.99
	2.500×10^{-3}	1.5389×10^{-10}	2.99
	1.250×10^{-3}	1.9395×10^{-11}	2.98
$\text{P-ERK}_{16,12,3}$	1.000×10^{-2}	7.4458×10^{-09}	
	5.000×10^{-2}	9.3272×10^{-10}	2.99
	2.500×10^{-3}	1.1589×10^{-10}	3.00
	1.250×10^{-3}	1.4525×10^{-11}	2.99
$\text{P-ERK}_{16,13,3}$	1.000×10^{-2}	5.1825×10^{-09}	
	5.000×10^{-2}	6.4872×10^{-10}	2.99
	2.500×10^{-3}	8.0589×10^{-11}	3.00
	1.250×10^{-3}	1.0052×10^{-11}	2.99
$\text{P-ERK}_{16,14,3}$	1.000×10^{-2}	3.0125×10^{-09}	
	5.000×10^{-2}	3.8095×10^{-10}	2.98
	2.500×10^{-3}	4.7589×10^{-11}	3.00
	1.250×10^{-3}	5.8595×10^{-12}	2.99
$\text{P-ERK}_{16,15,3}$	1.000×10^{-2}	1.0426×10^{-09}	
	5.000×10^{-2}	1.3195×10^{-10}	2.98
	2.500×10^{-3}	1.5689×10^{-11}	3.00
	1.250×10^{-3}	1.9084×10^{-12}	2.99
$\text{P-ERK}_{16,16,3}$	1.000×10^{-2}	7.6178×10^{-10}	
	5.000×10^{-2}	9.5115×10^{-11}	3.00
	2.500×10^{-3}	1.2051×10^{-11}	2.98
	1.250×10^{-3}	1.5195×10^{-12}	2.98
$\text{P-ERK}_{16,Random,3}$	2.500×10^{-3}	2.0125×10^{-09}	
	1.250×10^{-3}	2.4395×10^{-10}	3.00
	6.250×10^{-4}	3.0558×10^{-11}	2.98
	3.120×10^{-4}	3.8195×10^{-12}	2.98

5.4.2 Laminar Flow Over an SD7003 Airfoil

To investigate the application of third-order P-ERK schemes to the unsteady Navier-Stokes equations, we consider laminar flow over an SD7003 airfoil at $Re_c = U_\infty c/\nu = 10,000$, where U_∞ is free stream velocity, ν is kinematic viscosity and c is the chord length of the airfoil. The free stream mach number is $Ma_\infty = U_\infty/C = 0.2$, where C is the speed of the sound, the angle of attack is taken to be $\alpha = 4^\circ$, the Prandtl number is $Pr = 0.71$, and the specific heat ratio is $\gamma = 1.4$.

Since the flow is planar, we used a two-dimensional geometry of $[-20c, 40c]\times[-20c, 20c]$, composed of 17,997 quadratically curved quadrilateral elements. The mesh is refined in the wake to capture physics of unsteady flow at the airfoil's trailing edge. Figures 5.5a and 5.5b show the computational domain and near-surface mesh. The airfoil's surface is specified as a no-slip adiabatic wall, and the far-field boundary condition is specified as Riemann invariant. The FR approach was used for spatial discretization, and common interface fluxes are computed using the Rusanov scheme for inviscid fluxes and the second method of Bassi and Rebay (BR2) for viscous fluxes [95].

To assess performance, we ran simulations with third-order P-ERK$_{16,e,3}$ schemes, and the classical RK$_{3,3}$ scheme using solution polynomial degrees of $k = 1$ to $k = 4$. We first determined the maximum permissible time-step size iteratively for both RK$_{3,3}$ and P-ERK$_{16,16,3}$ schemes for all polynomial degrees. Then, the domain was split into regions using P-ERK$_{16,16,3}$, P-ERK$_{16,12,3}$, P-ERK$_{16,8,3}$, P-ERK$_{16,4,3}$, and P-ERK$_{16,3,3}$, based on minimum element edge lengths, as shown in Figure 5.6. Schemes with a large number of active stages were employed for small elements in the boundary layer and the wake region at the trailing edge of the airfoil. In contrast, schemes with fewer active stages were used further away from the airfoil surface, where element sizes are larger. Each simulation was run to a final simulation time of $100t_c$ where $t_c = c/U_\infty$ is a non-dimensional convective time.


(a) Farfield mesh

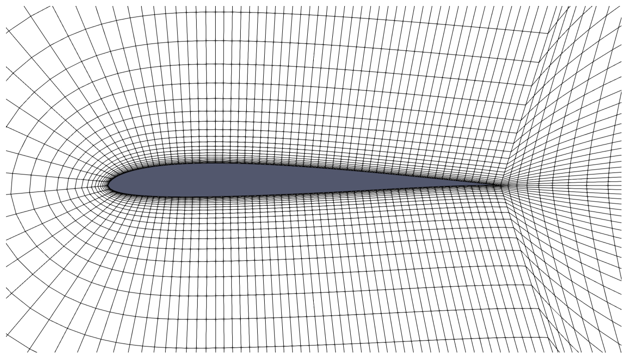

(b) Near-surface mesh

Figure 5.5. Domain and near surface mesh for laminar flow over an SD7003 airfoil.

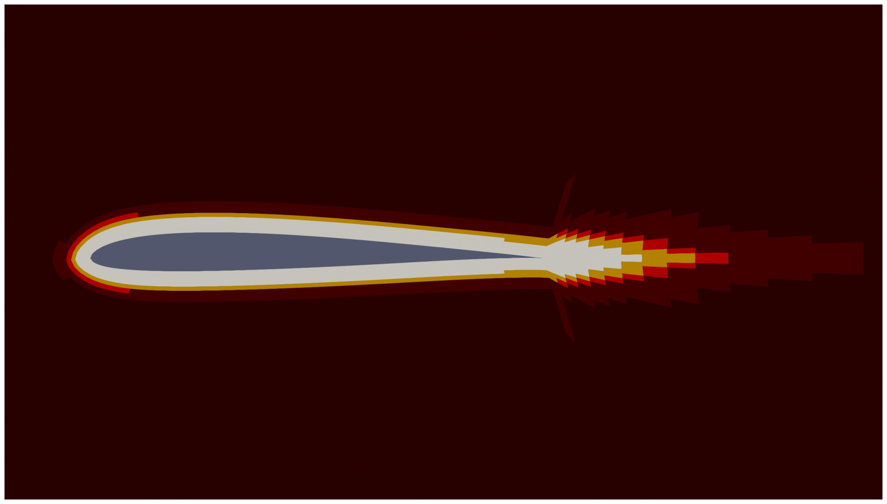
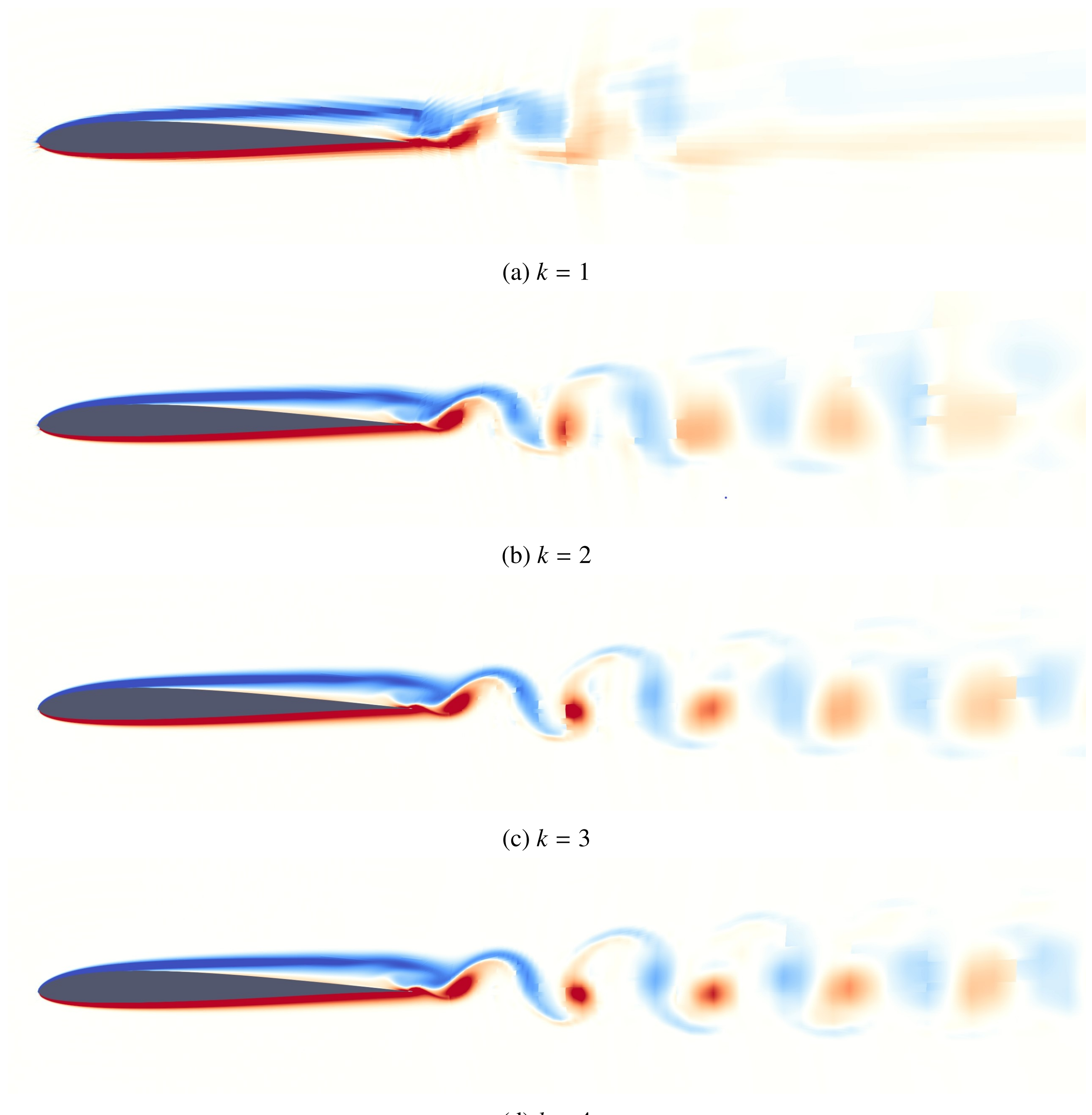


Figure 5.6. Distribution of P-ERK schemes, P-ERK$_{16,16,3}$ (white), P-ERK$_{16,12,3}$ (yellow), P-ERK$_{16,8,3}$ (orange), P-ERK$_{16,4,3}$ (red), P-ERK$_{16,3,3}$ (maroon red) for the laminar SD7003 airfoil.

Contours of the z-component of vorticity, shown in Figure 5.7, demonstrate that third-order P-ERK schemes are able to capture the boundary layer and unsteady wake structure for each polynomial degree. A summary of the time-step ratio, speedup factor and time-averaged aerodynamic forces for each solution polynomial degree is presented in Table 5.2. A time-step size ratio of up to 8.02 can be achieved using third-order P-ERK scheme, when compared to RK$_{3,3}$. This larger time-step size was achieved at the relatively small cost of additional derivative evaluations only for a few elements in the stiff part of the domain. This resulted in the P-ERK simulations being up to 4.51 times faster than those with classical RK$_{3,3}$. Moreover, time-averaged lift and drag coefficients, presented in Table 5.2, are in good agreement with available reference datasets, and, the results using third-order

P-ERK schemes are nearly identical to those with $\mathrm{RK}_{3,3}$.



(a) $k = 1$

(b) $k = 2$

(c) $k = 3$

(d) $k = 4$

Figure 5.7. Contours of the z-component of vorticity for laminar flow over an SD7003 airfoil using P-ERK$_{16,e,3}$ schemes.

Table 5.2. Quantitative results for laminar flow over an SD7003 airfoil using third-order P-ERK$_{16,e,3}$ and RK$_{3,3}$ schemes.

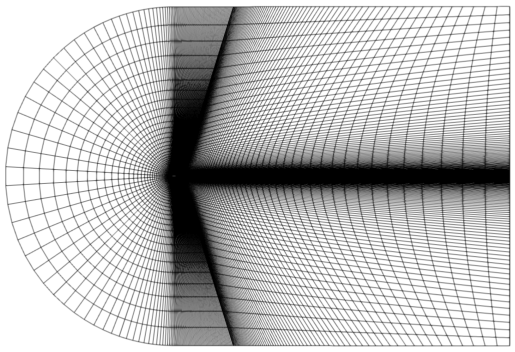
Polynomial Degree	Scheme	Time-Step Ratio	Speedup Factor	$\overline{C}_L$	$\overline{C}_D$
$k = 1$	RK$_{3,3}$	7.59	4.43	0.3315	0.0529
	P-ERK$_{16,e,3}$			0.3325	0.0529
$k = 2$	RK$_{3,3}$	7.99	4.51	0.3655	0.0495
	P-ERK$_{16,e,3}$			0.3654	0.0495
$k = 3$	RK$_{3,3}$	7.78	4.39	0.3848	0.0491
	P-ERK$_{16,e,3}$			0.3848	0.0492
$k = 4$	RK$_{3,3}$	8.02	4.47	0.4022	0.0492
	P-ERK$_{16,e,3}$			0.4021	0.0492
Uranga et al. [34]				0.3755	0.0498
Lopez-Morales et al. [96]				0.3719	0.0494

5.4.3 Turbulent Flow Over an SD7003 Airfoil

We now consider transitional and turbulent flow over an SD7003 airfoil at a higher Reynolds number of $Re_c = U_\infty c/\nu = 60,000$. We used a Mach number based on the free stream velocity of $Ma_\infty = 0.1$, an angle of attack $\alpha = 8^\circ$, a specific heat ratio of $\gamma = 1.4$ and a Prandtl number $Pr = 0.71$. This is a commonly used test case, with multiple reference datasets available for comparison [97, 98, 99]. We used a three-dimensional mesh of $[-20c, 40c] \times [-20c, 20c] \times [0, 0.2c]$ as shown in Figure 5.8. The quadratically curved mesh is composed of 60,720 hexahedral elements. The mesh is refined in the boundary layer and wake regions to capture complex unsteady turbulent phenomena expected in these regions. The airfoil surface is specified using a no-slip adiabatic wall boundary condition, and the far field is specified as Riemann invariant.

Two sets of simulations were run using P-ERK$_{16,e,3}$ and RK$_{3,3}$ with solution polynomials of degree $k = 1, 2,$ and 3. Each simulation used Gauss points within each volume and at each element face, with common fluxes computed using the Rusanov and BR2 schemes [95]. The boundary layer resolution gives $y^+ \approx 0.77, 0.41$ and 0.25 for $k = 1, 2,$ and 3, respectively [97], where $y^+ = u_\tau y/\nu$ is the wall normal distance to the first solution point normal to the wall, ν is the kinematic viscosity, $u_\tau = \sqrt{c_f/2}U_\infty$, and $c_f \approx 8.5 \times 10^{-3}$ is the maximum skin friction coefficient in the turbulent region reported by Garmann et al. [97]. The computational domain was split into P-ERK$_{16,16,3}$, P-ERK$_{16,12,3}$, P-ERK$_{16,8,3}$, P-ERK$_{16,4,3}$, and P-ERK$_{16,3,3}$, based on minimum element edge lengths as shown in Figure 5.9. Similar to the laminar case, maximum allowable time-step sizes were determined for each solution polynomial with both the RK$_{3,3}$ and P-ERK$_{16,16,3}$ schemes. All simulations were run until $t = 40t_c$, and time-averaged data were collected in the time interval $t_c \in [20, 40]$.

Figure 5.10 shows contours of q-criterion colored by velocity magnitude for all three solution polynomials. This shows that P-ERK time integration is well-resolved and remained stable. It is observed that a small separation bubble near the leading edge of the airfoil. The flow transition is seen along the bubble and fully turbulent wake downstream at trailing edge of the airfoil.


(a) Farfield mesh

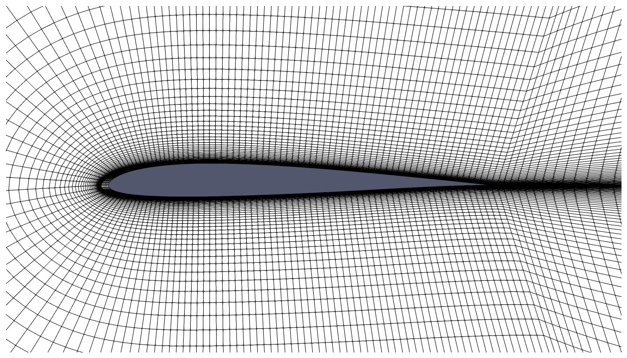

(b) Near-surface mesh

Figure 5.8. Domain and near surface mesh for turbulent flow over an SD7003 airfoil.

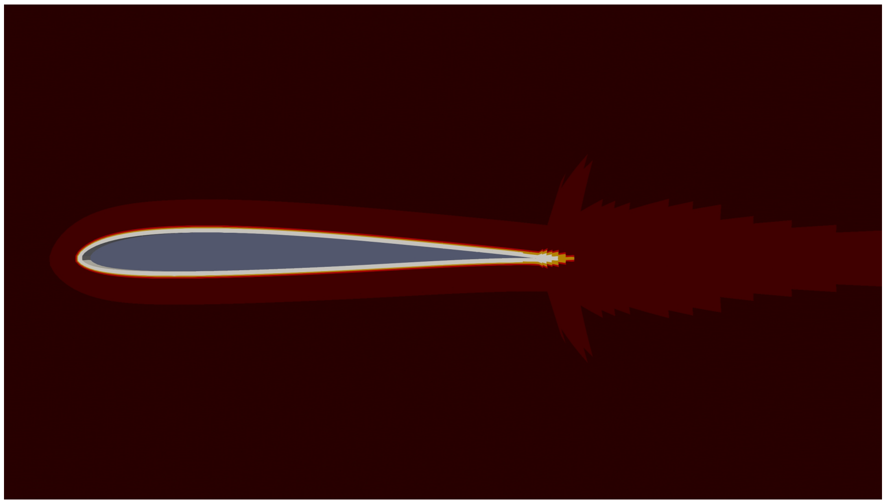
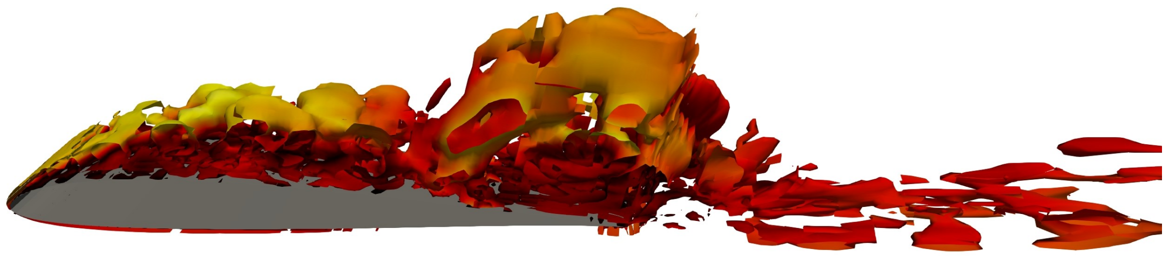
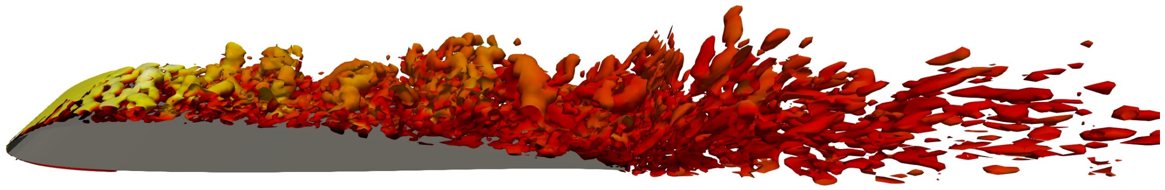


Figure 5.9. Distribution of P-ERK schemes, P-ERK$_{16,16,3}$ (white), P-ERK$_{16,12,3}$ (yellow), P-ERK$_{16,8,3}$ (orange), P-ERK$_{16,4,3}$ (red), P-ERK$_{16,3,3}$ (maroon red) for the turbulent SD7003 cases.

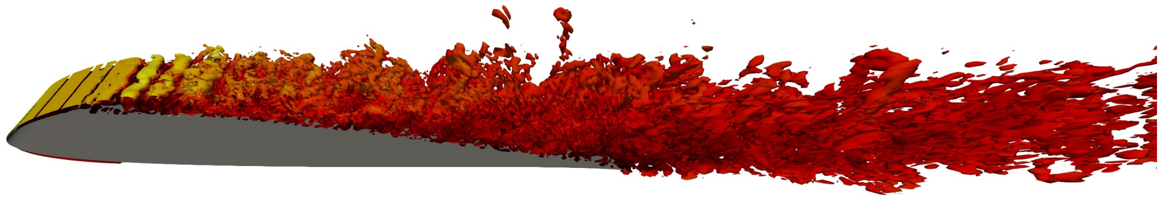
Table 5.3 shows the time-step ratio, speedup factor and time-averaged aerodynamic force coefficients for all simulations. Speedup factors of up to 4.25 were observed when using the P-ERK schemes relative to RK$_{3,3}$, while maintaining good agreement with the reference data in terms of lift and drag coefficients. The time-averaged pressure coefficients on the surface of airfoil that are plotted in Figure 5.11, display excellent agreement with the reference datasets as we refine the solution polynomial degree. These results demonstrate the utility of third-order P-ERK schemes for wall bounded turbulent flows, while accelerating the simulation by a factor of 4.2 relative to RK$_{3,3}$.



(a) $k = 1$



(b) $k = 2$



(c) $k = 3$

Figure 5.10. Contours of q-criterion colored by velocity magnitude for turbulent flow over an SD7003 airfoil using P-ERK$_{16,e,3}$.

Table 5.3. Quantitative results for turbulent flow over an SD7003 airfoil using third-order P-ERK$_{16,e,3}$ and RK$_{3,3}$ schemes.

Polynomial degree	Scheme	Time step ratio	Speedup factor	$\overline{C}_L$	$\overline{C}_D$
$k = 1$	RK$_{3,3}$	7.48	4.25	0.985	0.121
	P-ERK$_{16,e,3}$			0.984	0.122
$k = 2$	RK$_{3,3}$	7.35	4.09	0.958	0.058
	P-ERK$_{16,e,3}$			0.958	0.058
$k = 3$	RK$_{3,3}$	7.28	4.15	0.931	0.048
	P-ERK$_{16,e,3}$			0.931	0.048
Garmann et al. [97]				0.969	0.039
Beck et al. [98]				0.932	0.050
Vermeire et al. [99]				0.941	0.049

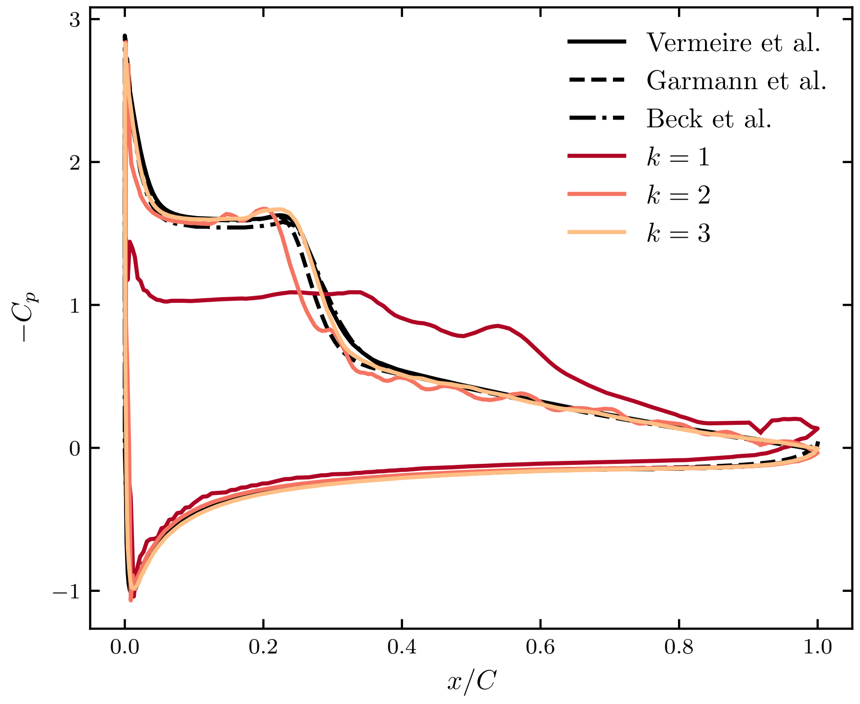


Figure 5.11. Pressure coefficient distribution for turbulent SD7003 airfoil simulations using P-ERK$_{16,e,3}$ schemes.

5.4.4 Tandem Spheres

As a final demonstration case, we consider an advanced benchmark test case from the Fifth International Workshop on High-Order CFD Methods (HiOCFD5) [87]. This case consists of two spheres of diameter D, separated by a distance $10D$ in the streamwise direction. The Reynolds number for this case is $Re_D = U_\infty D/\nu = 3900$, the Mach number is $Ma_\infty = 0.1$, the Prandtl number is $Pr = 0.72$, the specific heat ratio is $\gamma = 1.4$, and time is non-dimensionalized using the convective scale $t_D = tU_\infty/D$.
Laminar separation is expected over the front sphere, followed by transition into a turbulent wake. Further downstream, this turbulent wake passes over the trailing sphere [100]. Meshes for this case have been provided for the HiOCFD5 workshop by Steve Karman from Pointwise and Samuel James from GridPro with different resolutions using either tetrahedral or hexahedral elements. In this study, we used the GridPro mesh with 237,748 quadratically curved hexahedral elements and a solution polynomial of degree $k = 4$. Figure 5.13 shows the near-surface mesh around the spheres. We split the domain into P-ERK$_{16,16,3}$, P-ERK$_{16,12,3}$, P-ERK$_{16,8,3}$, P-ERK$_{16,4,3}$, and P-ERK$_{16,3,3}$, based on minimum element edge lengths as shown in Figure 5.14. Similar to previous cases, the interface fluxes were computed using the Rusanov and BR2 [95] schemes with Gauss points along each element face. The simulation was run until a final non-dimensionalized time of $t_D = 200$, and quantities of interests were averaged over the time interval of $t_D \in [100, 200]$, as suggested in the HiOCFD5 instructions. The maximum allowable time-step size and computational cost per convective time were determined using both P-ERK$_{16,16,3}$ and RK$_{3,3}$. The third-order P-ERK schemes have a time-step size ratio of 8.21 relative to RK$_{3,3}$ and a speedup factor of 4.06. Table 5.4, shows time-averaged drag coefficients of the front and back spheres using third-order P-ERK schemes alongside reference data from HiOCFD5, demonstrating good agreement. Figures 5.17 and 5.18 display the evolution of lift and drag coefficients over time.

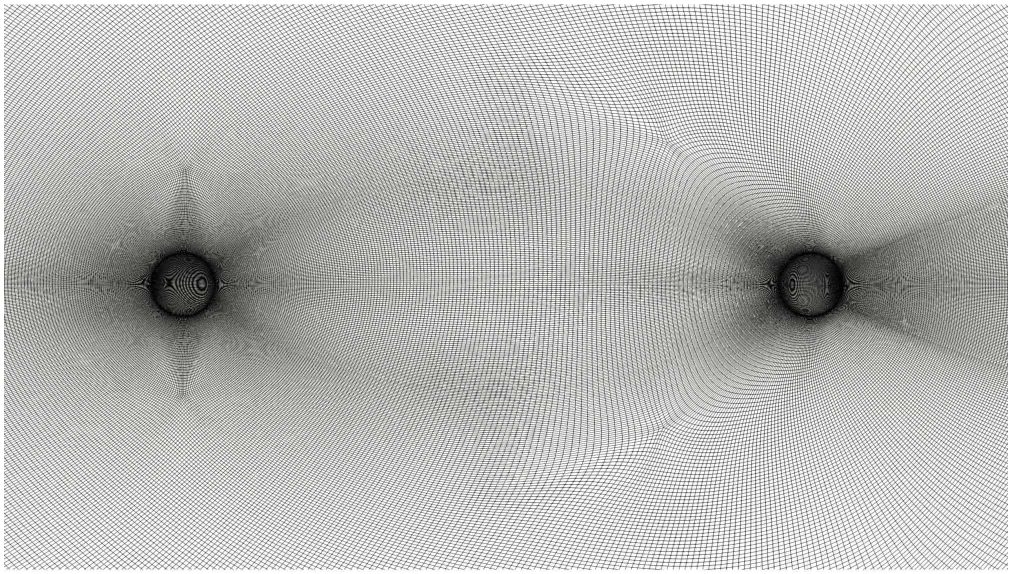


Figure 5.12. Near-surface mesh

Figure 5.13. The domain and near surface meshes for tandem spheres

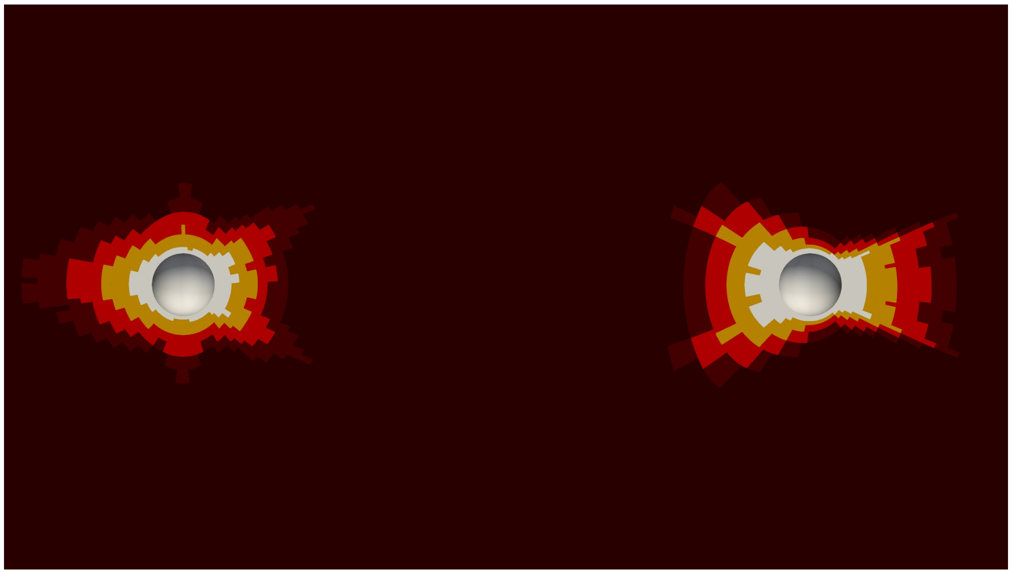


Figure 5.14. Distribution of P-ERK schemes, P-ERK$_{16,16,3}$ (white), P-ERK$_{16,12,3}$ (yellow), P-ERK$_{16,8,3}$ (orange), P-ERK$_{16,4,3}$ (red), P-ERK$_{16,3,3}$ (maroon red).

Figure 5.15 shows contours of vorticity magnitude at a non-dimensionalaized time $t = 193t_D$, which shows that third-order P-ERK time integration resolves the expected physics of this test case. In addition, Figure 5.16 displays contours of q-criterion colored by velocity magnitude. This shows that laminar separation occurs on the surface of the front sphere, and its transition into a fully turbulent wake before hitting the back sphere.

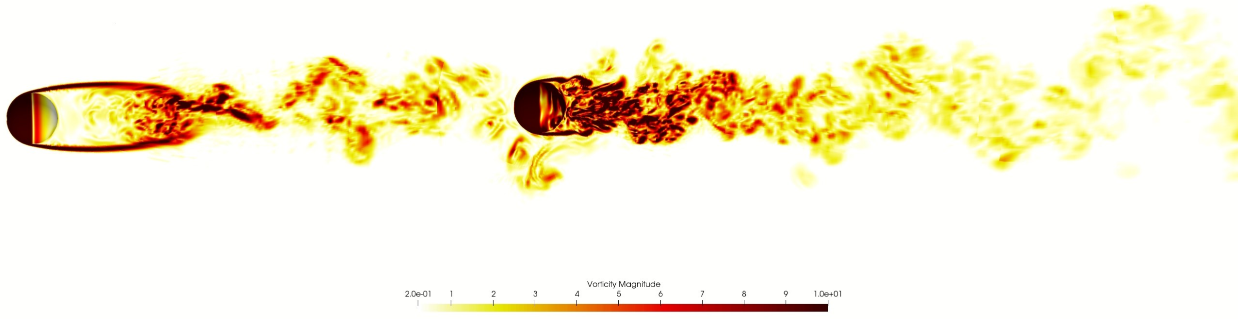


Figure 5.15. Contours of vorticity magnitude.

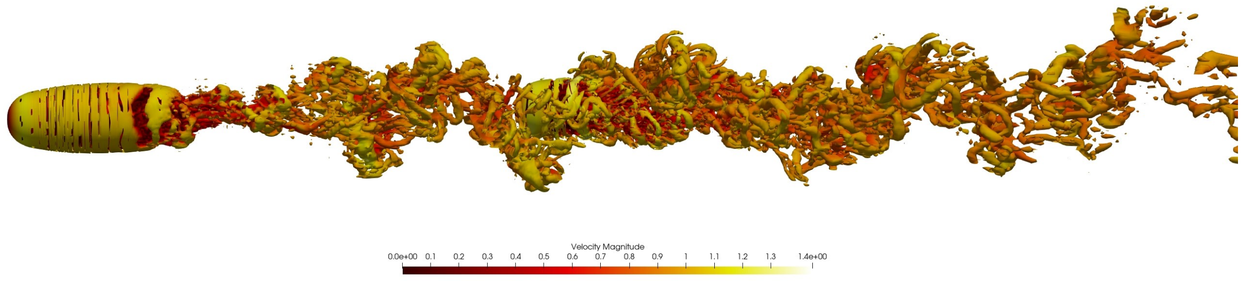


Figure 5.16. Contours of q-criterion colored by velocity magnitude.

Table 5.4. Quantitative results of tandem spheres case using P-ERK$_{16,e,3}$ along with reference data.

Configuration	Front Sphere $\overline{C}_D$	Back Sphere $\overline{C}_D$
30.1 DOFs using P-ERK$_{16,e,3}$	0.382	0.411
69.4 DOFs Holst et al. [101]	0.387	0.410
5.2M DOFs Xue et al. [102]	0.367	N/A
12.3M DOFs Xue et al. [102]	0.374	N/A
6.94M DOFs Jansson et al. [103]	0.440	0.445
69.4M DOFs Martinelli et al. [100]	0.411	0.439
21.5M DOFs Martinelli et.al [100]	0.457	0.458

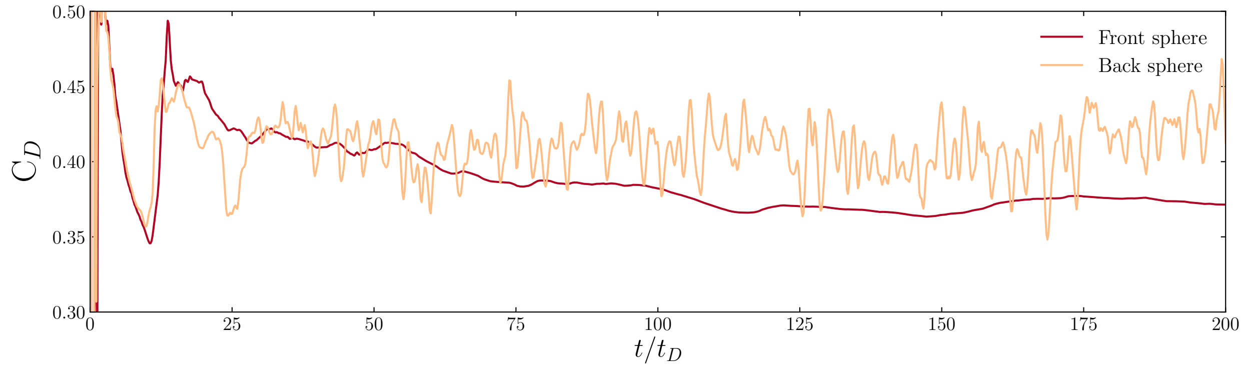


Figure 5.17. Evolution of drag coefficient for front and back spheres over time.

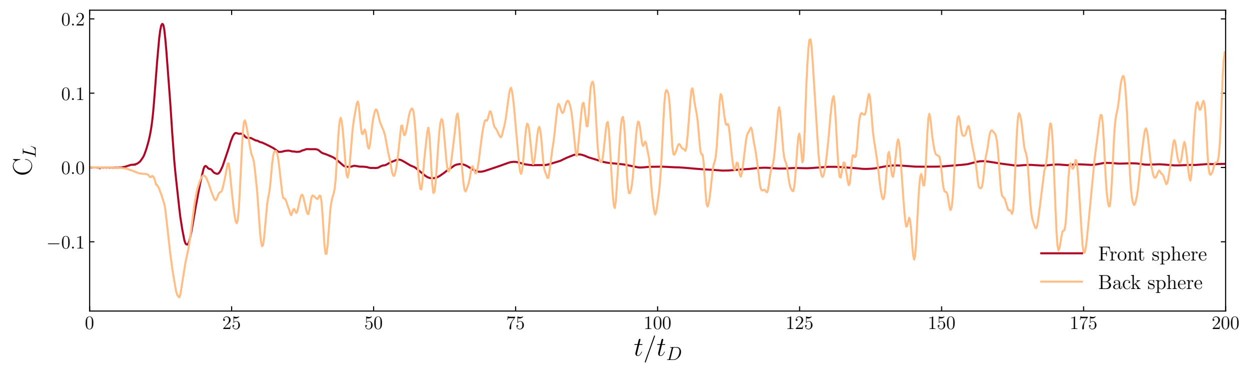


Figure 5.18. Evolution of lift coefficient for front and back spheres over time.

Figure 5.19 shows the time-averaged streamwise velocity component along the domain centreline. These are shown alongside reference results from Martinelli et al. [100], which were run using Pointwise 6.95 million tetrahedral elements and at a 69.5M DOFs. Some discrepancies are observed in the wake of the first sphere. This may be attributable to different resolution used in the current case, or the presence of a low frequency wake oscillation [104]. However, outside of this region, generally good agreement is observed, including in the wake region of the second sphere. Figures 5.20, 5.21, and 5.22 show the mean $\overline{u'u'}$, $\overline{v'v'}$ and $\overline{w'w'}$ along the centreline of the spheres. Similarly, some discrepancies are seen in the wake of first sphere. Figures 5.23, 5.24, 5.25, and 5.26 display time averaged $\overline{u}$, $\overline{v}$, $\overline{u'u'}$, and $\overline{v'v'}$ at five different locations along the flow direction, alongside reference results from Martinelli et al. [100]. Good agreement is again observed for all variables, apart from wake region immediately behind the front sphere.

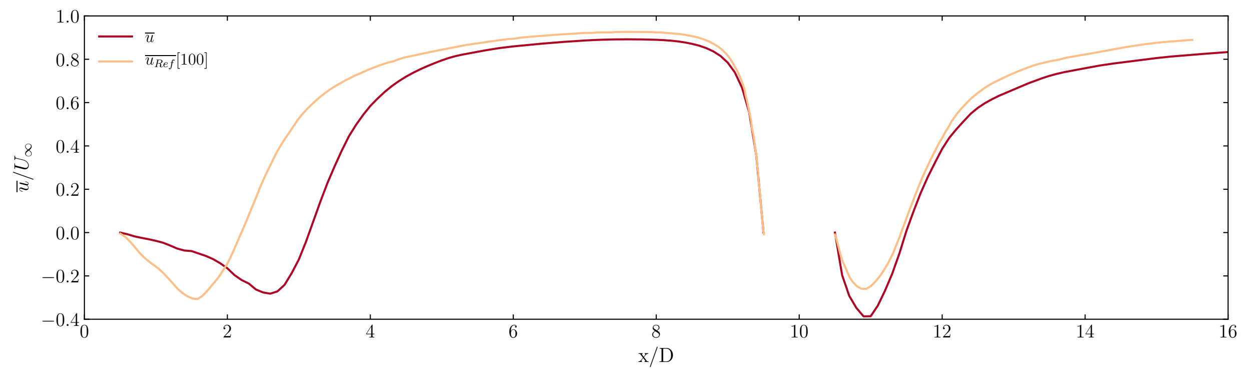


Figure 5.19. Mean $\overline{u}$ along the tandem spheres centerline.

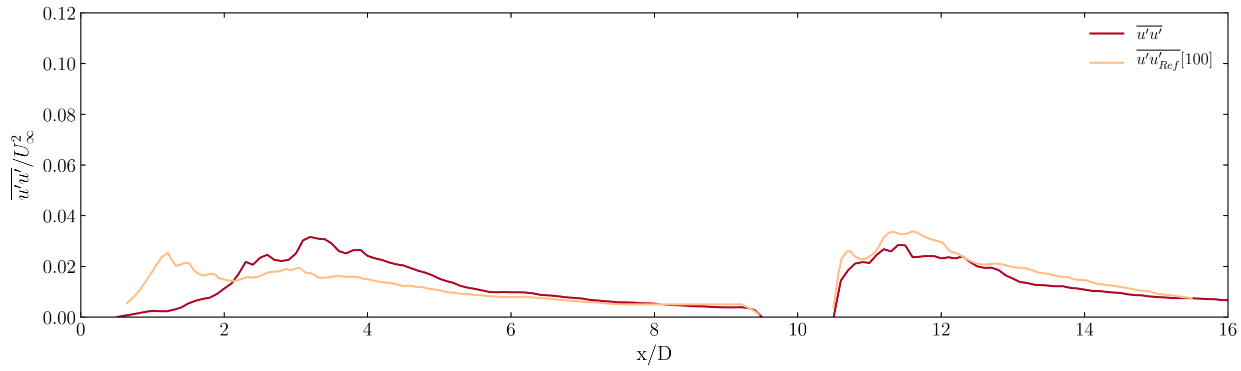


Figure 5.20. Mean $\overline{u'u'}$ along the tandem spheres centerline.

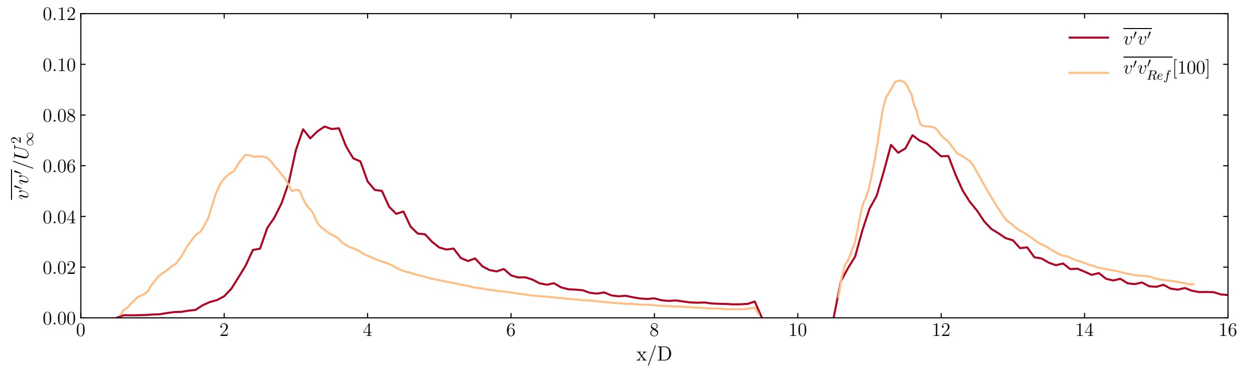


Figure 5.21. Mean $\overline{v'v'}$ along the tandem spheres centerline.

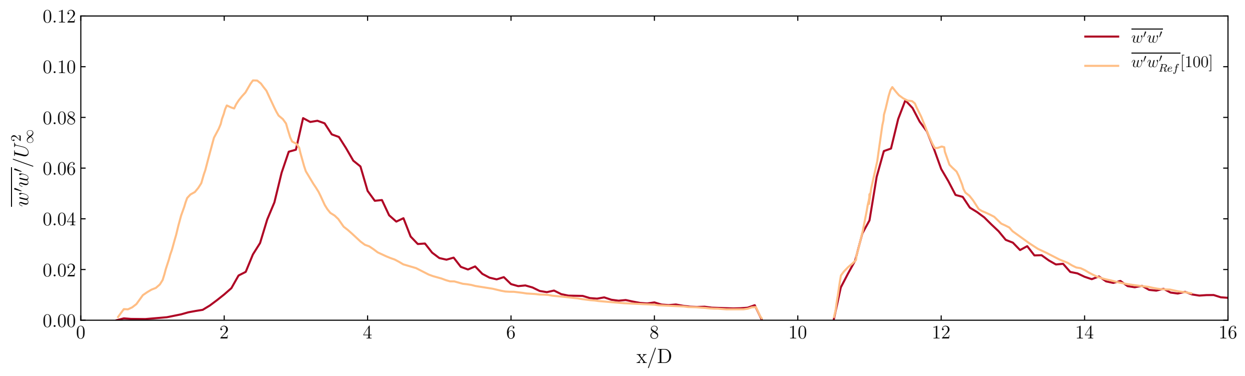


Figure 5.22. Mean $\overline{w'w'}$ along the tandem spheres centerline.

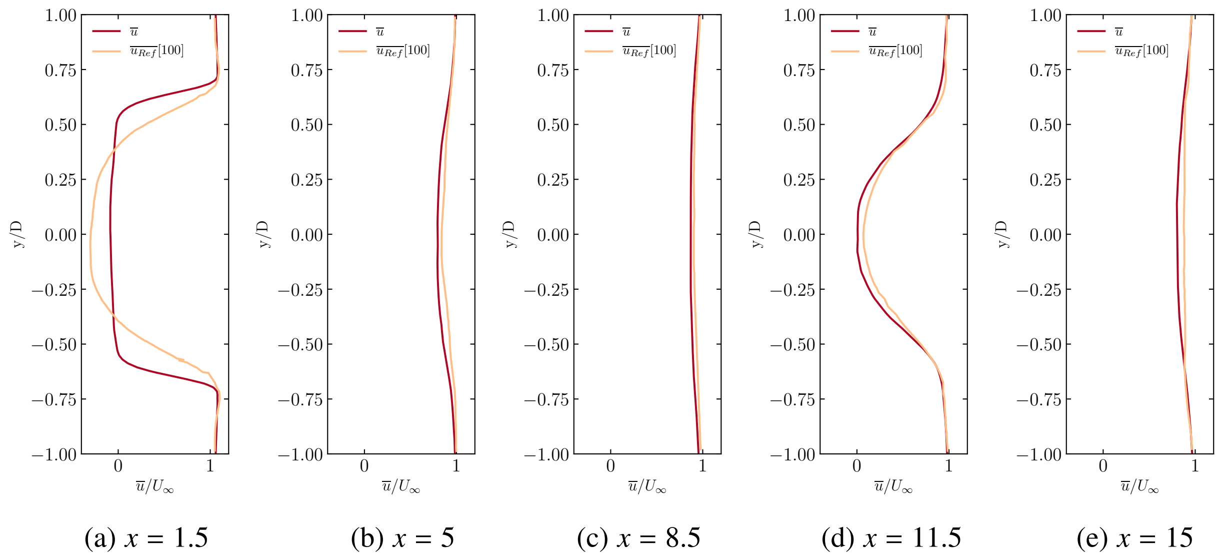
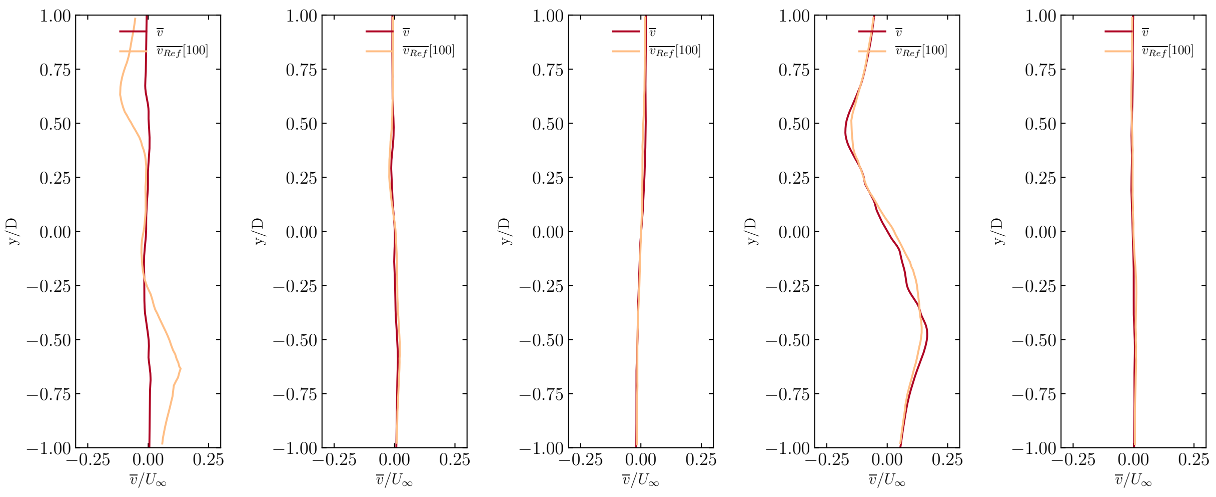
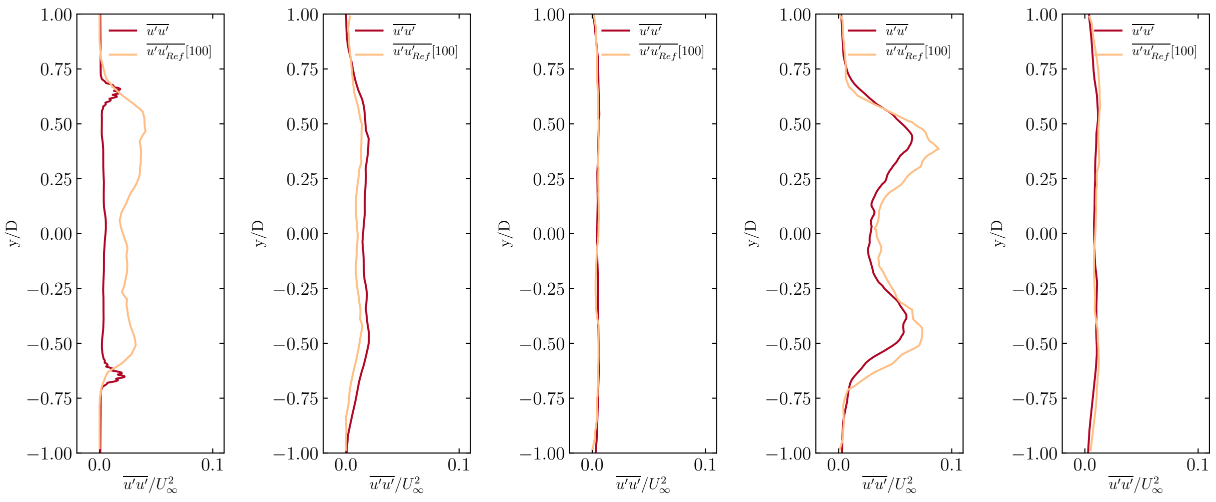


Figure 5.23. Mean $\overline{u}$ on different sections along flow direction.



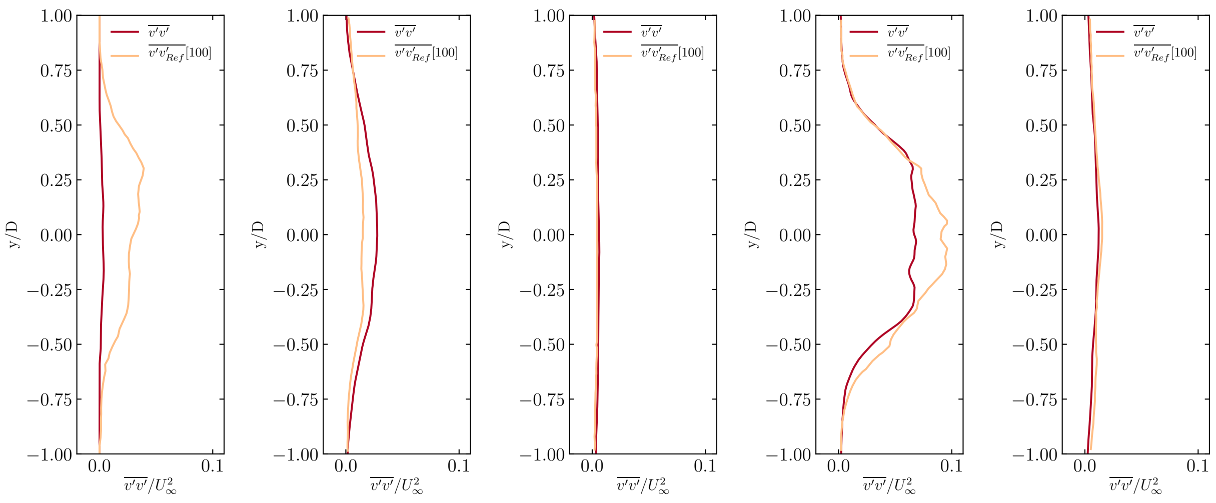
(a) x = 1.5 (b) x = 5 (c) x = 8.5 (d) x = 11.5 (e) x = 15

Figure 5.24. Mean $\overline{v}$ on different sections along flow direction.



(a) x = 1.5 (b) x = 5 (c) x = 8.5 (d) x = 11.5 (e) x = 15

Figure 5.25. Mean $\overline{u'u'}$ on different sections along flow direction.



(a) $x = 1.5$ (b) $x = 5$ (c) $x = 8.5$ (d) $x = 11.5$ (e) $x = 15$

Figure 5.26. Mean $\overline{v'v'}$ on different sections along flow direction.

5.5 Discussion

In this chapter, we have proposed a new family of third-order P-ERK schemes for locally-stiff systems of equations. Building on the original second-order P-ERK formulation, these third-order schemes allow Runge-Kutta schemes with different numbers of active stages to be assigned based on local stiffness criteria, while seamlessly pairing at their interface. We then generated families of schemes optimized for the high-order flux reconstruction spatial discretization, and applied them to a range of benchmark test cases, including isentropic vortex advection, laminar flow over an SD7003 airfoil, turbulent flow over an SD7003 airfoil, and turbulent flow over a tandem sphere configuration. A verification study using the isentropic vortex case confirmed that these P-ERK schemes achieve their designed third-order accuracy when used alone, or in arbitrary combinations with each other. Results for the laminar and turbulent cases demonstrated that P-ERK schemes consistently achieve speedup factors in excess of four for the Navier-Stokes equations, relative to the classical third-order Runge-Kutta scheme. Importantly, these speedup factors were achieved with negligible impact on quantitative results.

In summary, these results demonstrate that these new third-order P-ERK schemes provide an accurate and efficient framework for the solution of locally-stiff systems of equations, and that they significantly outperform classical approaches.

Chapter 6

Optimal Explicit Runge-Kutta Time Stepping for Density-Based Finite Volume Solvers

6.1 Authorship Statement

The mesh and configuration for the T106a turbine blade in this chapter was produced by Dr. Jean-Sébastien Cagnone at Ansys Canada Ltd. Apart from that, all of the contributions in this section were produced by Siavash Hedayati Nasab.

6.2 Background

A wide range of Computational Fluid Dynamics (CFD) applications require the numerical solution of unsteady turbulent flows. The ability to perform scale-resolving simulations of such flows, specifically via Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS), relies on numerical methods that are simultaneously accurate, efficient, and stable. In an industrial context, accurate schemes typically refers to those that are at least second-order accurate or higher, with minimal numerical dissipation and dispersion error. Efficiency relies on the respective abilities, and synergies therein, of the spatial discretization, temporal discretization, and chosen computing hardware architecture to reach a desired simulation time with minimal computational cost. Finally, stability can refer to both linear and non-linear stability of the resulting fully-discrete scheme. In the context of subsonic LES and DNS, stability is typically assessed in terms of the maximum achievable time step size.

High-fidelity simulations using LES/DNS are typically categorized as *fast transient* problems, due to the short lived nature of the smallest turbulent length scales. This means that relatively small time-step sizes are required to resolve these small scale structures. This encourages the use of explicit schemes due to their low cost per time-step. In the context of explicit time stepping, Runge-Kutta methods are widely used for both LES and DNS. They are appealing due to their ease of implementation, scalability and low computational cost per time step [38]. However, the maximum achievable time step size with these schemes is limited by their conditional stability properties [39, 70]. This stability limit is determined for Runge-Kutta methods via their associated stability polynomial, which is in turn a function of the Butcher tableau coefficients [59]. Previous research has shown the effectiveness of optimizing these stability polynomials for a given spatial discretization and system of equations to achieve the largest possible time step size for a given number of stages [55]. For example, the application of optimal of Runge-Kutta stability polynomials can be seen in the works of Van der Houwen et al. [71], Ruuth et al. [72, 73], Ketcheson et al. [59], Parsani et al. [74], Kubatko et al. [55], and Vermeire et al. [75]. In the context of the Finite-Volume Method (FVM), provably optimal stability polynomials for time dependent systems of equations have not as yet been generated. Previous efforts to optimize these stability polynomials for steady-state multigrid solutions of the Euler equation can be seen in the work of Van Leer et al. [105] and Lynn [106]. However, recent developments in the generation of provably optimal stability polynomials has not yet been exploited by the FVM community [107].

In this section, we generate optimal explicit Runge-Kutta methods for the FVM, and demonstrate their utility in a reduced dissipation formulation in Ansys Fluent. Fluent is an industry-standard Finite-Volume solver used to predict fluid-flow, heat transfer, mass transfer, and chemical reactions.

It leverages both explicit and implicit temporal schemes. However, based on the motivation above for solving fast-transient problems, we will modify the low-storage multi-stage explicit scheme in its density-based solver. Hence, the objective of this work is to generate optimal stability polynomials for various FVM formulations available in Fluent, and implement them in the multi-stage framework for the density based solver. These will be compared with the default schemes that are currently implemented in Fluent to determine what performance improvements can be obtained relative to the default settings. Ultimately, the utility of these schemes will be explored for the Euler and Navier-Stokes equations through a set of benchmark simulations.

6.3 Fluent's Space-Time Dicretizations Methods

6.3.1 Spatial Discretization

The simulations in this study are performed with the Ansys Fluent Finite-Volume Navier-Stokes solver [46]. Fluent uses a cell-centered scheme, operating on arbitrary grid topologies, including unstructured and polygonal meshes. In the density-based algorithm, the continuity, momentum and energy equations are solved simultaneously in a tightly coupled fashion. For each control-volume, the equations residuals are assembled by a flux summation over all of its faces. The fluxes are evaluated with the Roe [108] or AUSM+ [109] approximate Riemann solvers, responsible of providing the upwinding necessary for stability. Higher-order accuracy is achieved by a MUSCL-reconstruction of the left and right states at each face [110], and monotonicity is enforced with the Barth-Jespersen slope-limiter [111]. The cell-centered solution gradients required for higher-order accuracy are evaluated with an inverse-distance-weighted least-squares formula, operating on a stencil formed by the cell and its immediate face-neighbors. The density-based algorithm is implemented in primitive variables, and equations residuals are thus finalized by multiplying a local conservative-to-primitive transformation matrix.

Fluent's spatial discretization is derived from the divergence theorem applied to multidimensional conservation laws as

$$\frac{d\mathbf{u}}{dt} + \nabla.\mathbf{F}(\mathbf{u}) = 0, \tag{6.1}$$

by integrating

$$\int_V \frac{d\mathbf{u}}{dt} dV + \oint_S \mathbf{F}(\mathbf{u}).\mathbf{n} d\mathbb{S} = 0, \tag{6.2}$$

and discretizing the equation

$$V_i \frac{d\mathbf{u}_i}{dt} + \sum_{j \in \mathbb{S}_i} \hat{\mathbf{F}}(u_i, u_j, \mathbf{n_{i,j}}) S_{i,j} = 0, \tag{6.3}$$

where u_i denotes the cell-averaged conservative variable in the i^{th}-cell, $\mathbf{F}$ the cartesian components of the conserved flux, V_i the volume of the cell, $\mathbb{S}_i$ the set face neighbors to the cell, and $\mathbb{S}_{i,j}$ the area of shared surface and $\mathbf{n_{i,j}}$ is the normal to the shared surface. The symbol $\hat{\mathbf{F}}$ denotes a consistent, conservative and stable numerical approximation to the normal flux, whose construction determining the accuracy of the discretization is discussed hereafter.

The first-order upwind scheme is a straightforward application of Finite-Volume method, and is accessible in Fluent. The flux formulation in Equation 6.3 for the first order upwind can be described as

$$\hat{\mathbf{F}}(u_i, u_j, \mathbf{n_{i,j}}) = \begin{cases} \mathbf{F}(u_i).\mathbf{n} & \frac{d(\mathbf{F}.\mathbf{n})}{du} > 0 \\ \mathbf{F}(u_j).\mathbf{n} & \frac{d(\mathbf{F}.\mathbf{n})}{du} \leq 0. \end{cases}$$

where the state used to evaluate the flux is upwinded according to the sign of the flux jacobian. This scheme is extended to second-order accuracy by linearly-reconstructing the left and right state at flux quadrature points as

$$u_L = u_i + r_L.\nabla u_i, \tag{6.4}$$

and

$$u_R = u_j + r_R.\nabla u_j, \tag{6.5}$$

where r_L and r_R are the direction vectors between the upwind cell centroid and the face centroid. Hence, flux formulation in Equation 6.3 for second order upwind is

$$\hat{\mathbf{F}}(u_i, u_j, \mathbf{n_{i,j}}) = \begin{cases} \mathbf{F}(u_L).\mathbf{n} & \frac{d(\mathbf{F}.\mathbf{n})}{du} > 0 \\ \mathbf{F}(u_R).\mathbf{n} & \frac{d(\mathbf{F}.\mathbf{n})}{du} \leq 0. \end{cases}$$

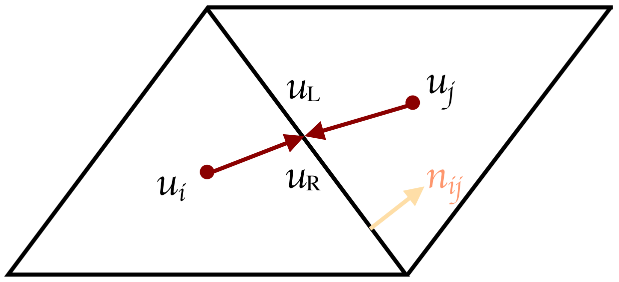


Figure 6.1. Linear reconstruction of the left and right state.

Based on Van Leer's formulation a variety of schemes can be obtained by combining the cell centered and face reconstructed values [110], where left and right states can be obtained by

$$u_L = u_i + \frac{1}{4}\Big((1 - \hat{\kappa})\Delta_i^+ + (1 + \hat{\kappa})\Delta_i^-\Big), \tag{6.6}$$

and

$$u_R = u_j - \frac{1}{4}\Big((1 - \hat{\kappa})\Delta_j^+ + (1 + \hat{\kappa})\Delta_j^-\Big), \tag{6.7}$$

where the difference operators are

$$\Delta_i^+ = \Delta_j^+ = u_j - u_i, \tag{6.8}$$

$$\Delta_i^- = 2r_L.\nabla u_i, \tag{6.9}$$

$$\Delta_i^- = -2r_R.\nabla u_j. \tag{6.10}$$

A particularly favorable choice of combination is $\hat{\kappa} = \frac{1}{3}$, that leads to

$$u_L = \frac{1}{6}(u_i + u_j) + \frac{2}{3}(u_i + r_L.\nabla u_i), \tag{6.11}$$

$$u_R = \frac{1}{6}(u_i + u_j) + \frac{2}{3}(u_j + r_R.\nabla u_j), \tag{6.12}$$

which theoretically provides third order of accuracy in Fluent and is known as the MUSCL scheme. A crucial ingredient of Finite-Volume method is the method of obtaining the reconstruction gradient. The most basic form is the cell based Green-Gauss formula, which approximates the cell gradient as

$$\nabla u_i = \frac{1}{V_i} \sum_{j \in \mathbb{S}_i} 0.5(u_i + u_j) S_{i,j} \mathbf{n_{i,j}}. \tag{6.13}$$

There is also node based Green-Gauss formula available in Fluent, which is based on a contour integral around vertices. Since the averaging used to calculate the vertex solution involves non face neighbors, the node-based scheme is typically the more accurate method. Finally, the most sophisticated method is the cell based least square scheme, where the local solution is expanded in a first order Taylor series as

$$u(x + dx) = u_i + dx\nabla u. \tag{6.14}$$

It is important to note that Roe flux [108] is dissipative. In fact, the Roe flux takes advantage of artificial diffusion to dissipate the solution and converge faster [112]. However, low dissipation Roe fluxes can be defined. This lower dissipation Roe flux can be achieved by taking smaller values for a coefficient α that determine the dissipation in the flux formulation. the default value of $\alpha = 0.2$. However, smaller values such as $\alpha = 0.1$ are often taken for scale resolving simulation. In the current study, considering the importance of scale resolving simulations in industry, we decided to use these low dissipation Roe fluxes with different values of α, along with default Roe flux in our test cases to explore the effects of these fluxes on the stability and computational cost of industry-standard simulations.

6.3.2 Temporal Discretization

Fluent's density-based algorithm is complemented by explicit and implicit iterative solution methods. The explicit procedure uses a low-storage multi-stage Runge-Kutta method based on the work of Rizzi et al. [113] and [114, 115]. Underlying their formulation, from time level t^n to $t^{n+1} = t^n + \Delta t$, an m-stage method takes the form of

$$u^0 = u^n, \tag{6.15}$$

$$u^i = u^n - \alpha_i \Delta t R(u^{i-1}) \qquad i = 1 \ldots m, \tag{6.16}$$

$$u^{n+1} = u^m, \tag{6.17}$$

where u^n and u^{n+1} are the values of the unknown at time level n and $n+1$, and $\alpha_i = \frac{\Delta t_k}{\Delta t}$ is the time-step ratio which relate to the $m-1$ coefficients c_k. The time-step in Equation 6.17 is associated with CFL number according to the following formula

$$\Delta t = \frac{2\ \mathbf{CFL}\ V}{\sum_f \lambda_f^{max} A_f}. \tag{6.18}$$

This is particular to Runge-Kutta schemes that only have one non zero value in the b vector ($b_s = 1$), and only lower-diagonal values in the A matrix, which are then equal to their respective components of the c vector. In fact, a Butcher tableau of Fluent's explicit temporal schemes can be cast as

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} = \begin{array}{c|cccccc} 0 & 0 & & & & & \\ c_2 & a_{2,1} & 0 & & & & \\ c_3 & 0 & a_{3,2} & 0 & & & \\ c_4 & 0 & 0 & a_{4,3} & \ddots & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 & \\ c_s & 0 & 0 & 0 & \ldots & a_{s,(s-1)} & 0 \\ \hline & 0 & 0 & 0 & \ldots & 0 & b_s = 1 \end{array} \tag{6.19}$$

where

$$c_i = a_{i,i-1.} \tag{6.20}$$

Hence, the stability polynomial can be obtained from the Butcher tableau as

$$P_{s,p}(z) = 1 + z + c_s z^2 + c_s c_{s-1} z^3 + c_s c_{s-1} c_{s-2} z^4 + ... + c_s c_{s-1} ... c_2 z^s. \tag{6.21}$$

Since there is only one non zero component in b, it can be proven that these methods can, at most, achieve second-order accuracy for non-linear differential equations [38].

For time-accurate calculations, Fluent uses the following default coefficients $c_i^{\mathrm{t}} = [0.25, 0.3333, 0.5, 1.0]$, which recovers a second-order four stage Runge-Kutta integrator. For steady-state calculations, a three-stage scheme with coefficients $c_i^{\mathrm{ss}} = [0.2075, 0.5915, 1.0]$ is favored. Proposed by Lynn [106], these coefficients have been optimized for efficient multigrid smoothing.

In Equation 6.16, the residual ($R(u^{i-1})$) must be evaluated using an updated value of the integrating variable u and its associated right and left reconstructed values (u_L and u_R) for second order and MUSCL schemes. However, in the current version of Ansys Fluent [46], the value of residuals are evaluated at intermediate stages with updated value of u, but with the values of reconstructed gradients at the first Runge-Kutta stage. In other words, the reconstruction gradient is only evaluated with the value of u at initial time level t^n and it is not updated until the next time level at t^{n+1}. Although, keeping the reconstruction gradient frozen in all intermediate Runge-Kutta schemes reduces computational cost, it potentially increases the error in comparison with updated gradients using the same time-step size. In this study we had access to a special build of Ansys Fluent with an option of switching between updated and frozen gradients. Hence, we will compare the optimal RK schemes using updated gradients with the default frozen gradient scheme.

6.4 Von Neumann Analysis of Fluent Finite Volume Solver

In order to generate the aforementioned optimal stability polynomials of Finite-Volume methods, we perform Von Neumann analysis to obtain the Fourier spectra of the semi-discretization. This analysis consists of evaluating a scheme's wave propagation properties by considering the 1-dimensional linear advection equation

$$\frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} = 0, \tag{6.22}$$

where $u = u(x, t)$ is the scalar solution variable, αu is the linear advection flux with α the advection velocity, t is time and x is the spatial coordinate. We divide the computational domain into N_e elements. Then, we seek plane wave solutions of the form

$$u(x, t) = e^{I(\hat{\kappa}x - \nu t)}, \tag{6.23}$$

where $\hat{\kappa}$ denotes the wavenumber, ν is the frequency of the wave and $I = \sqrt{-1}$ is the imaginary number. We consider $\alpha = 1$ and hence the exact dispersion relation is simply $\nu = \hat{\kappa}$. After a simple projection, we may write for any given element

$$u(x, t) \approx u_i(x, t) = e^{I(\hat{\kappa}x_i - \nu t)}\boldsymbol{U}, \tag{6.24}$$

where $\boldsymbol{U}$ is an unknown that contains the amplitudes of the wave in numerical space, and x_i is the center coordinate of the element. Note that the solution within any given element is the same and hence the choice of the subscript $i \in [1, N_e]$ is arbitrary (assuming periodic boundary condition). After discretizing the linear advection equation using Finite-Volume method with an first-order upwind, second-order upwind and MUSCL flux, we can find the time derivative of the form

$$\frac{d}{dt}u_i(x, t) = \frac{d}{dt}e^{I(\hat{\kappa}x_i - \nu t)}\boldsymbol{U}. \tag{6.25}$$

Considering all wavenumbers defines a spectrum in the complex plane that determines the stability properties of the Finite-Volume method, which are then scaled by the time-step size to fit within the stability region of the explicit schemes. Figure 6.2 displays the spectrum for all considered upwind schemes using the Finite-Volume spatial discretizations for resolvable wavenumbers.

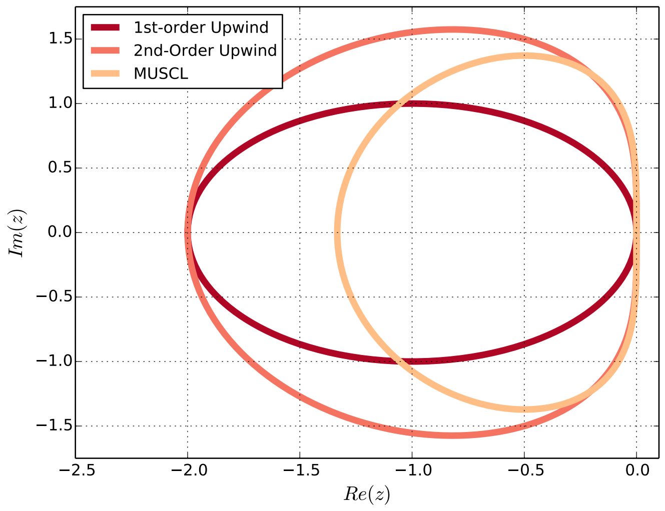


Figure 6.2. Stability spectrum of Finite Volume spatial discretization.

6.5 Optimizing Stability Polynomials

6.5.1 Generation of optimal Polynomials

As mentioned in the Section 3, when there are more RK stages than required to achieve a desired order of accuracy in time, the higher-degree terms in the stability polynomial can be optimized to improve other desirable properties. In other words, in the monomial basis representation $P_{s,p}$ the first $p + 1$ coefficients $(\gamma_0, \gamma_1, ..., \gamma_p)$ are simply taken to satisfy the order conditions [59]. As a results, the space of decision variables has dimension of $s + 1 - p$, and consists of the coefficients $(\gamma_{p+1}, \gamma_{p+2}, ..., \gamma_s)$, as well as the time-step size Δt. In fact, we optimize these unknown coefficients to increase the maximum stable time-step size Δt_{max}. This allows a larger time-step size to be taken for a given number of stages, reducing the total number of iterations and computational cost. Hence, we use Algorithm 1 to generate optimal stability polynomials.

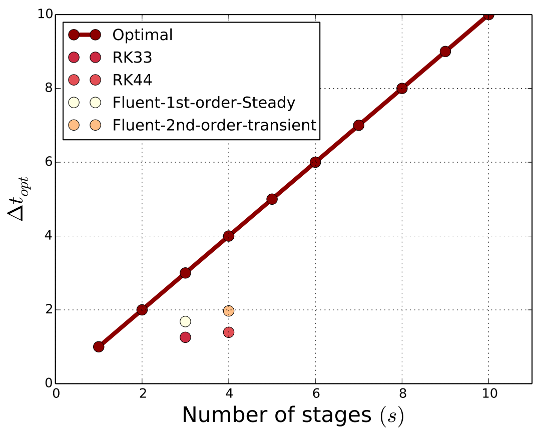
6.5.2 Optimized Stability Polynomials

Using the one dimensional Von Neumann analysis and the optimization method proposed in Section 6.5.1, optimal stability polynomials have been generated for the first-order upwind, second-order upwind, and MUSCL schemes with first- and second-order temporal accuracy with up to 10 stages. Figures 6.3a, 6.3c and 6.3e display plots of the permissible time-step size for optimized first-order accurate temporal schemes, along with the maximum permissible time-step for classical RK schemes and Fluents default three-stage first-order and four-stages second-order scheme. This comparison is shown for the first- and second-order upwind and MUSCL spatial discretizations. Similarly, Figure 6.4a, 6.4c and 6.4e show optimal time-step sizes for the second-order accurate temporal scheme and the maximum time-step size of classical schemes for the first- and second-order upwind and MUSCL schemes. An approximately linear increase in time-step size is observed as the number of stages increases, which is consistent with previous studies [55].
In all cases a larger time-step size can be taken with the optimal schemes in comparison with the classical and default schemes in Fluent. For Example, in Figure 6.4e, the time-step ratio of the optimal four stages second-order scheme and ten stages second-order scheme to Fluent's default temporal scheme is 1.18 and 3.21 respectively. Figures 6.3b, 6.3d, 6.3f, 6.4b, 6.4d and 6.4f demonstrate the relative performance of optimal first- and second-order schemes as well as classical and default schemes in Fluent. It is observed that, increasing the number of stages still continues to improve the performance of the optimal RK methods with the MUSCL scheme beyond the maximum number of stages considered in this study.
In Figure 6.3a, the time-step sizes of first-order multistage Runge-Kutta schemes versus number of stages, which are optimized for the first-order upwind scheme, have been plotted. It is well-known that the first-order Euler method is optimal for a first-order upwind spatial discretization and the improvement in performance of the optimal first order multistage schemes is not expected, as can be observed in Figure 6.3b. However, for second-order upwind and MUSCL schemes performance

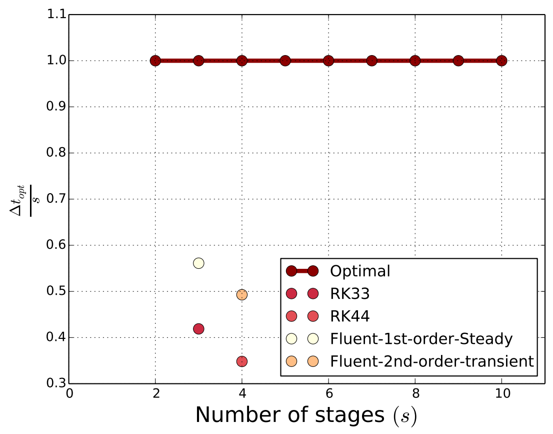
improvements can be seen up to six stages in Figure 6.3d and 6.3f. Figures 6.4b, 6.4d and 6.4f, demonstrate the performance improvement of the optimal second order multistage RK methods. It can be observed that the second-order schemes show improvements for all three discretizations. From Figure 6.4d and 6.4f, it is worth mentioning that, the classical RK_{33} scheme has a maximum time-step close to the second-order three stage optimal RK scheme.

Ultimately, in order to verify our analysis we plotted the footprints of each scheme scaled with Δt_{opt} of optimal four-stage second-order scheme within its absolute stability region. For all three upwind methods, these footprints are contained within their respective regions of absolute stability, which verifies our approach to generate optimal RK multistage stability polynomials for the Finite Volume solver in Fluent.

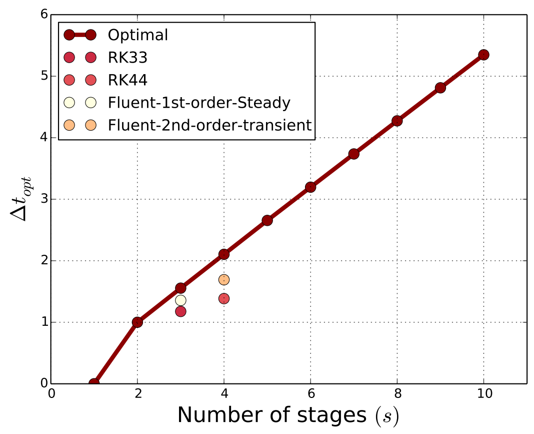
The optimal time-step size, stability polynomials, and **c** vector in multistage method with first-order and second-order accuracy in time for first-order upwind, second-order upwind and the MUSCL scheme can be found in Appendix A.



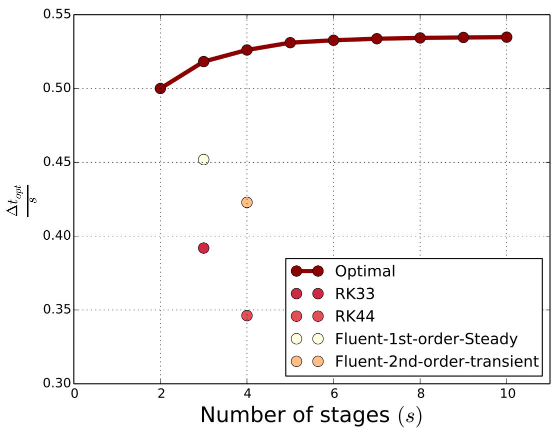
(a) Δt_{opt} for first-order upwind scheme.



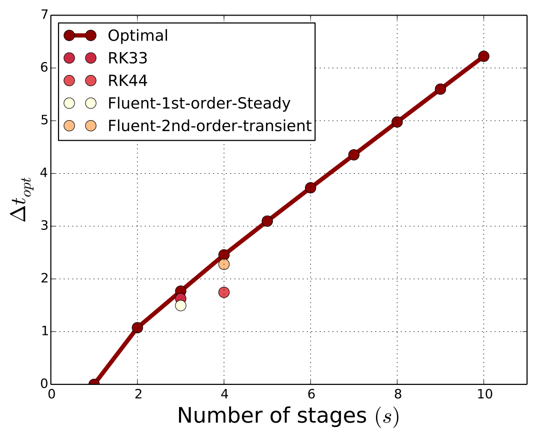
(b) $\Delta t_{opt}/s$ for first-order upwind scheme.



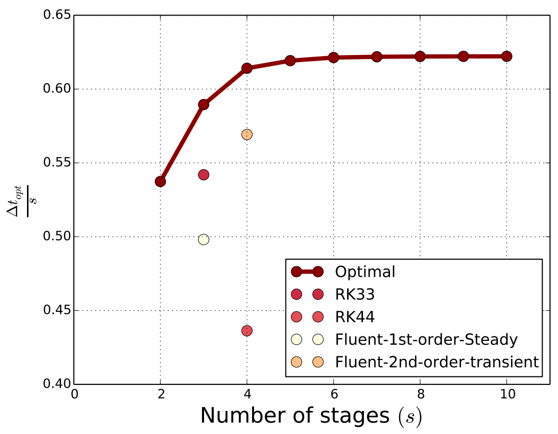
(c) Δt_{opt} for second-order upwind scheme.



(d) $\Delta t_{opt}/s$ for second-order upwind scheme.

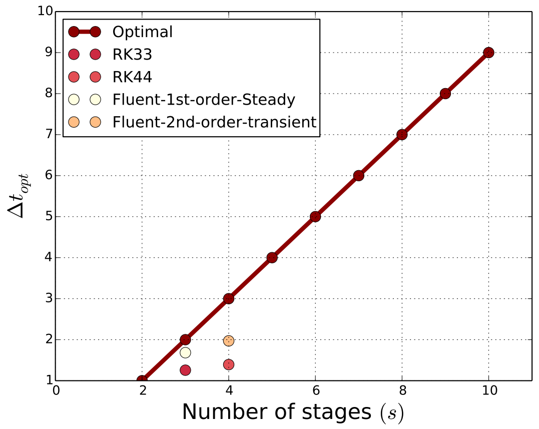


(e) Δt_{opt} for MUSCL scheme.

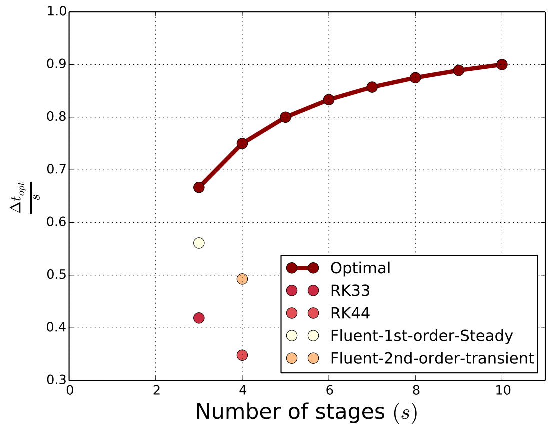


(f) $\Delta t_{opt}/s$ for MUSCL scheme.

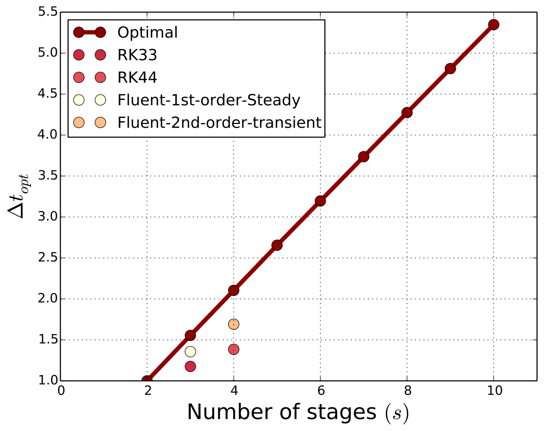
Figure 6.3. Plots of the Δt and $\Delta t_{opt}/s$ for **first**-order RK optimized for Fluent's upwind schemes.



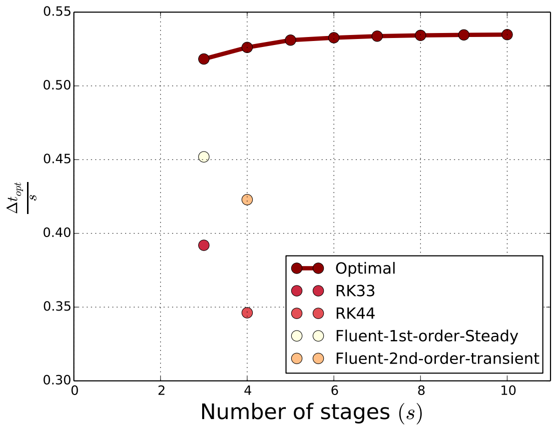
(a) Δt_{opt} for first-order upwind scheme



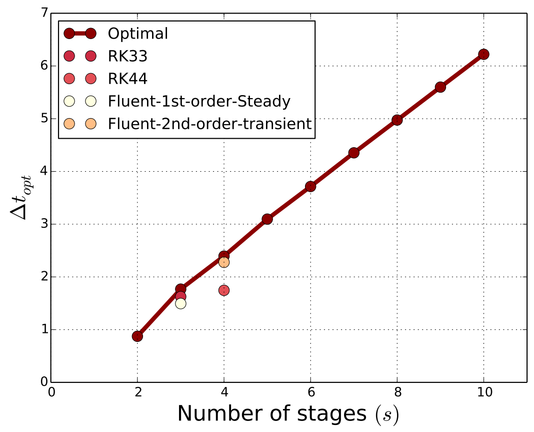
(b) $\Delta t_{opt}/s$ for first-order upwind scheme



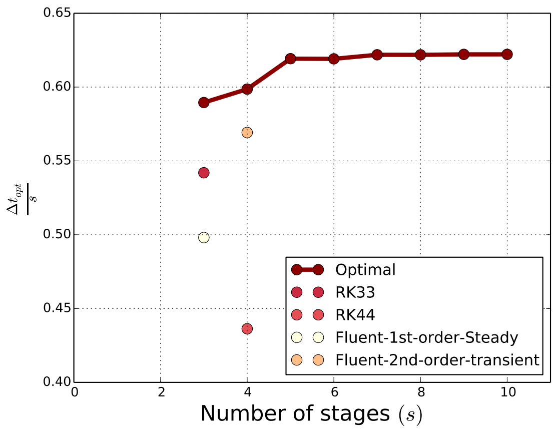
(c) Δt_{opt} for second-order upwind scheme



(d) $\Delta t_{opt}/s$ for second-order upwind scheme.

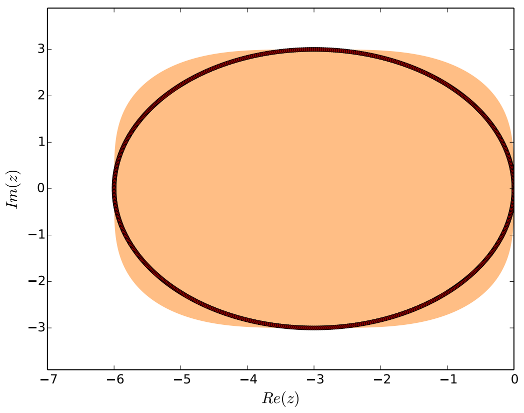


(e) Δt_{opt} for MUSCL scheme.

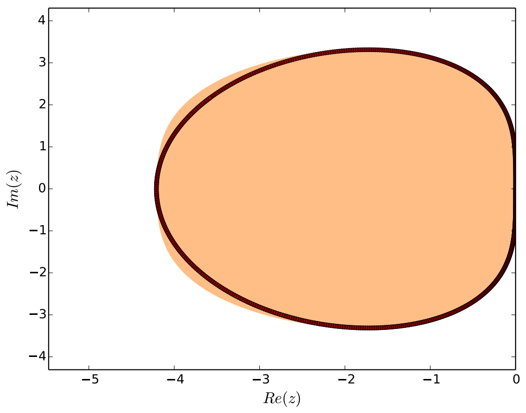


(f) $\Delta t_{opt}/s$ for MUSCL scheme.

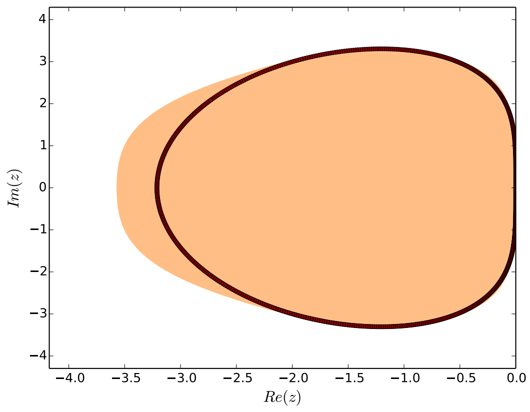
Figure 6.4. Plots of the Δt and $\Delta t_{opt}/s$ for **second**-order RK optimized for Fluent's upwind schemes.



(a) First order upwind.



(b) Second order upwind.



(c) MUSCL scheme.

Figure 6.5. Eigenspectra of Fluent's flux schemes scaled by Δt_{opt} (red) within region of absolute stability of corresponding optimal second-order four stage stability polynomial $P_{4,2_{opt}}$.

6.6 Numerical Results

To demonstrate the utility of theses optimal multistage methods relative to Fluent's default temporal schemes, particularly with the frozen reconstruction gradient, we present a set of benchmark simulations. Since, the MUSCL scheme has the highest theoretical order of accuracy, we decided to run all test cases using this scheme.

6.6.1 Euler Vortex

To explore the accuracy and performance of the optimal schemes we consider an isentropic vortex using the Euler equations on a periodic domain. This problem has an exact analytical solution, making it useful for verification and quantitative error comparisons. The exact solution is simply the propagation of the isentropic vortex with the flow. The initial flow field is specified as

$$\begin{aligned}
\rho &= \left(1 - \frac{S^2 M^2(\gamma - 1)e^{2f}}{8\pi^2}\right)^{\frac{1}{\gamma-1}},\\
u &= \frac{S y e^f}{2\pi r_c},\\
v &= 1 - \frac{S x e^f}{2\pi r_c},\\
P &= \frac{\rho^\gamma}{\gamma M^2},
\end{aligned} \tag{6.26}$$

where ρ is the density, u and v are the velocity components, p is the pressure, $f = \frac{(1-x^2-y^2)}{2r_c^2}$, S=13.5 is the vortex strength, $M = 0.4$ is the free stream Mach number, and $r_c = 1.5$ is vortex radius, and finally, $\gamma = 1.4$ is specific heat ratio . Figure 6.6 displays the initial density contours of the vortex. In order to investigate the effect of freezing or updating the gradient at every Runge-Kutta stage, we ran a set of simulations with 160×160 quadrilateral elements on a 20×20 two-dimensional domain with periodic boundary conditions on all four edges. Different time-step sizes up to the stability limit of each scheme were used for one cycle of the vortex through the domain with Fluent's default four stage second-order temporal scheme and the error was compared for equivalent time step sizes. To evaluate the accuracy of each scheme we considered the L_2 norm of density error at the end of each simulation, defined as

$$\sigma = \sqrt{\int_0^{20}\int_0^{20}\Big(\rho_n(x,y) - \rho_e(x,y)\Big)^2 dxdy}, \tag{6.27}$$

where $\rho_n(x, y)$ is the final numerical solution and $\rho_e(x, y)$ is the exact analytical solution, which is identical to the initial condition. Figure 6.7 demonstrates the error for the frozen and updated gradient configurations of Fluent. As the time-step size decreases the error of simulation with the frozen gradient gets close to the error of the simulation with updated gradients. However, as the time-step size increases towards the stability limit, the error of the simulation with frozen gradients increases rapidly, while the error of the simulation with updated gradients increases approximately linearly, as expected based on the order of the temporal scheme. Hence, increasing the time-step size ultimately pollutes the accuracy of simulations using frozen gradients.

To compare the computational cost of the optimal schemes with Fluent's default scheme with updated and frozen gradients we used three different grid sizes of 120×120, 160×160, and 200×200 with the same initial condition, and ran sets of simulations with frozen and updated gradients in intermediate stages with default four-stage second-order ($s = 4, p = 2$) temporal scheme in fluent, as well as optimal schemes from four-stages up to ten-stages. Each simulation was run to a final time of $t = 20$ with largest stable time-step size. Figure 6.8 demonstrates the error versus average total runtime of all optimal schemes alongside Fluent's default scheme with frozen and updated gradients for each grid size. Although the frozen gradients approach with the default scheme has lower cost in comparison with the schemes with updated gradients, the error is significantly larger. It is also apparent that the optimal scheme with ten stages has significantly lower error while maintaining comparable computational cost to the default scheme with frozen gradients. Hence, by switching from the default RK scheme with frozen gradients to an optimal RK scheme with updated gradients, significantly more accurate results can be obtained without increasing total computational cost.

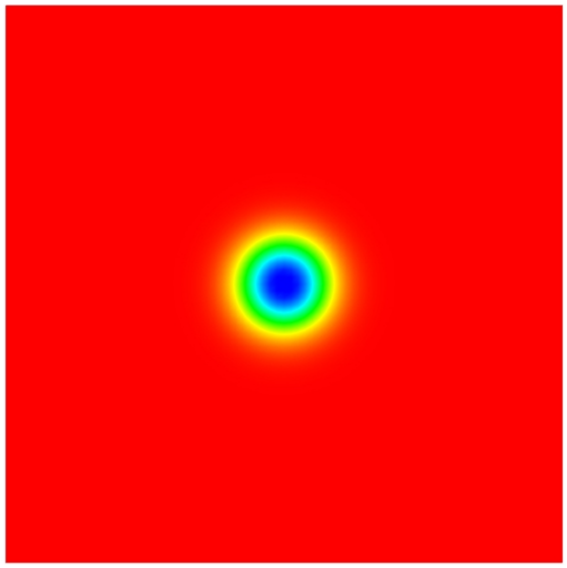


Figure 6.6. Initial Contour of density of the isentropic vortex.

It is also observed that the maximum permissible time step size with the optimal schemes is in line with expectations based on the prior linear analysis. In comparison with a configuration using Fluent's default RK scheme while updating the gradient every RK stage, an approximately 3.12 times larger time-step can be taken using the optimal second-order ten stage scheme, which is close to the theoretical analysis that was shown in Figure 6.4e). This results in an approximately 25% reduction in computational cost by switching to the optimal RK scheme when using updated gradients. Furthermore, if a comparison is made between the optimal second-order ten stages scheme with updated gradients and Fluent's second-order four stages scheme with frozen gradients, an approximately 40 % performance improvement is observed. From Figure 6.8, it is also apparent that approximately the same level of error we can be obtained while using a coarser mesh (130×130) when updating gradients every RK stage, rather than using a relatively fine mesh (200×200) with frozen gradients. This results has a factor of five reduction in total computational cost for the same level of error compared to the default configuration used by Fluent. Figure 6.9 shows contours of density for the isentropic vortex simulation after one vortex cycle with frozen and updated gradients. It is apparent that the error of simulations using updated gradients is significantly lower than that with the frozen gradients.

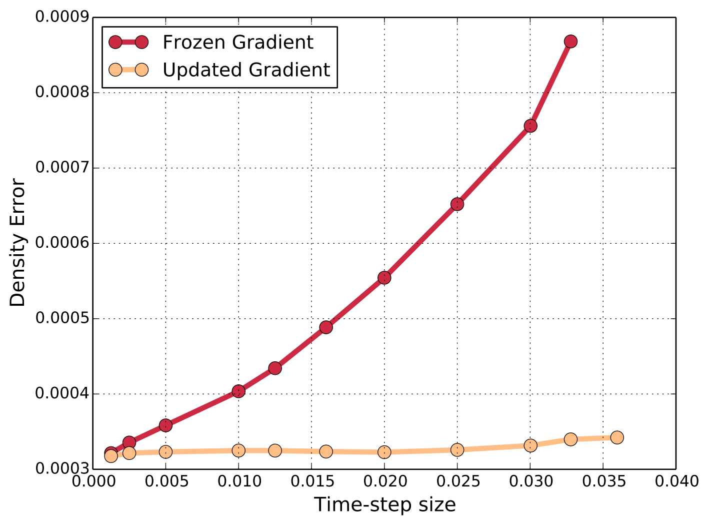


Figure 6.7. Density error versus time-step size for frozen and updated gradient in every stage of RK method.

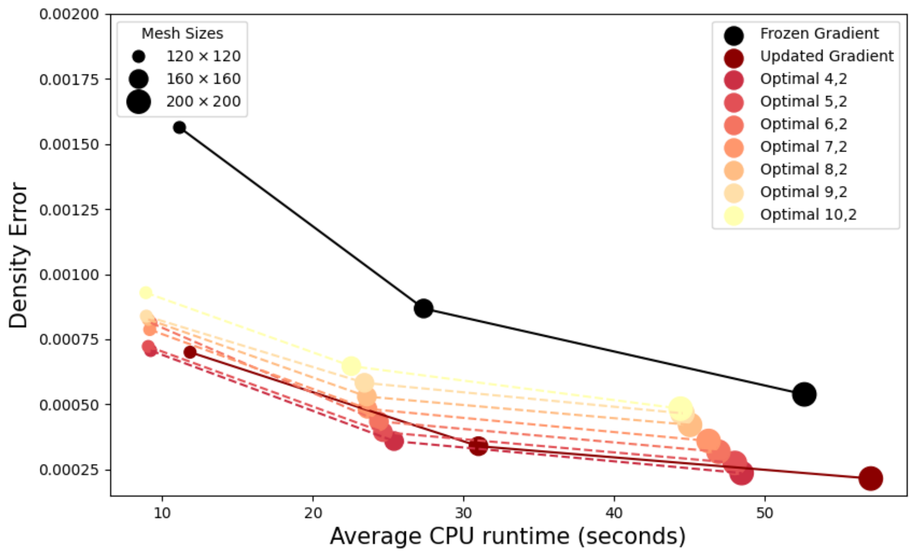
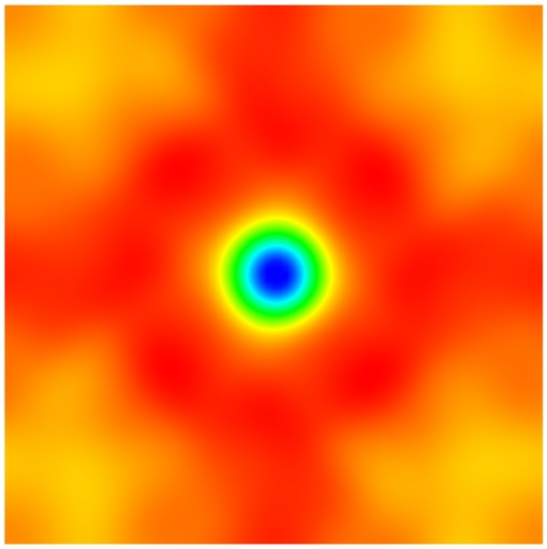


Figure 6.8. Error versus average runtime for optimal schemes with three grid size.


(a) Frozen reconstruction gradient.

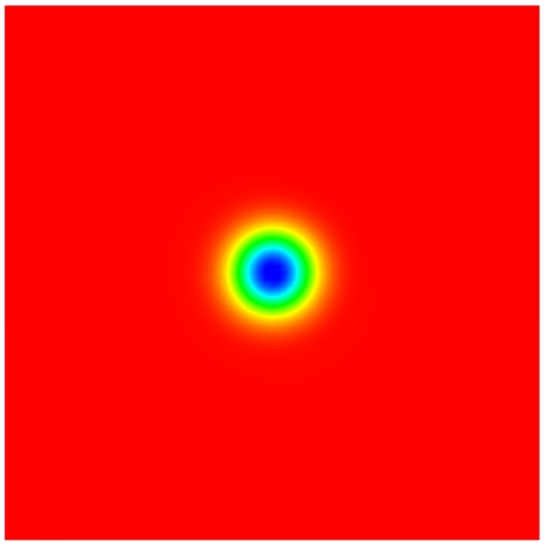

(b) Updated reconstruction gradient.

Figure 6.9. Contours of density at $t = 20$ with Fluent's default scheme.

An additional sets of simulations were conducted with the low diffusion Roe flux [112] using $\alpha = 0.2$ and $\alpha = 0.1$ with the 160×160 mesh. Tables 6.1 and 6.2 show error and total runtime of simulations using the default temporal scheme with updated and frozen gradients and optimal ten stage schemes. It is observed that the stability limits of schemes using updated gradients are nearly identical with the low dissipation Roe fluxes. However, this destroys the stability of the frozen gradient scheme, meaning significantly smaller time step sizes are required with increased total computational cost. Hence, when using a low dissipation Roe flux the optimal RK schemes with updated gradients can be over 17 times faster than when using frozen gradients.

Table 6.1. Density error of isentropic vortex.

Scheme	Frozen Gradient	Updated Gradient	Optimal 10,2
Low Diffusion flux $\alpha = 0.1$	0.0000974569	0.0002299260	0.0004881511
Low Diffusion flux $\alpha = 0.2$	0.0004451824	0.0002748718	0.0005792512
Default Roe Flux	0.0008681105	0.0003397621	0.0006471066

Table 6.2. Total runtime of isentropic vortex in seconds.

Scheme	Frozen Gradient	Updated Gradient	Optimal 10,2
Less diffusion flux $\alpha = 0.1$	396.370	30.104	24.784
Less diffusion flux $\alpha = 0.2$	176.321	30.985	23.014
Default Roe Flux	29.947	29.314	22.547

6.6.2 Taylor-Green Vortex

To explore the utility of the optimized stability polynomials for Navier-Stokes equations we consider the transitional and turbulent Taylor-Green vortex. The initial flow field for this test case is is specified as [116]

$$\begin{aligned}
u_x &= +U_0 \sin(x/L)\cos(y/L)\cos(z/L), \\
u_y &= -U_0 \cos(x/L)\sin(y/L)\cos(z/L), \\
u_z &= 0, \\
P_0 &= \frac{\rho_o U_0^2}{16}\left(\cos(2x/L) + \cos(2y/L)\right)\left(\cos(2z/L) + 2\right), \\
\rho &= \frac{p}{RT_0},
\end{aligned} \tag{6.28}$$

where, u, v, and w are the velocity components, p is the pressure, ρ is the density, R is the gas constant, and T_0 and U_0 are constants specified such that the flow Mach number based on U_0 is $Ma = 0.1$, effectively incompressible. The domain is a periodic cube with the dimensions $0 \leq x, y, z \leq 2\pi L$. For the current study we consider a Reynolds number $Re = 1600$ based on the length scale L and velocity scale U_0. The test case is run to a final non-dimensional time of $t = 20t_c$ where $t_c = \frac{L}{U_0}$, and reference spectral DNS data is available from Van Rees et al. [88]. The temporal evolution of the total kinetic energy, which can be found by

$$E_k = \frac{1}{\rho_0 \Omega}\int_\Omega \rho \frac{\mathbf{u}\cdot\mathbf{u}}{2} d\Omega, \tag{6.29}$$

and, more specifically, its dissipation rate $\epsilon = -\frac{dE_k}{dt}$. Furthermore, the temporal evolution of enstrophy can be computed via

$$\varepsilon = \frac{1}{\rho_0 \Omega} \int_{\Omega} \rho \frac{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}{2} d\Omega, \tag{6.30}$$

where $\boldsymbol{\omega}$ is the vorticity. For incompressible flows the dissipation rate can be related to the enstrophy by $\epsilon = 2\frac{\mu}{\rho_o}\varepsilon$ [88]. Hence, the enstropy is a direct measure of the expected physical dissipation, and the dissipation rate computed from the kinetic energy is typically higher due to the additional numerical dissipation of the scheme.

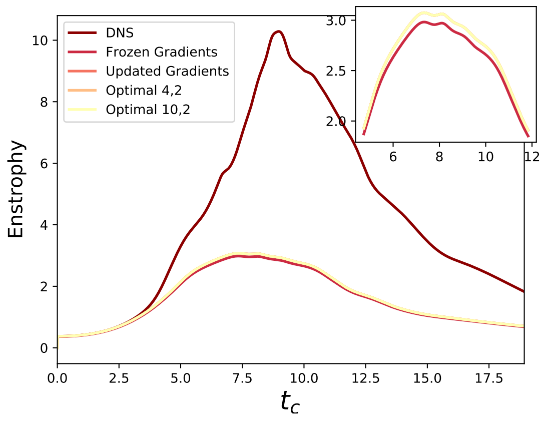
A set of simulations was run with 128^3 degrees of freedom (The DNS resolution was 512^3) using the default scheme with frozen and updated gradients and the the optimal schemes from $s = 4$ to $s = 10$ with updated gradients. We then repeated this procedure with low diffusion Roe fluxes using $\alpha = 0.2$ and $\alpha = 0.1$. Results in Table 6.3 show that schemes with updated reconstruction gradients in intermediate Runge-Kutta stages have identical stability constraints with the low diffusion Roe flux, and we can see improvements in time-step size and ultimately computational cost with number of stages when using the optimal RK schemes. Hence, when using low dissipation Roe fluxes, updating the reconstruction gradient in every intermediate stage is critical in reducing computational cost. Additional speedup is then achieved by using the optimal RK schemes.

Table 6.3. Total runtime of Taylor-Green vortex in seconds.

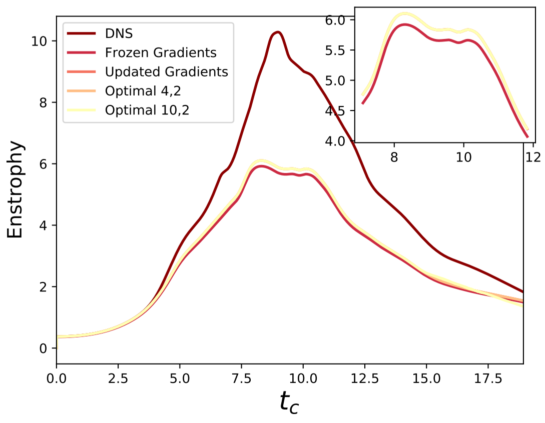
Scheme	Frozen Grad	Updated Grad	Optimal 4,2	Optimal 7,2	Optimal 10,2
Low Diffusion Flux $\alpha = 0.1$	56640.802	7812.429	7425.157	7388.125	7029.475
Low Diffusion Flux $\alpha = 0.2$	31502.704	7764.584	7345.219	7204.018	6907.296
Default Roe Flux	22702.584	7658.245	7228.205	7018.515	6854.254

Figure 6.10 shows the temporal evolution of enstrophy for the frozen and updated reconstruction gradient solutions, as well as those with the optimal four stages and ten stages RK schemes alongside the DNS results [105]. Although simulations with the frozen reconstruction gradient have a smaller time-step size, they are still significantly more dissipative. Meanwhile, simulations with Fluent's default temporal scheme with updated gradients approximately overlaps with the enstrophy plot of

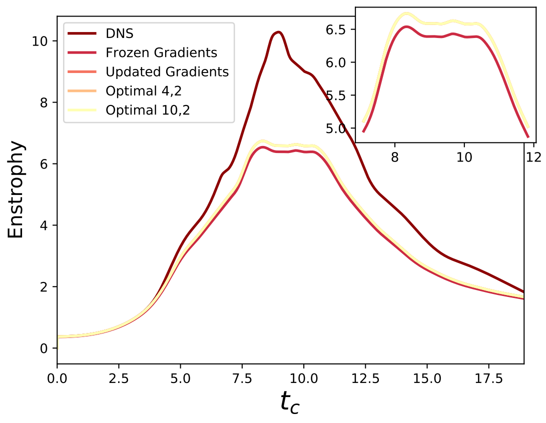
optimal RK schemes, despite the significantly lower computational cost.



(a) Default Roe flux.



(b) Low dissipation flux $\alpha = 0.2$.



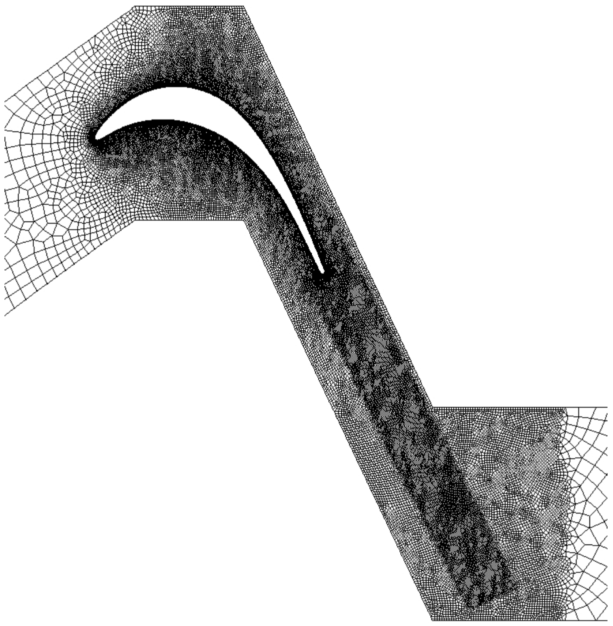
(c) Low dissipation flux $\alpha = 0.1$.

Figure 6.10. Enstrophy versus dimensionless time.

6.6.3 Turbine Blade T106A Cascade

The final test case is a scale-resolved simulation of transitional flow over the T106A low-pressure turbine cascade. This configuration is characterized by a pitch-to-chord ratio of 0.789 and a stagger angle of 30.7° degrees. This case was studied experimentally by Stadtmüller [117], and was the subject of numerous direct numerical simulations, notably by Wissink et al. [118], Sandberg et al. [119] and Michelassi et al. [120, 121]. The flow conditions correspond to a Reynolds number of $Re = 60,000$, an isentropic Mach number at the outlet of $Ma_{2,is} = 0.405$, and an inflow angle of 45.4° degrees. The present computational grid consists of a two-dimensional unstructured mesh, shown in Figure 6.11a, extruded in 40 equally-spaced layers over a spanwise extent of 0.1 chord units, for a total of 2.1 million cells. Total and static pressure boundary conditions are imposed at the inlet and outlet planes, respectively, and conformal periodicity is applied in the pitchwise and spanwise directions. As this case may be prone to unphysical reflections at the inlet boundary [119], a coarse mesh and a reduction to first-order accuracy over a few layers of cells adjacent to the inlet is applied. This treatment acts as a rudimentary sponge-layer, and is sufficient to damp the destabilizing acoustic modes. The computation was carried-out with third-order MUSCL spatial discretization and low-diffusion Roe flux. The flow was established over 10 flow-through periods, and statistics were collected over 8 additional flow-through periods.

Figure 6.11b presents a visualization of the instantaneous flow, obtained by displaying the contours of normalized density-gradient magnitude $||\nabla\rho||/\rho$. In accordance with previous observations [119, 121], the flow is laminar for most of the blade extent, and forms a small separation zone at the trailing-edge, where the flow transitions to the turbulent regime. The time-averaged surface pressure coefficient, defined as $c_p = (p - p_{s,2})/(p_{t,1} - p_{s,2})$, is shown in Figure 6.12a where it is observed to agree well with the experimental data of Stadtmüller [117]. Figure 6.12b shows the time-averaged total-pressure loss coefficient, defined as $\Omega = (p_{t,1} - p_t(y))/(p_{t,1} - p_{s,2})$. The vertical extraction plane is located in the wake, at a distance of 0.4 chord-length past the trailing edge. Once more the agreement with the experimental data is generally satisfactory. Although the current mesh resolution is too coarse to claim DNS accuracy, the present agreement appears sufficient to capture the largest scales of this flow, and we can now turn our attention to the effect of the temporal discretization.


(a) Computational mesh.

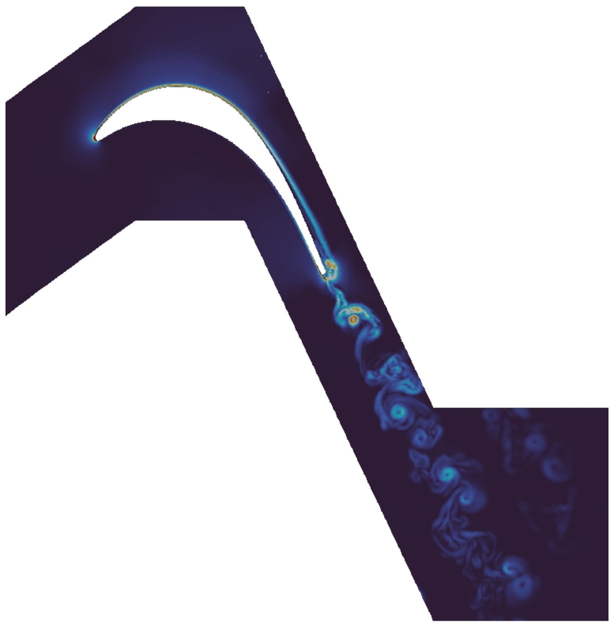
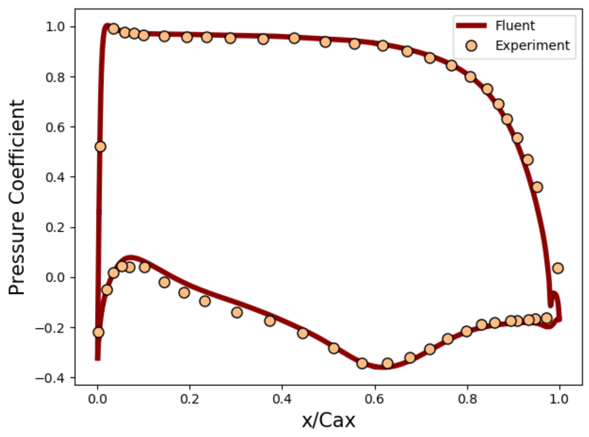
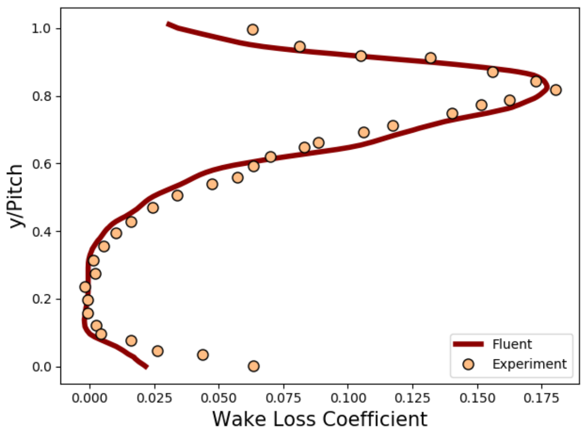

(b) Visualization of the instantaneous flow.

Figure 6.11. T106a blade simulation.



(a) Pressure coefficient distribution.



(b) Total-pressure loss coefficient across the wake plane.

Figure 6.12. T106a time-averaged flow statistics.

The T106a turbine cascade simulation was run with low diffusion Roe fluxes using $\alpha = 0.2$ and $\alpha = 0.1$ for one flow passage over the blade. Tables 6.4 and 6.5 present the time-step size and simulation runtime of one passage of flow over the chord. It is observed that, similar to previous test cases, reducing the diffusion in the Roe flux hinders the stability of simulations with frozen reconstruction gradients. It is also evident that for scale-resolving simulations with low diffusion fluxes, updating the gradient in every stage of the RK scheme is crucial and that cost can be additionally reduced by using the optimal RK schemes.

Table 6.4. Time-step size, performance and one flow passage runtime with low diffusion Roe flux $\alpha = 0.2$.

Scheme	Δt	$\Delta t/s$	Passage Runtime
Frozen Gradients-Default Scheme	0.0002000020000	0.0000500005000	44403.531
Updated Gradients-Default Scheme	0.0006060606061	0.0001515151515	41769.085
10 Stages Optimal Scheme	0.0016806722690	0.0001680672269	36266.199

Table 6.5. Time-step size, performance and one flow passage runtime with low diffusion Roe flux $\alpha = 0.1$.

Scheme	Δt	$\Delta t/s$	Passage Runtime
Frozen Gradients-Default Scheme	0.0001000000000	0.0000250000000	88403.531
Updated Gradients-Default Scheme	0.0006060606061	0.0001515151515	41769.085
10 Stages Optimal Scheme	0.0016806722690	0.0001680112185	35189.992

6.7 Discussion

In this section, we generated optimal Runge-Kutta stability polynomials for Fluent's Finite-Volume schemes having first-, second- and third-order accuracy in space. It has been shown that, by adding more Runge-Kutta stages and terms to the corresponding stability polynomial, larger optimized time steps can be obtained. Notably, the relative efficiency $\frac{\Delta t_{opt}}{s}$ increases significantly with the number of stages for the MUSCL scheme. Furthermore, we observed that although freezing the reconstruction gradients can reduce computational cost per time step, it significantly increases error and hampers stability, limiting time step size. In addition, for scale-resolving simulations, where a low diffusion Roe flux is desirable, it is critical to use updated gradients in every stage of the RK method. This additional cost is more than offset by a significantly larger stable time-step size, upon which further additional speedups are observed by switching to the optimal RK schemes. Finally, results from the Euler vortex, Taylor-Green vortex, and T106A turbine blade demonstrated that combining updated gradients in every stage in conjunction with optimal RK schemes enables a significant reduction in computational cost while improving solution fidelity, particularly with low diffusion Roe fluxes.

Chapter 7

Conclusions and Recommendations for Future Work

7.1 Summary and Conclusions

The objective of this work was to present efficient and accurate explicit temporal schemes for the simulation of transitional and turbulent flows. We generated optimal stability polynomials for multidimensional element types for the discontinuous Galerkin method recovered via the flux reconstruction approach. We explored the performance of optimal schemes compared to classical Runge-Kutta methods, and observed a speed-up factor of up to 1.97. Furthermore, it was shown that using specifically optimized schemes for prismatic, tetrahedral and triangular element types yielded extra performance benefits. We then validated these speed-up factors with a Taylor-Green vortex case.

We introduced families of third-order Paired Explicit Runge-Kutta (P-ERK) schemes. We then formulated a family of third-order P-ERK schemes, and we applied them to several benchmark test cases. We observed that P-ERK schemes achieve a significant speed-up factor compared to classical RK schemes for laminar and turbulent flows over an SD7003 airfoil and a tandem sphere configuration. We verified convergence of third-order P-ERK schemes using an isentropic vortex test case.

Finally, we developed optimal temporal schemes for Fluent's density-based finite volume solver. We have shown that although freezing the reconstruction gradients can reduce computational cost per time step, it significantly increases error and reduces the stable time-step size. We also investigated the performance of the optimal schemes and observed significant speed-up when combined with updated reconstruction gradients at every Runge-Kutta stage. We demonstrated these results using an isentropic vortex case, Taylor-Green vortex and laminar flow over a T106a Blade.

7.2 Recommendations for future work

Overall, we have proposed efficient, accurate, and stable explicit time advancement methods to accelerate the simulation of turbulent flows in conjunction with high-order spatial discretizations. These new approaches significantly accelerate CFD simulations. Based on the presented results, we can make the following recommendations for future studies.

We have demonstrated the utility of optimal explicit schemes for multidimensional element types in conjunction with high-order FR/DG spatial discretizations. It would be helpful to evaluate the performance and accuracy of optimal schemes for multidimensional element types with industrial benchmark test cases. One approach to future studies, would be considering different configuration s of reference elements. It would be useful to examine the limitations of optimal schemes for unstructured grids. Another direction for Future work would be extending optimal multidimensional stability polynomials to multi-step methods. Another interesting study would be implementing these explicit schemes on GPUs and evaluating their performance and speed-up factors once run on these architectures.

We introduced third-order P-ERK schemes and explored their applications to the benchmark test cases. However, an interesting study could be an implementation of P-ERK schemes on GPUs and evaluate their performance and compare them with implicit time marching methods on CPUs. Another interesting study is the expansion of P-ERK schemes to higher-order accuracy or combine them with accelerating methods in space such as polynomial adaptation. Furthermore, combining P-ERK schemes with AIMEX schemes is a promising area for future work.

Finally, in the context of FVM, we have demonstrated the utility of optimal RK schemes for MUSCL spatial discretizations. However, the application of P-ERK schemes in this context is another promising area for future work.

Bibliography

[1] J.S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[2] J.F. Wendt. *Computational fluid dynamics: an introduction*. Springer Science & Business Media, 2008.

[3] M.B. Abbot and D.R. Basco. Computational fluid dynamics. *An introduction for engineers. Harlow Longman*, 1989.

[4] B. Cockburn and C. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, 1998.

[5] Z.J. Wang. Spectral (finite) volume method for conservation laws on unstructured grids. basic formulation. *Journal of computational physics*, 178(1):210–251, 2002.

[6] H.T. Huynh. A Flux Reconstruction approach to high-order schemes including discontinuous Galerkin methods. *18th AIAA Computational Fluid Dynamics Conference*, 2007.

[7] H.T. Huynh. A reconstruction approach to high-order schemnes including discontinuous Galerkin for diffusion. In *47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, page 403, 2009.

[8] J.A. Ekaterinaris. High-order accurate, low numerical diffusion methods for aerodynamics. *Progress in Aerospace Sciences*, 41(3-4):192–300, 2005.

[9] B. Vermeire. Adaptive implicit-explicit time integration and high-order unstructured methods for implicit large eddy simulation. *PhD dissertation at McGill University*, 2014.

[10] P.E. Vincent, P. Castonguay, and A. Jameson. Insights from Von Neumann analysis of high-order Flux Reconstruction schemes. *Journal of Computational Physics*, 2011.

[11] P. Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media, 2006.

[12] F.M. White. Fluid mechanics, 1999. *Me Graw-Hill*, 1979.

[13] F.M. White and J. Majdalani. *Viscous fluid flow*, volume 3. McGraw-Hill New York, 2006.

[14] D.C. Wilcox. *Turbulence modeling for CFD*, volume 2. DCW industries La Canada, CA, 1998.

[15] A.N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Cr Acad. Sci. USSR*, 30:301–305, 1941.

[16] P.A. Davidson. *Turbulence: an introduction for scientists and engineers*. Oxford university press, 2015.

[17] M. Lesieur. *Turbulence in Fluids*, volume 40. Springer Science & Business Media, 2012.

[18] P.R. Spalart and S.R. Allmaras. A one equation turbulence model for aerodynamic flows. *Aerospace Research Journal*, pages 5–52, 1994.

[19] Joseph Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.

[20] E. Garnier, N. Adams, and P. Sagaut. *Large eddy simulation for compressible flows*. Springer Science & Business Media, 2009.

[21] C. Meneveau and J. Katz. Scale-invariance and turbulence models for large-eddy simulation. *Annual Review of Fluid Mechanics*, 32(1):1–32, 2000.

[22] P. Fernández. *High-order implicit large-eddy simulation for transitional aerodynamics flows*. PhD thesis, Massachusetts Institute of Technology, 2016.

[23] J.P. Boris. On large eddy simulation using subgrid turbulence models comment 1. In *Whither turbulence? Turbulence at the crossroads*, pages 344–353. Springer, 1990.

[24] P.G. Tucker and S. Lardeau. Applied large eddy simulation, 2009.

[25] C. Fureby and F.F. Grinstein. Large eddy simulation of high-Reynolds-number free and wall-bounded flows. *Journal of Computational Physics*, 181(1):68–97, 2002.

[26] D. Drikakis, M. Hahn, A. Mosedale, and B. Thornber. Large eddy simulation using high-resolution and high-order methods. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1899):2985–2997, 2009.

[27] K. Ritos, I. W Kokkinakis, and D. Drikakis. Performance of high-order implicit large eddy simulations. *Computers & Fluids*, 173:307–312, 2018.

[28] M. Galbraith and M. Visbal. Implicit large eddy simulation of low Reynolds number flow past the SD7003 airfoil. In *46th AIAA aerospace sciences meeting and exhibit*, page 225, 2008.

[29] F.F. Grinstein and C.R. DeVore. Dynamics of coherent structures and transition to turbulence in free square jets. *Physics of Fluids*, 8(5):1237–1251, 1996.

[30] M.R. Visbal, R.E. Gordnier, and M.C. Galbraith. High-fidelity simulations of moving and flexible airfoils at low Reynolds numbers. In *Animal Locomotion*, pages 341–360. Springer, 2010.

[31] C. Fureby, G. Tabor, H.G. Weller, and A.D. Gosman. A comparative study of subgrid scale models in homogeneous isotropic turbulence. *Physics of fluids*, 9(5):1416–1429, 1997.

[32] C. Fureby and F.F. Grinstein. Large eddy simulation of high-reynolds-number free and wall-bounded flows. *Journal of Computational Physics*, 181(1):68–97, 2002.

[33] S.M. Murman, L. Diosady, A. Garai, and M. Ceze. A space-time discontinuous-galerkin approach for separated flows. In *54th AIAA Aerospace Sciences Meeting*, page 1059, 2016.

[34] A. Uranga, P.O. Persson, M. Drela, and J. Peraire. Implicit large eddy simulation of transition to turbulence at low reynolds numbers using a discontinuous galerkin method. *International Journal for Numerical Methods in Engineering*, 87(1-5):232–261, 2011.

[35] Y. Zhou and Z.J. Wang. Implicit large eddy simulation of low reynolds number transitional flow over a wing using high-order spectral difference method. In *40th fluid dynamics conference and exhibit*, page 4442, 2010.

[36] J.S. Park, F.D. Witherden, and P.E. Vincent. High-order implicit large-eddy simulations of flow over a naca0021 aerofoil. *AIAA Journal*, 55(7):2186–2197, 2017.

[37] B.C. Vermeire and P.E. Vincent. On the behaviour of fully-discrete Flux Reconstruction schemes. *Computer Methods in Applied Mechanics and Engineering*, 315:1053–1079, 2017.

[38] J.C. Butcher. *Numerical methods for ordinary differential equations*. John Willy and Sons, Ltd, 2002.

[39] G. Wanner and E. Hairer. *Solving ordinary differential equations II: stiff and differential-algebraic problems*. Springer Verlag, 1996.

[40] H. Schlichting and J. Kestin. *Boundary layer theory*, volume 121. Springer, 1961.

[41] F. Moukalled, L. Mangani, M. Darwish, et al. *The finite volume method in computational fluid dynamics*, volume 113. Springer, 2016.

[42] B. Cockburn and C.W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. ii. general framework. *Mathematics of computation*, 52(186):411–435, 1989.

[43] B. Cockburn, S.Y. Lin, and C.W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws iii: one-dimensional systems. *Journal of Computational Physics*, 84(1):90–113, 1989.

[44] B. Cockburn, S. Hou, and C.W. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. iv. the multidimensional case. *Mathematics of Computation*, 54(190):545–581, 1990.

[45] Y. Liu, M. Vinokur, and Z.J. Wang. Discontinuous spectral difference method for conservation laws on unstructured grids. In *Computational fluid dynamics 2004*, pages 449–454. Springer, 2006.

[46] ANSYS Fluent user's guide, 2020R1, 2020.

[47] C. Hirsch. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.

[48] Y.A. Çengel, R.H. Turner, J.M. Cimbala, and M. Kanoglu. *Fundamentals of thermal-fluid sciences*, volume 703. McGraw-Hill New York, 2008.

[49] L. Euler. Principes généraux de l'état d'équilibre des fluides. *Mémoires de l'académie des sciences de Berlin*, pages 217–273, 1757.

[50] Z.J. Wang and H.T. Huynh. A review of flux reconstruction or correction procedure via reconstruction method for the Navier-Stokes equations. *Mechanical Engineering Reviews*, 3(1):15–00475, 2016.

[51] D.M. Williams, P. Castonguay, P.E. Vincent, and A. Jameson. Energy stable flux reconstruction schemes for advection–diffusion problems on triangles. *Journal of Computational Physics*, 250:53–76, 2013.

[52] T. Haga, K. Sawada, and Z.J Wang. An implicit LU-SGS scheme for the spectral volume method on unstructured tetrahedral grids. *Comunications in Computational Physics*, 2009.

[53] T. Haga, H. Gao, and Z.J. Wang. A high-order unifying discontinuous formulation for the Navier-Stokes equations on 3D mixed grids. *Computational Aerodynamics*, 2011.

[54] A. Kanevsky, M.H. Carpenter, D. Gottlieb, and J.S. Hesthaven. Application of implicit-explicit high-order Runge-Kutta methods to discontinuous Galerkin schemes. *Journal of Computational Physics*, August 2007.

[55] E.J. Kubatko, B.A. Yeager, and D.I. Ketcheson. Optimal Strong Stability Preserving Runge–Kutta time discretizations for discontinuous Galerkin methods. *Journal of Computational Science*, 2014.

[56] F.D. Witherden, A.M. Farrington, and P.E. Vincent. Pyfr: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach. *Computer Physics Communications*, 185(11):3028–3040, 2014.

[57] M.R. López, A. Sheshadri, J.R. Bull, T.D. Economon, J. Romero, J.E. Watkins, D.M. Williams, F. Palacios, A. Jameson, and D.E. Manosalvas. Verification and validation of hifiles: a high-order les unstructured solver on multi-gpu platforms. In *32nd AIAA applied aerodynamics conference*, page 3168, 2014.

[58] F. Hindenlang, G.J. Gassner, C. Altmann, A. Beck, M. Staudenmaier, and C. Munz. Explicit discontinuous Galerkin methods for unsteady problems. *Computers & Fluids*, 61:86–93, 2012.

[59] D.I. Ketcheson and A.J. Ahmadia. Optimal stability polynomials for numerical intergration of initial value problems. *Communications in Applied Mathematics and Computational Science*, 7(02), 2012.

[60] M.J. Zahr and P. Persson. Performance tuning of Newton-GMRES methods for discontinuous galerkin discretizations of the navier-stokes equations. In *21st AIAA Computational Fluid Dynamics Conference*, page 2685, 2013.

[61] R. Alexander. Diagonally implicit Runge–Kutta methods for stiff ODE's. *SIAM Journal on Numerical Analysis*, 14(6):1006–1021, 1977.

[62] P. Persson and J. Peraire. An efficient low memory implicit dg algorithm for time dependent problems. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 113, 2006.

[63] S. Rothe, A. Hamkar, K.J. Quint, and S. Hartmann. Comparison of diagonal-implicit, linear-implicit and half-explicit Runge–Kutta methods in non-linear finite element analyses. *Archive of Applied Mechanics*, 82(8):1057–1074, 2012.

[64] C.C. De Wiart, K. Hillewaert, L. Bricteux, and G. Winckelmans. Implicit LES of free and wall-bounded turbulent flows based on the discontinuous Galerkin/symmetric interior penalty method. *International Journal for Numerical Methods in Fluids*, 78(6):335–354, 2015.

[65] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, November 1997.

[66] B.C. Vermeire and S. Nadarajah. Adaptive IMEX schemes for high-order unstructured methods. *Journal of Computational Physics*, 2015.

[67] B.C. Vermeire and S. Hedayati Nasab. Accelerated implicit-explicit Runge-Kutta schemes for locally stiff systems. *Journal of Computational Physics*, 2020.

[68] L. Pareschi and G. Russo. Implicit-explicit Runge-Kutta schemes for stiff systems of differential equations. *Recent trends in numerical analysis*, 3:269–289, 2000.

[69] P. Vincent, F. Witherden, B. Vermeire, J.S. Park, and A. Iyer. Towards green aviation with python at petascale. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2016.

[70] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations I: Nonstiff problems*. Springer Verlag, 1993.

[71] P.J. van Der Houwen and B.P. Sommeijer. On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values. *ZAMM-Journal of Applied Mathematics and Mechanics*, 60(10):479–485, 1980.

[72] S.J. Ruuth. Global optimization of explicit Strong Stability Preserving Runge-Kutta methods. *Mathematics of Computation*, 75(253):183–207, 2006.

[73] S.J. Ruuth and R.J. Spiteri. High-order Strong Stability Preserving Runge-Kutta methods with downwind-biased spatial discretizations. *SIAM Journal on Numerical Analysis*, 42(3):974–996, 2004.

[74] M. Parsani, D.I. Ketcheson, and W. Deconinck. Optimized explicit Runge-Kutta schemes for the spectral difference method applied to wave propagation problems. *SIAM Journal on Scientific Computing*, 35(2):A957–A986, 2013.

[75] B.C. Vermeire, N.A. Loppi, and P.E. Vincent. Optimal Runge–Kutta schemes for pseudo time-stepping with high-order unstructured methods. *Journal of Computational Physics*, 383:55–71, 2019.

[76] M. Shoeybi, M. Svärd, F.E. Ham, and P. Moin. An adaptive implicit-explicit scheme for the DNS and LES of compressible flows on unstructured grids. *Journal of Computational Physics*, August 2010.

[77] W. Hundsdorfer and J. Jaffré. Implicit-explicit time stepping with spatial discontinuous finite elements. *Applied Numerical Mathematicss*, 2003.

[78] R.C. Moura, S.J. Sherwin, and J. Peiró. Linear dispersion–diffusion analysis and its application to under-resolved turbulence simulations using discontinuous Galerkin spectral/hp methods. *Journal of Computational Physics*, 298:695–710, 2015.

[79] K. Van den Abeele. Development of high-order accurate schemes for unstructured grids. *PhD Thesis at Vrije Universiteit Brussel, Belgium*, 2009.

[80] H. Yang, F. Li, and J. Qiu. Dispersion and dissipation errors of two fully discrete discontinuous Galerkin methods. *Journal of Scientific Computing*, 55(3):552–574, 2013.

[81] B.C Vermeire and P.E. Vincent. On the properties of energy stable Flux Reconstruction schemes for implicit large eddy simulation. *Journal of Computational Physics*, 327:368–388, 2016.

[82] C.A. Pereira and B.C. Vermeire. Fully-discrete analysis of high-order spatial discretizations with optimal explicit Runge-Kutta methods. *Journal of Scientific Computing*, 83(3):1–35, 2020.

[83] P. Castonguay. High-order energy stable Flux Reconstruction schemes for fluid flow simulations on unstructured grids. *PhD dissertation at Stanford University*, 2012.

[84] B.C Vermeire. Paired explicit Runge-Kutta schemes for stiff systems of equations. *Journal of Computational Physics*, 393:465–483, 2019.

[85] Matlab 2016a user's guide, 2016.

[86] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, 2013.

[87] Z.J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, and H.T. Huynh. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

[88] W.M. Van Rees, A. Leonard, D.I. Pullin, and P. Koumoutsakos. A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high reynolds numbers. *Journal of Computational Physics*, 230(8):2794–2805, 2011.

[89] H.J. Stetter. *Analysis of discretization methods for ordinary differential equations*, volume 23. Springer, 1973.

[90] S. Hedayati Nasab, C.A. Pereira, and B.C. Vermeire. Optimal Runge-Kutta stability polynomials for multidimensional high-order methods. *Journal of Scientific Computing*, 89:11, 2021.

[91] E.M. Constantinescu and A. Sandu. Multirate timestepping methods for hyperbolic conservation laws. *Journal of Scientific Computing*, 33(3):239–278, 2007.

[92] B. Seny, J. Lambrechts, R. Comblen, V. Legat, and J.F. Remacle. Multirate time stepping for accelerating explicit discontinuous Galerkin computations with application to geophysical flows. *International Journal for Numerical Methods in Fluids*, 71(1):41–64, 2013.

[93] M. Schlegel, O. Knoth, M. Arnold, and R. Wolke. Multirate Runge-Kutta schemes for advection equations. *Journal of Computational and Applied Mathematics*, 226(2):345–357, 2009.

[94] S. Hedayati Nasab and B.C Vermeire. Paired explicit Runge-Kutta method for stiff systems of equations. *Under Review*, 2021.

[95] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier–stokes equations. *Journal of computational physics*, 131(2):267–279, 1997.

[96] M.R. López, A. Sheshadri, J.R. Bull, T.D. Economon, J. Romero, J.E. Watkins, D.M. Williams, F. Palacios, A. Jameson, and D.E. Manosalvas. Verification and validation of HiFiLES: a high-order les unstructured solver on multi-GPU platforms. In *32nd AIAA Applied Aerodynamics Conference*, page 3168, 2014.

[97] D.J. Garmann, M.R. Visbal, and P.D. Orkwis. Comparative study of implicit and subgrid-scale model large-eddy simulation techniques for low-Reynolds number airfoil applications. *International Journal for Numerical Methods in Fluids*, 71(12):1546–1565, 2013.

[98] A.D. Beck, T. Bolemann, D. Flad, H. Frank, G.J. Gassner, F. Hindenlang, and C.D. Munz. High-order discontinuous Galerkin spectral element methods for transitional and turbulent

flow simulations. *International Journal for Numerical Methods in Fluids*, 76(8):522–548, 2014.

[99] B.C. Vermeire, F.D. Witherden, and P.E. Vincent. On the utility of GPU accelerated high-order methods for unsteady flow simulations: A comparison with industry-standard tools. *Journal of Computational Physics*, 334:497–521, 2017.

[100] L. Martinelli and M. Lohry. Implicit time integration of discontinuous Galerkin approximations to the Navier-Stokes equations. In *AIAA Scitech 2020 Forum*, page 0774, 2020.

[101] K.R. Holst, R.S. Glasby, J.T. Erwin, D.L. Stefanski, and R.B. Bond. High-order large eddy simulation validation in HPCMP CREATE (TM)-AV kestrel component coffe. In *2018 AIAA Aerospace Sciences Meeting*, page 0781, 2018.

[102] Y. Xue and Z. Wang. Tandem spheres presentation. In *5th International Workshop on High-Order CFD Methods*, 2018.

[103] J. Jansson, E. Krishnasamy, and M. Leoni. Adaptive direct FEM simulation with unicorn/FEniCS-HPC for cs1. In *5th International Workshop on High Order CFD Methods*, 2018.

[104] O. Lehmkuhl, I. Rodríguez, R. Borrell, and A. Oliva. Low-frequency unsteadiness in the vortex formation region of a circular cylinder. *Physics of Fluids*, 25(8):085109, 2013.

[105] B. Van Leer, P.T. Lee, P.L. Roe, K.G. Powell, and C.H. Tai. Design of optimally smoothing multistage schemes for the Euler equations. *Communications in applied numerical methods*, 8(10):761–769, 1992.

[106] J.F. Lynn. *Multigrid solution of the Euler equations with local preconditioning*. PhD thesis, University of Michigan, 1995.

[107] S. Hedayati Nasab, J.S. Cagnone, and B.C Vermeire. Optimal explicit Runge-Kutta time stepping for density-based finite volume solver. *Under Review*, 2021.

[108] P.L. Roe. Characteristic based schemes for the Euler equations. *Annual review of fluid mechanics*, 18:337–365, 1996.

[109] M.S. Liu. A sequel to AUSM: AUSM+. *Journal of computational physics*, 129:364–382, 1996.

[110] B. Van Leer. Toward the ultimate conservative difference scheme IV. a second order sequel to Godunov's method. *Journal of Computational Physics*, 32:101–136, 1979.

[111] T.J. Barth and D. Jespersen. The design and application of upwind schemes on unstructured meshes. AIAA 27th Aerospace Sciences Meeting, Reno, Nevada, 1989. AIAA-89-0366.

[112] C.M. Winkler, A.J. Dorgan, and M. Mani. A reduced dissipation approach for unsteady flows on unstructured grids. In *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 570, 2012.

[113] A. Rizzi and L.E. Eriksson. Computation of flow around wings based on the euler equations. *Journal of Fluid Mechanics*, 148:45–71, 1984.

[114] A. Jameson. Numerical solution of the Euler equations for compressible inviscid fluids. In *Numerical methods for the Euler equations of Fluid Dynamics*, volume 21, pages 199–245. SIAM Philadelphia, 1985.

[115] A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the euler equations by finite volume methods using Runge-Kutta time stepping schemes. In *14th fluid and plasma dynamics conference*, page 1259, 1981.

[116] J. DeBonis. Solutions of the taylor-green vortex problem using high-resolution explicit finite difference methods. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 382, 2013.

[117] M.P. Stadtmuller. Investigation of wake-induced transition on the lp turbine cascade t106. In *A-EIZ. Tech. rep., University of the Armed Forces, Munich*, 2001.

[118] J.G. Wissink and W. Rodi. DNS of a laminar separation bubble in the presence of oscillating external flow. *Flow, Turbulence and Combustion*, 71:311–331, 2003.

[119] R.D. Sandberg, R. Pichler, and L. Chen. Assessing the sensitivity of turbine cascade flow to inflow disturbances using direct numerical simulation. In *13th International Symposium for Unsteady Aerodynamics, Aeroacoustics and Aeroelasticity in Turbomachinery (ISUAAAT)*, September 2012.

[120] V. Michelassi, J. Wissink, and W. Rodi. Analysis of DNS and LES of flow in a low pressure turbine cascade with incoming wakes and comparison with experiments. *Flow, turbulence and combustion*, 69(3-4):295–329, 2002.

[121] V. Michelassi, L.W. Chen, R. Pichler, and R.D. Sandberg. Compressible Direct Numerical Simulation of Low-Pressure Turbines—Part II: Effect of Inflow Disturbances. *Journal of Turbomachinery*, 137(7), 07 2015. 071005.

Appendix A

Optimal Stability Polynomials for Fluent Finite Volume Density-Based Solver

In this section, we present optimal first and second-order accurate stability polynomials for first-order upwind, second-order upwind and MUSCLE schemes in Fluent density-based finite volume solver.

A.1 First-Order Optimal Stability Polynomials for First-Order Upwind Scheme

1	1	0.250000								
1	1	0.333333	0.037037							
1	1	0.374999	0.062499	0.003906						
1	1	3.999999e-01	7.999999e-02	7.999999e-03	3.199999e-04					
1	1	4.166666e-01	9.259259e-02	1.157407e-02	7.716049e-04	2.143347e-05				
1	1	4.285714e-01	1.020408e-01	1.457726e-02	1.249479e-03	5.949902e-05	1.214266e-06			
1	1	4.375000e-01	1.093750e-01	1.708984e-02	1.708984e-03	1.068115e-04	3.814697e-06	5.960464e-08		
1	1	4.444444e-01	1.152263e-01	1.920439e-02	2.133821e-03	1.580608e-04	7.526706e-06	2.090752e-07	2.581175e-09	
1	1	4.500000e-01	1.200000e-01	2.100000e-02	2.520000e-03	2.100000e-04	1.200000e-05	4.500000e-07	1.000000e-08	1.000000e-10

A.2 Second-Order Optimal Stability Polynomials for First-Order Upwind Scheme

1	1	0.500000	0.083348							
1	1	0.500000	0.111126	0.009262						
1	1	0.500000	1.249981e-01	1.562448e-02	7.812179e-04					
1	1	0.500000	1.333157e-01	1.999468e-02	1.599505e-03	5.331936e-05				
1	1	0.500000	1.388624e-01	2.313927e-02	2.313695e-03	1.285375e-04	3.060538e-06			
1	1	0.500000	1.428931e-01	2.552301e-02	2.917564e-03	2.084382e-04	8.509147e-06	1.519724e-07		
1	1	0.500000	1.457948e-01	2.732925e-02	3.415492e-03	2.845893e-04	1.524494e-05	4.764022e-07	6.616919e-09	
1	1	0.500000	1.481057e-01	2.879005e-02	3.837820e-03	3.552979e-04	2.255636e-05	9.398021e-07	2.320480e-08	2.578371e-10

A.3 First-Order Optimal Stability Polynomials for Second-Order Upwind Scheme

1	1	0.499999								
1	1	0.499998	0.104150							
1	1	0.499999	0.130126	0.016104						
1	1	0.499999	0.142916	0.023734	0.001874					
1	1	4.999996e-01	1.501834e-01	2.856443e-02	3.265650e-03	1.810352e-04				
1	1	4.999997e-01	1.544617e-01	3.165613e-02	4.293176e-03	3.594076e-04	1.466797e-05			
1	1	4.999998e-01	1.572890e-01	3.378069e-02	5.052434e-03	5.148954e-04	3.291545e-05	1.030603e-06		
1	1	4.999998e-01	1.592307e-01	3.529496e-02	5.622192e-03	6.432810e-04	5.111416e-05	2.576811e-06	6.380435e-08	
1	1	4.999999e-01	1.606281e-01	3.641321e-02	6.058689e-03	7.481602e-04	6.774607e-05	4.318118e-06	1.760541e-07	3.532975e-09

A.4 Second-Order Optimal Stability Polynomials for Second-Order Upwind Scheme

1	1	0.500000	0.104151							
1	1	0.500000	0.130126	0.016104						
1	1	0.500000	0.142916	0.023734	0.001874					
1	1	0.500000	1.501834e-01	2.856443e-02	3.265644e-03	1.810341e-04				
1	1	0.500000	1.544619e-01	3.165616e-02	4.293181e-03	3.594081e-04	1.466797e-05			
1	1	0.500000	1.572891e-01	3.378072e-02	5.052441e-03	5.148962e-04	3.291551e-05	1.030604e-06		
1	1	0.500000	1.592308e-01	3.529498e-02	5.622197e-03	6.432817e-04	5.111422e-05	2.576813e-06	6.380439e-08	
1	1	0.500000	1.606282e-01	3.641324e-02	6.058694e-03	7.481610e-04	6.774616e-05	4.318124e-06	1.760544e-07	3.532982e-09

A.5 First-Order Optimal Stability Polynomials for MUSCL Scheme

1	1	0.642751								
1	1	0.499996	0.141464							
1	1	0.512855	0.139260	0.023174						
1	1	0.499999	0.151868	0.027244	0.002898					
1	1	5.008999e-01	1.548706e-01	3.173194e-02	4.043633e-03	2.992380e-04				
1	1	4.999997e-01	1.581995e-01	3.402885e-02	5.056306e-03	4.827917e-04	2.626766e-05			
1	1	5.000574e-01	1.600544e-01	3.575068e-02	5.709207e-03	6.463868e-04	4.798128e-05	1.999902e-06		
1	1	4.999999e-01	1.614582e-01	3.691928e-02	6.203116e-03	7.687085e-04	6.870514e-05	4.079370e-06	1.343812e-07	
1	1	5.000032e-01	1.624374e-01	3.778499e-02	6.567447e-03	8.659628e-04	8.597755e-05	6.231756e-06	3.027068e-07	8.078337e-09

A.6 Second-Order Optimal Stability Polynomials for MUSCL Scheme

1	1	0.500000	0.141464							
1	1	0.500000	0.140766	0.021847						
1	1	0.500000	0.151869	0.027244	0.002898					
1	1	0.500000	1.551034e-01	3.173117e-02	4.055830e-03	2.976718e-04				
1	1	0.500000	1.581997e-01	3.402889e-02	5.056314e-03	4.827927e-04	2.626772e-05			
1	1	0.500000	1.600845e-01	3.576178e-02	5.712427e-03	6.467884e-04	4.801906e-05	2.000459e-06		
1	1	0.500000	1.614582e-01	3.691929e-02	6.203119e-03	7.687089e-04	6.870518e-05	4.079373e-06	1.343813e-07	
1	1	0.500000	1.624404e-01	3.778652e-02	6.567876e-03	8.660414e-04	8.598745e-05	6.232638e-06	3.027566e-07	8.079921e-09